MAVEN

INTRODUCCIÓN A MAVEN

1

INTRODUCCIÓN A MAVEN

Apache Maven es una herramienta de gestión y construcción de proyectos para Java, que facilita la compilación, dependencia y administración del ciclo de vida del desarrollo. Permite automatizar procesos de construcción y simplificar el manejo de proyectos complejos.

Características:

https://maven.apache.org/

- Software libre.
- Gestión de dependencias automáticas.
- Mecanismo distribuido de distribución de librerías.
- Configuración y administración a través de archivos XML llamados Proyect Object Model –**POM**- **(**pom.xml).
- Integración con sistemas de control de versiones y herramientas de CI/CD.
- Alternativa a Ant.

INTRODUCCIÓN A MAVEN

- **Descarga**
  - https://maven.apache.org/download.cgi
- **Instalación**
  - https://maven.apache.org/install.html
- **mvn --version**

```
Apache Maven 3.9.9 (8e8579a9e76f7d015ee5ec7bfcdc97d260186937)
Maven home: C:\Apps\apache-maven-3.9.9
Java version: 22.0.2, vendor: Oracle Corporation, runtime: C:\Java\jdk-22
Default locale: es_ES, platform encoding: UTF-8
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"
```

INTRODUCCIÓN A MAVEN

- **¿Qué se puede hacer con Maven?**
  - **Compilar código fuente.**
  - **Gestionar dependencias.**
  - **Ejecutar pruebas unitarias.**
  - **Generar el archivo JAR/WAR.**
  - **Crear un repositorio local.**
  - **Ejecutar el ciclo de vida de un proyecto. Maven utiliza un conjunto de fases predefinidas para automatizar el proceso de construcción, desde la compilación hasta la instalación y despliegue.**
  - **Generar documentación del proyecto.**
  - **Generar un informe de dependencias.**
  - **Lanzar un servidor web.**
  - **Realizar un análisis estático de código.**
  - **Publicar artefactos en un repositorio remoto.**
  - **Verificar dependencias transitivas.**
  - **Gestionar perfiles de construcción.**
  - **Gestionar versiones de artefactos.**
  - **Ejecutar un proyecto en diferentes entornos: Usando perfiles, puedes ejecutar un proyecto con distintas configuraciones en diferentes entornos (desarrollo, pruebas, producción).**
  - **Realizar pruebas de integración.**
  - **Generar informes de cobertura de pruebas.**
  - **Desplegar artefactos en servidores de integración continua: Maven se integra fácilmente con herramientas de integración continua (como Jenkins) para realizar despliegues automáticos y pruebas en servidores de integración.**
  - **Limpiar el proyecto: Maven permite limpiar el directorio de salida, eliminando los archivos generados en compilaciones anteriores (por ejemplo, los archivos .class).**

ARQUETIPOS

INTRODUCCIÓN A MAVEN

- Generación de proyectos a partir de arquetipos:

```
mvn archetype:generate \
  -DgroupId=com.miempresa.proyecto \
  -DartifactId=nombre-del-proyecto \
  -DarchetypeGroupId=org.apache.maven.archetypes \
  -DarchetypeArtifactId=maven-archetype-quickstart \
  -DarchetypeVersion=1.4 \
  -DinteractiveMode=false
```

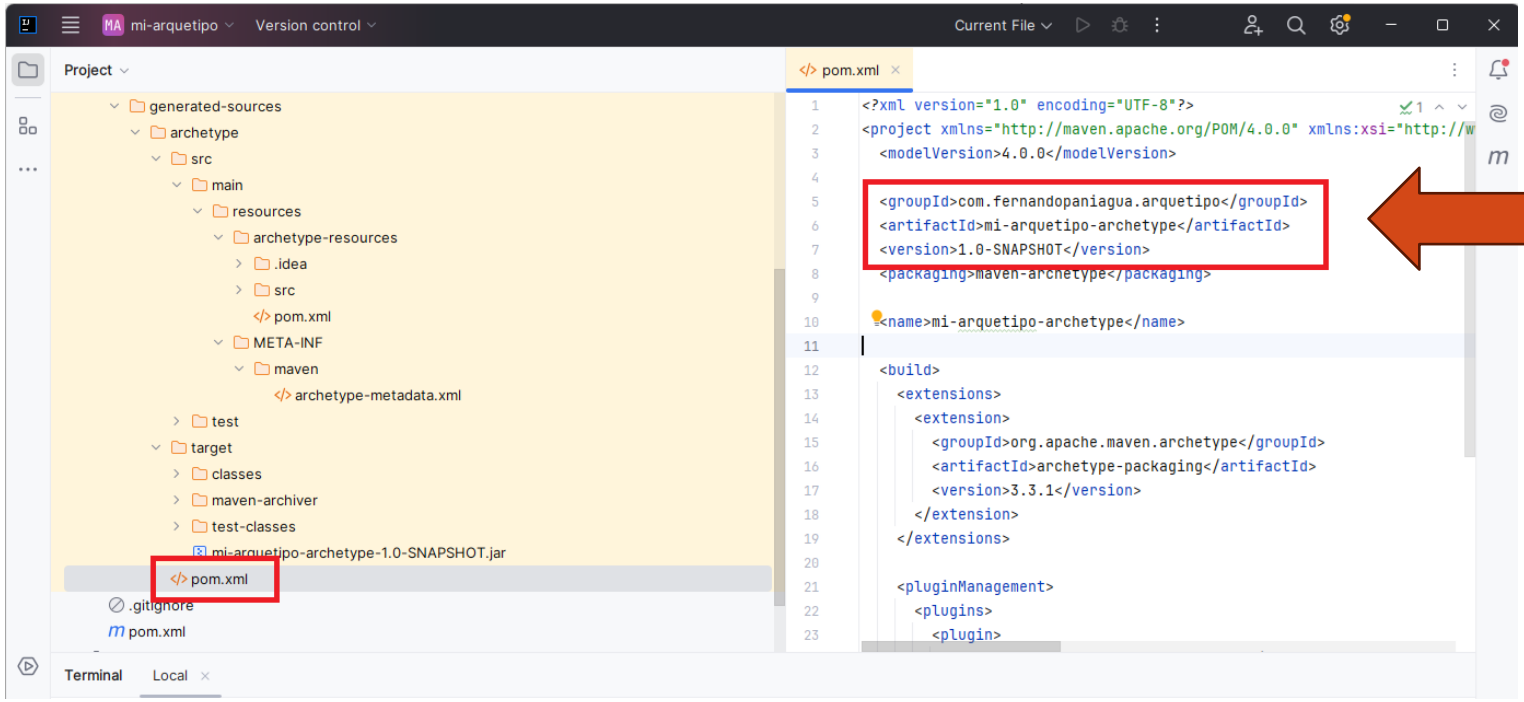
- Catálogo de arquetipos:
  - https://repo.maven.apache.org/maven2/archetype-catalog.xml

