Maven

Introducción

Maven es una **herramienta de construcción y gestión de proyectos** para Java. Sirve para:

- **Compilar código**.
- **Descargar dependencias** automáticamente.
- **Ejecutar pruebas**.
- **Empaquetar** en `.jar` o `.war`.
- **Publicar** el proyecto en repositorios.

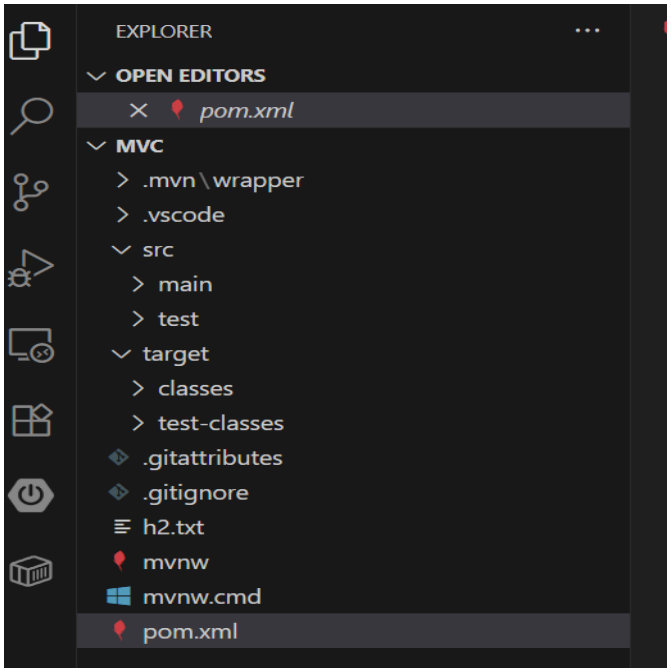
Se basa en un archivo `pom.xml` (**P**roject **O**bject **M**odel), donde defines:

- **Datos del proyecto**: nombre, versión, tipo de empaquetado.
- **Dependencias**: librerías externas que tu código necesita.
- **Plugins**: tareas adicionales que Maven ejecutará.
- **Perfiles**: configuraciones específicas para diferentes entornos (dev, test, prod).

Ventajas:

- No necesitas agregar manualmente los `.jar` al proyecto.
- Todos los miembros del equipo usan la misma configuración.
- Integración con herramientas modernas (Spring Boot, Jenkins, GitHub Actions, etc.).

Estructura típica de un proyecto Maven:



Comandos Maven

1.- Crear proyecto Maven básico

mvn archetype:generate -DgroupId=com.ejemplo -DartifactId=mi-app \ -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

→ Crea estructura estándar con pom.xml y carpetas src/main/java y src/test/java }

2.- (Opcional) **Crear pom.xml** manualmente → Archivo de configuración del proyecto (groupId, artifactId, version, dependencias)

3.- Limpiar carpeta target

mvn clean → Borra los archivos generados (compilados, empaquetados) para empezar de cero

4.- Compilar código fuente

mvn compile → Convierte .java en .class dentro de target/classes

5.- Ejecutar pruebas unitarias

mvn test → Corre todos los tests en src/test/java

6.- Empaquetar proyecto

mvn package → Compila y crea un .jar o .war en target/

7.- Instalar artefacto en repositorio local

mvn install → Guarda el .jar en ~/.m2/repository para usarlo en otros proyectos

8.- Instalar sin pruebas

mvn install -DskipTests → Igual que install pero no ejecuta los tests

9.- Ejecutar clase main

mvn exec:java -Dexec.mainClass="com.ejemplo.App" → Corre la aplicación directamente desde Maven

10.- Ejecutar proyecto Spring Boot

mvn spring-boot:run → Levanta la app Spring Boot sin empaquetar

11.- Ver árbol de dependencias

mvn dependency:tree → Lista dependencias y sus relaciones

12.- Analizar dependencias

mvn dependency:analyze → Muestra librerías usadas y no usadas

13.- Descargar dependencia específica

mvn dependency:get -Dartifact=org.springframework.boot:spring-boot-starter-web:3.2.5 → Descarga una dependencia al repo local sin agregarla al pom

14.- Ver dependencias con versiones nuevas

mvn versions:display-dependency-updates → Lista actualizaciones disponibles

15.- Ver pom efectivo

mvn help:effective-pom → Muestra pom con herencias, perfiles y propiedades resueltas

16.- Obtener valor de propiedad

mvn help:evaluate -Dexpression=project.version -q -DforceStdout → Imprime el valor de una propiedad del pom

17.- Empaquetar con perfil

mvn clean package -Pdev → Construye usando configuraciones específicas

mvn clean package -Pprod -DskipTests → Construye usando configuraciones específicas

18.- Compilar en paralelo

mvn -T 1C package → Usa todos los núcleos de CPU para acelerar build

19.- Modo debug

mvn -X clean package → Muestra salida detallada para diagnóstico

20.- Generar sitio/reporte

mvn site → Crea documentación HTML del proyecto en target/site

21.- Instalar JAR local manualmente

mvn install:install-file -Dfile=libs/mi-lib.jar -DgroupId=com.acme \ -DartifactId=mi-lib -Dversion=1.0.0 -Dpackaging=jar → Agrega un jar manual al repositorio local

22.- Limpiar caché local

mvn dependency:purge-local-repository -DreResolve=false → Borra dependencias del repo local

23.- Subir a repositorio remoto

mvn deploy → Publica artefacto en un repo remoto configurado

24.- Ciclo completo

mvn clean install → Limpia, compila, prueba, empaqueta e instala localmente

25.- Validar/verificar proyecto

mvn validate → verify: chequea calidad y pruebas

mvn verify → validate: verifica estructura;

26.- Cambiar versión del proyecto

mvn versions:set -DnewVersion=1.0.1 → Modifica versión en pom.xml

27.- Mostrar ruta del repo local

mvn help:evaluate -Dexpression=settings.localRepository -q -DforceStdout → Devuelve la ubicación de ~/.m2/repository

28.- Ejecutar integración saltando tests unitarios

mvn verify -DskipTests → Corre pruebas de integración sin unitarias

29.- Info detallada de un plugin

mvn help:describe -Dplugin=compiler -Ddetail → Muestra opciones y parámetros del plugin

30.- Ejecutar jar generado

java -jar target/mi-app-1.0.0-SNAPSHOT.jar → Corre el artefacto empaquetado