INTRODUCCIÓN A MAVEN

- Creación de arquetipos:
  - Comprobar que existe el fichero *settings.xml* en la carpeta *usuario/*.m2/.
  - Crear el proyecto base.
  - Desde la raíz del proyecto ejecutar desde un terminal:
    - mvn archetype:create-from-Project
  - Desde la carpeta del proyecto target\generated-sources\archetype ejecutar desde un terminal:
    - mvn install

INTRODUCCIÓN A MAVEN

- Creación de arquetipos:



Información del arquetipo

```
mvn archetype:generate -DarchetypeGroupId=com.fernandopaniagua.arquetipo -DarchetypeArtifactId=mi-arquetipo-archetype -DarchetypeVersion=1.0-SNAPSHOT -DgroupId=com.fernandopaniagua.elproyeto -DartifactId=elproyecto3 -Dversion=1.0-SNAPSHOT -DinteractiveMode=false
```

HOLA MUNDO

INTRODUCCIÓN A MAVEN

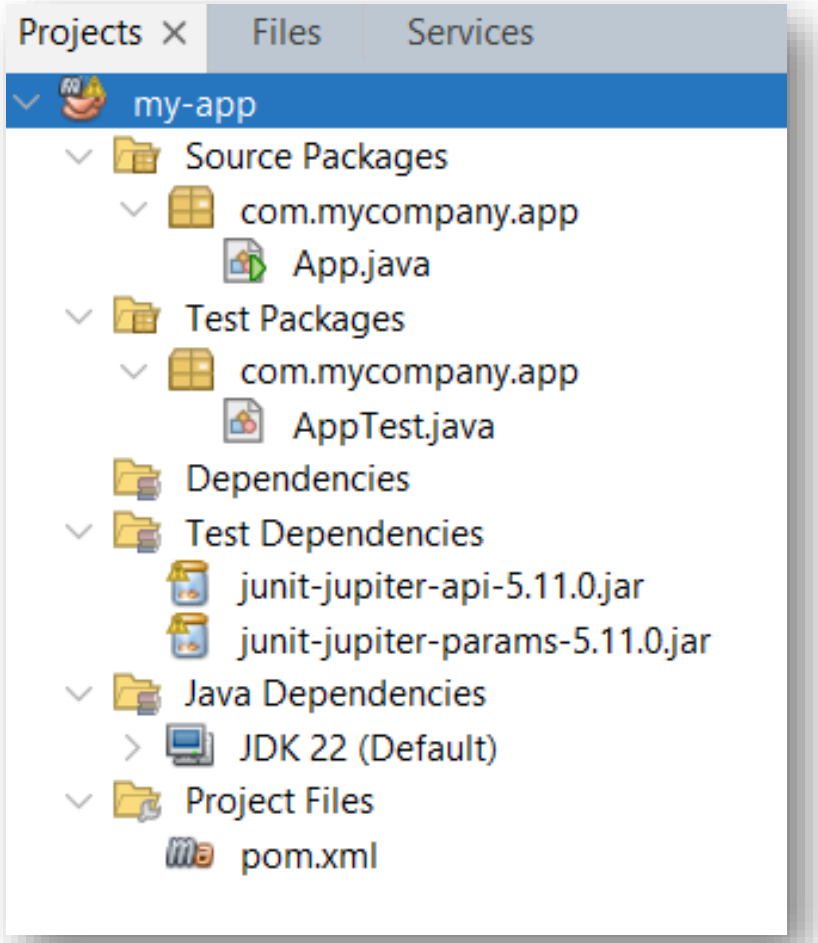
- **“Hola Mundo” versión Maven (proyecto básico):**
- `mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app -DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=1.5 -DinteractiveMode=false`

Parte	Explicación
`mvn archetype:generate`	Comando de Maven que genera un nuevo proyecto a partir de un *archetype* (plantilla predefinida).
`-DgroupId=com.mycompany.app`	Define el Group ID del proyecto, generalmente relacionado al nombre del paquete (dominio invertido de la empresa o proyecto).
`-DartifactId=my-app`	Define el Artifact ID, es decir, el nombre de la carpeta/proyecto que se creará.
`-DarchetypeArtifactId=maven-archetype-quickstart`	Indica qué plantilla (*archetype*) usar. `maven-archetype-quickstart` crea un proyecto Java básico.
`-DarchetypeVersion=1.5`	Especifica la versión de la plantilla (*archetype*) que quieres usar (`1.5`).
`-DinteractiveMode=false`	Ejecuta Maven en modo no interactivo (no hace preguntas y usa directamente los valores que le diste).

INTRODUCCIÓN A MAVEN

- **“Hola Mundo” versión Maven (proyecto básico)**

```
my-app
|-- pom.xml
`-- src
    |-- main
    |   `-- java
    |       `-- com
    |           `-- mycompany
    |               `-- app
    |                   `-- App.java
    `-- test
        `-- java
            `-- com
                `-- mycompany
                    `-- app
                        `-- AppTest.java
```



INTRODUCCIÓN A MAVEN

- **Comandos:**
  - **Eliminación de target: mvn clean**
  - **Compilación: mvn compile**
  - **Ejecución de test: mvn test**
  - **Empaquetado: mvn package**
  - **Instalación en repo local de Maven: mvn install**
    - **~/.m2/repository**
    - **~ es tu carpeta de usuario (en Linux/macOS: /home/usuario, en Windows: C:\Users\usuario)**
    - **.m2 es una carpeta oculta donde Maven guarda su configuración y el repositorio local**
  - **Despliegue: mvn deploy**
  - **Ejecución: mvn exec:java -Dexec.mainClass="com.fernandopaniagua.app.App“**
  - **Generación de documentación: mvn site**

CARACTERÍSTICAS

INTRODUCCIÓN A MAVEN

- **Terminología**
  - **Proyecto.** Es la unidad básica en Maven. Representa una aplicación, librería o módulo que se está desarrollando. Está definido por un archivo **POM**.
  - **Arquetipo.** Es una **plantilla de proyecto**. permite crear proyectos nuevos con una estructura base predefinida.
  - **Repositorio.** Es un almacén donde se guardan los artefactos (dependencias, plugins, librerías, etc.).
    - Tipos:
      - **Local**: está en la máquina local.
      - **Remoto**: servidores externos como Maven Central.
    - Al compilar, Maven busca primero en el repositorio local, y si no encuentra una dependencia, la descarga del remoto.

INTRODUCCIÓN A MAVEN

- **El archivo POM.**
  - **El archivo pom.xml es el núcleo de la configuración de un proyecto en Maven.**
  - **Es un único archivo de configuración que contiene la mayor parte de la información necesaria para construir un proyecto exactamente como se desea.**
  - **El POM es enorme y complejo.**
  - **Cada proyecto Maven incluye un fichero POM.**

INTRODUCCIÓN A MAVEN

- Estructura del fichero POM:

```
<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>

  <!-- Datos básicos del proyecto -->
  <groupId>com.ejemplo</groupId>   <!-- Identificador del grupo/empresa -->
  <artifactId>mi-proyecto</artifactId>  <!-- Nombre del proyecto (artefacto) -->
  <version>1.0.0</version>  <!-- Versión del proyecto -->
  <packaging>jar</packaging> <!-- Tipo de empaquetado (jar, war, etc.) -->
```

INTRODUCCIÓN A MAVEN

- Estructura del fichero POM (Resumen):

  - **<groupId>, <artifactId>, <version>**: identifican de manera única tu proyecto.
  - **<dependencies>**: librerías de las que depende tu proyecto.
  - **<build>**: cómo construir el proyecto (plugins de compilación, empaquetado, etc.).
  - **<properties>**: variables que puedes reutilizar en todo el POM.
  - **<repositories>**: si necesitas fuentes distintas al repositorio Maven Central.
  - **<dependencyManagement>**: controla versiones de dependencias de forma centralizada.
  - **<profiles>**: permite configuraciones para distintos entornos (desarrollo, producción...).

INTRODUCCIÓN A MAVEN

- Estructura del fichero POM (1/5):

```
<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>

  <!-- Datos básicos del proyecto -->
  <groupId>com.ejemplo</groupId>   <!-- Identificador del grupo/empresa -->
  <artifactId>mi-proyecto</artifactId>  <!-- Nombre del proyecto (artefacto) -->
  <version>1.0.0</version>  <!-- Versión del proyecto -->
  <packaging>jar</packaging> <!-- Tipo de empaquetado (jar, war, etc.) -->
```

INTRODUCCIÓN A MAVEN

- Estructura del fichero POM (2/5):

```
<!-- Información adicional -->
 <name>Mi Proyecto</name>
 <description>Un proyecto de ejemplo usando Maven</description>
 <url>https://ejemplo.com/mi-proyecto</url>

 <!-- Dependencias (librerías necesarias) -->
 <dependencies>
  <dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-web</artifactId>
   <version>2.7.5</version>
  </dependency>
  <!-- Otras dependencias -->
 </dependencies>
```

INTRODUCCIÓN A MAVEN

- Estructura del fichero POM (3/5):

```
<!-- Configuración de build (compilación) -->
 <build>
  <plugins>
   <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.8.1</version>
    <configuration>
     <source>17</source>    <!-- Versión de Java -->
     <target>17</target>
    </configuration>
   </plugin>
  </plugins>
 </build>
```

INTRODUCCIÓN A MAVEN

- Estructura del fichero POM (4/5):

```
<!-- Repositorios adicionales (si es necesario) -->
 <repositories>
  <repository>
   <id>central</id>
   <url>https://repo.maven.apache.org/maven2</url>
  </repository>
 </repositories>

 <!-- Propiedades generales -->
 <properties>
  <java.version>17</java.version>
 </properties>
```

INTRODUCCIÓN A MAVEN

- Estructura del fichero POM (5/5):

```
<!-- Gestión de dependencias (opcional) -->
<dependencyManagement>
 <dependencies>
  <!-- Versiones controladas para dependencias -->
 </dependencies>
</dependencyManagement>

<!-- Perfil para configuraciones específicas (opcional) -->
<profiles>
 <profile>
  <id>desarrollo</id>
  <properties>
   <entorno>dev</entorno>
  </properties>
 </profile>
</profiles>

</project>
```

INTRODUCCIÓN A MAVEN

- **Estructura de un proyecto Maven:**

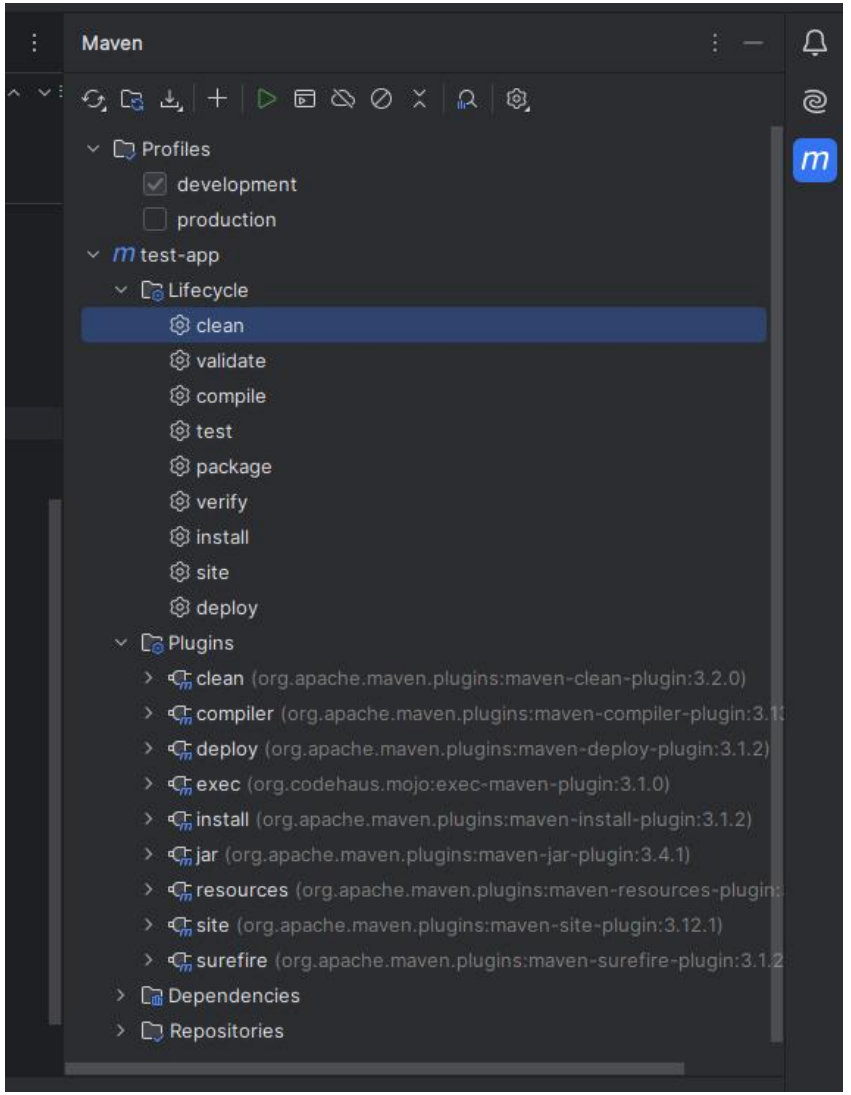
`src/main/java`	Application/Library sources
`src/main/resources`	Application/Library resources
`src/main/filters`	Resource filter files
`src/main/webapp`	Web application sources
`src/test/java`	Test sources
`src/test/resources`	Test resources
`src/test/filters`	Test resource filter files
`src/it`	Integration Tests (primarily for plugins)
`src/assembly`	Assembly descriptors
`src/site`	Site
`LICENSE.txt`	Project's license
`NOTICE.txt`	Notices and attributions required by libraries that the project depends on
`README.txt`	Project's readme

INTRODUCCIÓN A MAVEN

- Organización de Maven.
  - Ciclos de vida de construcción (“build lifecycles”)
    - Fases (“phases”)
      - Objetivos (“goals”)
- Ciclo de vida → un conjunto de fases ordenadas.
- Fase → una etapa dentro del ciclo (ej. compilar, testear, empaquetar).
- Objetivo → una tarea específica que Maven ejecuta (ej. compile, test, install).

INTRODUCCIÓN A MAVEN

- En IntelliJ hay una sección específica para Maven:



INTRODUCCIÓN A MAVEN

- **Ciclos principales de Maven:**

Ciclo de Vida	¿Qué hace?
default	Construye, prueba y empaqueta el proyecto.
clean	Limpia archivos de builds anteriores.
site	Genera documentación del proyecto.

- Los ciclos de vida **clean** y **site** se ejecutan:
  - mvn *nombre_ciclo*

INTRODUCCIÓN A MAVEN

- **Fases del ciclo default:**

Fase	¿Qué hace?
validate	Verifica que el proyecto esté bien configurado.
compile	Compila el código fuente del proyecto (`src/main/java`).
test	Compila y ejecuta los tests unitarios (`src/test/java`).
package	Empaqueta el proyecto en un `.jar`, `.war`, etc.
verify	Verifica que el paquete cumple con ciertos requisitos.
install	Instala el paquete en el repositorio local de Maven.
deploy	Sube el paquete a un repositorio remoto para compartirlo.

- Las fases se ejecutan: mvn *nombre_fase*
- Cada fase contiene a las fases anteriores

INTRODUCCIÓN A MAVEN

- **Fases del ciclo clean:**

Fase	¿Qué hace?
pre-clean	Preparativos antes de limpiar.
clean	Borra los archivos generados (target/).
post-clean	Cosas que se hacen después de limpiar.

INTRODUCCIÓN A MAVEN

- **Fases del ciclo site:**

Fase	¿Qué hace?
pre-site	Prepara para la generación de documentación.
site	Genera el sitio web/documentos del proyecto.
post-site	Ajustes después de generar la documentación.
site-deploy	Sube el sitio generado a un servidor remoto.

INTRODUCCIÓN A MAVEN

- Ejecución de fases:
  - Una única fase:
    - Ejemplo: **mvn site**
  - Un conjunto de fases en serie:
    - Ejemplo: **mvn clean install site**
    - Se ejecutan en el orden en el que aparecen.
    - Cada fase requiere que la anterior haya ido bien.

INTRODUCCIÓN A MAVEN

- Objetivos:
  - Cada fase se puede dividir en objetivos (**goals**).
  - Para ejecutar un objeto: **mvn fase:objetivo**
  - Se pueden concatenar. **mvn fase1 fase2:objetivo1 fase3:objetivo2**

DEPENDENCIAS

INTRODUCCIÓN A MAVEN

- Dependencias:
  - Son bibliotecas o componentes externos que un proyecto necesita para compilarse, ejecutarse o realizar pruebas. Maven gestiona estas dependencias automáticamente descargándolas de repositorios centralizados (como el repositorio central de Maven) y colocándolas en el proyecto para su uso.

INTRODUCCIÓN A MAVEN

- Se definen en el archivo pom.xml del proyecto, dentro del bloque <dependencies>.
  - Cada dependencia se especifica con:
    - <dependency>: Contenedor de la dependencia
      - <groupId>: El grupo o la organización del proyecto.
      - <artifactId>: El nombre del artefacto (la librería específica).
      - <version>: La versión de la librería.

INTRODUCCIÓN A MAVEN

- Existen tres tipos de ámbito de aplicación (scopes):
  - **compile**. Por defecto, si no se especifica otro scope se usará éste. Estas dependencias se utilizan en todas las tareas de compilación.
  - **provided**. Se utilizan en compilación y test, pero no en el producto final ya que estarán incluidas en la plataforma de ejecución de destino(por ejemplo, el servidor web).
  - **runtime**. Indica que la dependencia será necesaria durante la ejecución de tu aplicación, pero no durante la compilación.
  - **test**. Dependencias necesarias para los test, pero no para el producto final.
  - **system**. Similar a **provided**, pero la dependencia está disponible en la estructura de archivos (obsoleta).
  - **import**. Es un caso especial que solo se usa dentro de proyectos que utilizan el tipo de empaquetado pom. Tienen relación con el bloque <DependencyManager>

INTRODUCCIÓN A MAVEN

- Repositorio local:
  - Crear con mvn install
    - Copia el .jar en la carpeta ~/.m2/repository
  - Referencias en el pom.xml:
    - No existe diferencia en la forma de añadirla.

```
<dependencies>
    <dependency>
      <groupId>com.fernandopaniagua.testlibrary</groupId>
      <artifactId>testlibrary</artifactId>
      <version>1.0-SNAPSHOT</version>
    </dependency>
  </dependencies>
```

DependencyManagement

INTRODUCCIÓN A MAVEN

- <DependencyManagement>
  - Es una sección especial del pom.xml que sirve para controlar las versiones de las dependencias de manera centralizada sin forzar su inclusión directa.
  - <dependencyManagement> se definen dependencias y versiones.
  - Permite usar dependencias en el proyecto, módulos o subproyectos sin especificar la versión.

```
<dependencyManagement>
 <dependencies>
  <dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-web</artifactId>
   <version>2.7.5</version>
  </dependency>
 </dependencies>
</dependencyManagement>

<dependencies>
 <dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
  <!-- No necesitas especificar la versión aquí -->
 </dependency>
</dependencies>
```

TEST

INTRODUCCIÓN A MAVEN

- Test unitarios:
  - Inclusión de test.
  - Inclusión de dependencias.
  - Lanzamiento independiente: mvn test
  - Lanzamiento integrado: mvn clean package

```
<dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-api</artifactId>
        <version>5.12.2</version>
        <scope>test</scope>
    </dependency>
```

Determina en qué fase se va a utilizar la dependencia

DOCUMENTACIÓN

INTRODUCCIÓN A MAVEN

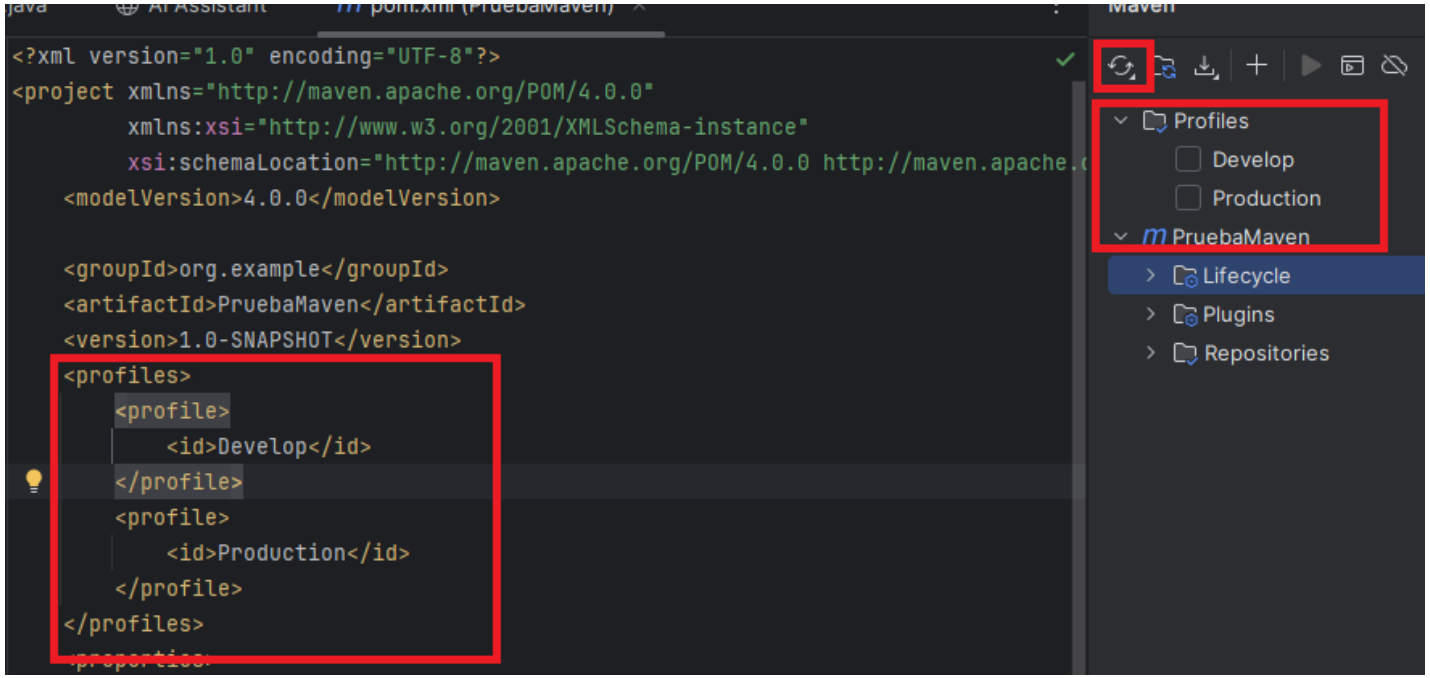
- mvn site:
  - Genera información del proyecto en la carpeta target/site
- mvn javadoc:javadoc

```
<plugin>
   <groupId>org.apache.maven.plugins</groupId>
   <artifactId>maven-javadoc-plugin</artifactId>
   <version>3.11.2</version>
</plugin>
```

PERFILES

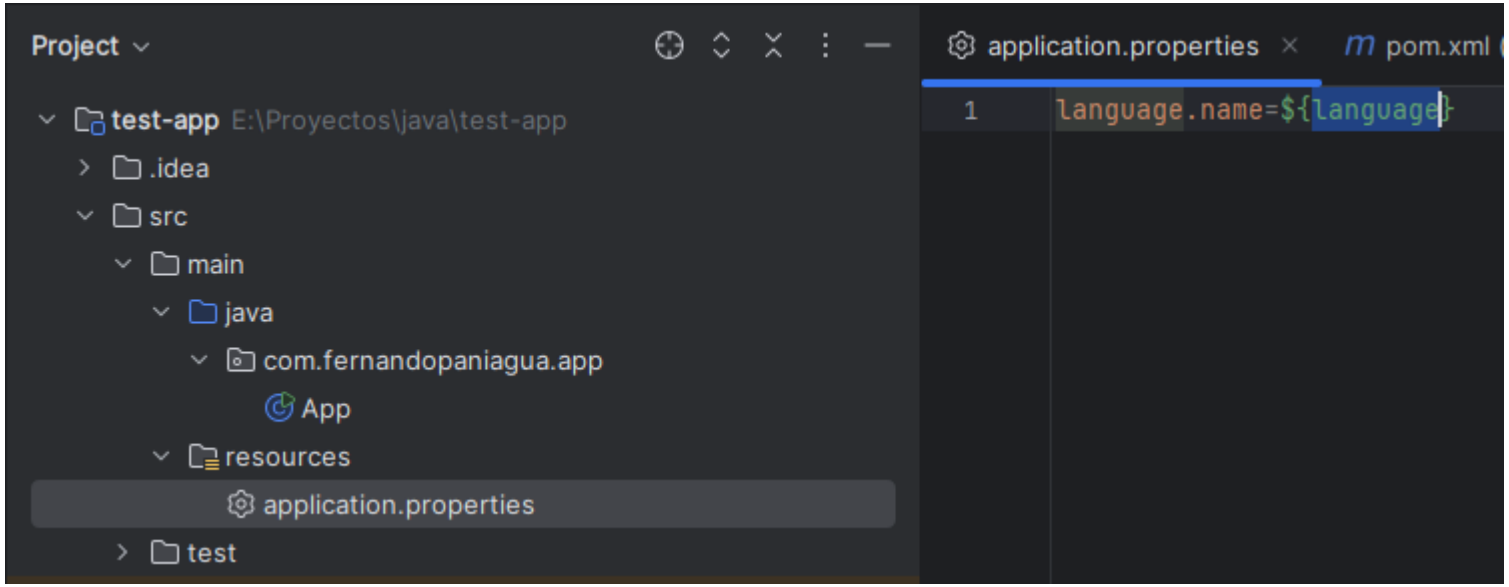
INTRODUCCIÓN A MAVEN

- Permiten definir diferentes configuraciones en un único fichero POM. Ejemplo: configuración para el entorno de desarrollo y para el de producción.



INTRODUCCIÓN A MAVEN

- Ejemplo: fichero de properties



El fichero application.properties debe estar en src/main/resources/ para que se copie en el target/classes al compilar.

INTRODUCCIÓN A MAVEN

- Ejemplo: fichero pom.xml

```
<profiles>
  <profile>
   <id>development</id>
   <properties>
    <env.tipo>dev</env.tipo>
    <language>Java</language>
   </properties>
   <activation>
    <activeByDefault>true</activeByDefault>
   </activation>
  </profile>

  <profile>
   <id>production</id>
   <properties>
    <env.tipo>prod</env.tipo>
    <language>Python</language>
   </properties>
  </profile>
</profiles>
```

Identificador del perfil

Perfil por defecto

INTRODUCCIÓN A MAVEN

- Ejemplo: fichero pom.xml

```
<build>
 <resources>
  <resource>
   <directory>src/main/resources</directory>
   <filtering>true</filtering>
  </resource>
 </resources>
</build>
```

La opción <filtering>true</filtering> dentro de un <resource> en Maven indica que se deben procesar los archivos del directorio antes de copiarlos al target/classes, haciendo un reemplazo de variables.
Maven buscará placeholders tipo ${variable} dentro de tus archivos de src/main/resources y los sustituirá automáticamente por los valores definidos en el pom.xml.

INTRODUCCIÓN A MAVEN

- Ejemplo: lectura de propiedad

```
Properties props = new Properties();
try (InputStream input = App.class.getClassLoader().getResourceAsStream("application.properties")) {
        props.load(input);
} catch (IOException e) {
        throw new RuntimeException(e);
}
String valor = props.getProperty("language.name");
System.out.println("Valor de config.valor: " + valor);
```

INTRODUCCIÓN A MAVEN

- Ejemplo: Si se quiere que en la propia ejecución se elija uno u otro perfil:

```
<build>
  <plugins>
   <plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>exec-maven-plugin</artifactId>
    <version>3.1.0</version>
    <configuration>
     <mainClass>com.fernandopaniagua.app.App</mainClass>
    </configuration>
   </plugin>
  </plugins>
```

mvn clean package –Pproduction

mvn exec:java -Pproduction

INTRODUCCIÓN A MAVEN

- Ejemplo: Profiles con distintas configuraciones y librerías

```
<profiles>
    <profile>
      <id>sqlite</id>
      <activation>
        <activeByDefault>true</activeByDefault>
      </activation>
      <properties>
        <db.type>sqlite</db.type>
        <db.url>jdbc:sqlite:E:/Proyectos/java/data/aves.sqlite</db.url>
        <db.user></db.user>
        <db.password></db.password>
      </properties>
      <dependencies>
        <!-- https://mvnrepository.com/artifact/org.xerial/sqlite-jdbc -->
        <dependency>
          <groupId>org.xerial</groupId>
          <artifactId>sqlite-jdbc</artifactId>
          <version>3.49.1.0</version>
        </dependency>
      </dependencies>
    </profile>
    <profile>
      <id>postgre</id>
      <properties>
        <db.type>postgre</db.type>
        <db.url>jdbc:postgresql://localhost:5432/aves</db.url>
        <db.user>user-aves</db.user>
        <db.password>user-aves</db.password>
      </properties>
      <dependencies>
        <!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->
        <dependency>
          <groupId>org.postgresql</groupId>
          <artifactId>postgresql</artifactId>
          <version>42.7.5</version>
        </dependency>
      </dependencies>
    </profile>
  </profiles>
```

PLUGINS

INTRODUCCIÓN A MAVEN

- Maven es, en esencia, un *framework* de ejecución de plugins.
- Existen dos tipos:
  - De compilación: deben ir dentro del elemento **<build>**
  - De informes: deben ir dentro del elemento **<reporting>**
- Se pueden crear los propios plugin.
- Listado de plugins:
  - https://repo.maven.apache.org/maven2/org/apache/maven/plugins/

INTRODUCCIÓN A MAVEN

- Plugins:
  - https://maven.apache.org/plugins/

Plugin	Type*	Version	Release Date	Description
Core plugins				**Plugins corresponding to default core phases (ie. clean, compile). They may have multiple goals as well.**
clean	B	3.4.1	2025-02-15	Clean up after the build.
compiler	B	3.14.0	2025-02-17	Compiles Java sources.
deploy	B	3.1.4	2025-02-23	Deploy the built artifact to the remote repository.
failsafe	B	3.5.3	2025-03-31	Run the JUnit integration tests in an isolated classloader.
install	B	3.1.4	2025-02-24	Install the built artifact into the local repository.
resources	B	3.3.1	2023-03-21	Copy the resources to the output directory for including in the JAR.
site	B	3.21.0	2024-10-18	Generate a site for the current project.
surefire	B	3.5.3	2025-03-31	Run the JUnit unit tests in an isolated classloader.
verifier	B	1.1	2015-04-14	Useful for integration tests - verifies the existence of certain conditions.

INTRODUCCIÓN A MAVEN

- Plugins más utilizados:

Plugin	Función
maven-compiler-plugin	Compila el código fuente Java.
maven-surefire-plugin	Ejecuta pruebas unitarias durante la fase de test.
maven-jar-plugin	Crea archivos .jar del proyecto.
maven-assembly-plugin	Empaqueta aplicaciones con sus dependencias (por ejemplo, JAR con dependencias).
maven-surefire-report-plugin	Genera reportes HTML de pruebas unitarias.
maven-failsafe-plugin	Ejecuta pruebas de integración (separa de unitarias).
maven-site-plugin	Genera un sitio web estático con documentación del proyecto (basado en POM).
maven-dependency-plugin	Inspecciona y gestiona dependencias (copiar, listar, analizar conflictos).
maven-install-plugin	Instala el artefacto generado en el repositorio local de Maven.
maven-deploy-plugin	Publica (deploy) el artefacto a un repositorio remoto (Nexus, Artifactory, etc.).

INTRODUCCIÓN A MAVEN

- Plugins:
  - exec-maven-plugin → Ejecutar proyecto (mvn exec)
  - maven-compiler-plugin → Compilar código Java (mvn compile)
  - maven-surefire-plugin → Ejecutar tests unitarios (mvn test)
  - maven-jar-plugin → Crear un .jar empaquetado (mvn package)
  - maven-install-plugin → Instalar el artefacto en tu repo local (mvn install)
  - spring-boot-maven-plugin → Para aplicaciones Spring Boot (mvn spring-boot:run)

MÓDULOS

INTRODUCCIÓN A MAVEN

- Maven permite organizar proyectos en módulos.
- Los módulos pueden estar jerarquizados y compartir dependencias del POM superior.
- POM general:

```
<groupId>com.fernandopaniagua.demomodules</groupId>
<artifactId>demo-modules</artifactId>
```

- POM módulo:

```
<modelVersion>4.0.0</modelVersion>
<parent>
    <groupId>com.fernandopaniagua.demomodules</groupId>
    <artifactId>demo-modules</artifactId>
    <version>1.0-SNAPSHOT</version>
</parent>

<artifactId>main-module</artifactId>
```

INTRODUCCIÓN A MAVEN

- Herencia de dependencias.
  - No se heredan automáticamente.
  - Se declaran en el POM principal como <dependencyManagement>

```
<dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>com.fernandopaniagua.testlibrary</groupId>
        <artifactId>testlibrary</artifactId>
        <version>1.0-SNAPSHOT</version>
      </dependency>
    </dependencies>
  </dependencyManagement>
```

  - Se declaran en los POM de los módulos como <dependency> sin indicar la versión.

```
<dependencies>
    <dependency>
      <groupId>com.fernandopaniagua.testlibrary</groupId>
      <artifactId>testlibrary</artifactId>
    </dependency>
  </dependencies>
```

INTRODUCCIÓN A MAVEN

- Ejecución de los módulos.
  - El código de arranque la aplicación se encuentra en uno de los módulos.
  - Los módulos se referencian entre sí como dependencias comunes. Requiere ejecutar mvn install para que en ejecución sean visibles.

```
<dependency>
        <groupId>com.fernandopaniagua.demomodules</groupId>
        <artifactId>aux-module</artifactId>
        <version>1.0-SNAPSHOT</version>
        <scope>compile</scope>
</dependency>
```

  - Hay que realizar el mvn exec:java desde la carpeta del módulo principal.

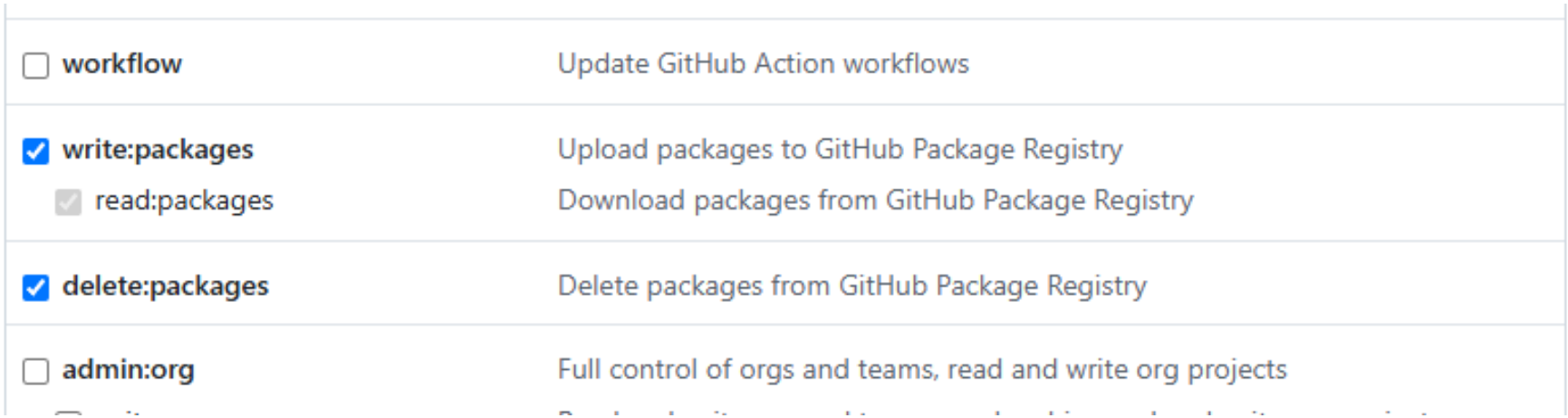
Publicación de paquetes en GitHub Packages

INTRODUCCIÓN A MAVEN

- GitHub Packages es un servicio de alojamiento de paquetes de software que le permite alojar sus paquetes de software de forma privada o pública y utilizar paquetes como dependencias en sus proyectos.
- https://docs.github.com/es/packages/learn-github-packages/introduction-to-github-packages

INTRODUCCIÓN A MAVEN

- Crear el repositorio en GitHub.
- Obtener un token "classic" con los siguientes permisos:



INTRODUCCIÓN A MAVEN

- Configurar fichero settings.xml
  - Versión global, en la carpeta **conf** de la instalación de Maven.
  - Versión de usuario, en la carpeta ~/.m2/settings.xml (Linux/MacOS) o C:\Users\<tu_usuario>\.m2\settings.xml (Windows).

```
<settings>
  <servers>
    <server>
      <id>github</id>
      <username>identificador-cuenta-github</username>
      <password>token-github</password>
    </server>
  </servers>
</settings>
```

Debe coincidir con el id del fichero pom.xml

INTRODUCCIÓN A MAVEN

- En el fichero pom.xml:

```
<distributionManagement>
    <repository>
      <id>github</id>
      <name>GitHub OWNER Apache Maven Packages</name>
      <url>https://maven.pkg.github.com/fpaniaguajava/demo-githubpackages</url>
    </repository>
  </distributionManagement>
```

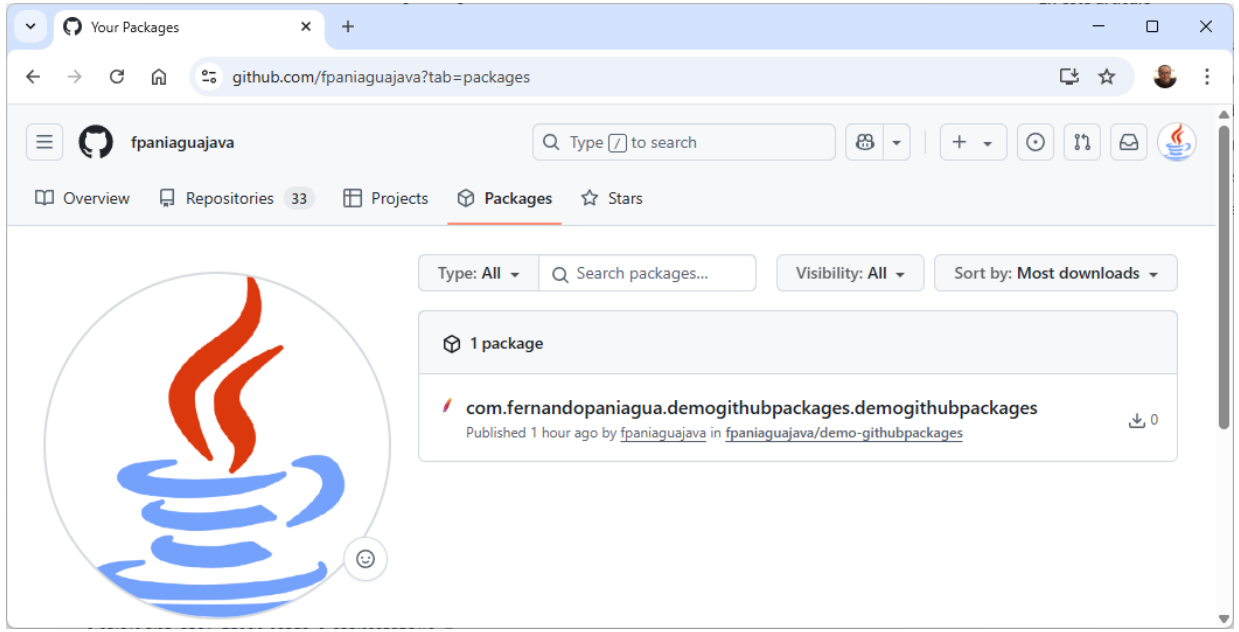
Debe coincidir con el id del fichero settings.xml

Identificador de la cuenta de GitHub

Identificador del repositorio

INTRODUCCIÓN A MAVEN

- Ejecutar el despliegue mediante la fase deploy:
  - mvn clean package deploy
- El paquete debe aparecer en la sección “packages” de la cuenta de GitHub.



INTRODUCCIÓN A MAVEN

- Proyecto **"consumidor"**:
  - Se requiere un token con permisos read:packages
  - Fichero pom.xml:

```
<repositories>
   <repository>
      <id>github</id>
      <url>https://maven.pkg.github.com/fpaniaguajava/demo-githubpackages</url>
   </repository>
 </repositories>

 <dependencies>
      <dependency>
         <groupId>com.fernandopaniagua.demogithubpackages</groupId>
         <artifactId>demogithubpackages</artifactId>
         <version>1.0-SNAPSHOT</version>
      </dependency>
   </dependencies>
```