

Maven

Conceptos generales



ORACLE®



Maven™

COMPETENCIA DE MAVEN



Gradle Build Tool



Herramientas de BUILD

Tasks	maven			 gradle			 ivy		
IDEs	Eclipse	IDEA	NetBeans	Eclipse	IDEA	NetBeans	Eclipse	IDEA	NetBeans
Import a project build structure into IDE	yes	yes	yes	yes	yes	yes	yes	yes	yes
Submit a build from IDE	yes	yes	yes	yes	yes	yes	yes	yes	yes
Dependency management in IDE	yes	yes	yes	yes	yes	yes	yes	yes	yes
Automatic download of dependencies	yes	yes	yes	yes	yes	yes	yes	yes	yes
Support/Wizard to create build scripts	yes	yes	yes	yes	yes	yes	yes	yes	yes
Enablement	plugin	built-in	built-in	plugin	built-in	plugin	Ant: built-in Ivy: plugin	Ant: built-in Ivy: plugin	Ant: built-in Ivy: plugin

HISTORIA

- Proyecto de Apache
 - <https://maven.apache.org/>
- Que es?
 - Una herramienta de gerenciamiento y comprension de proyectos de software
 - Involucra la aplicacion de **patrones** para construir la infraestructura de un Proyecto
 - Sus objetivos principales son :
 - Facilitar el proceso de construccion
 - Proveer un Sistema de construccion uniforme
 - Proveer una guia para la major practica de desarrollo
 - Permitir transparencia en la migracion a nuevas características
 - Maven ofrece
 - Una serie de estandares de construccion
 - Un modelo de repositorio de artefactos
 - Un motor que describe proyectos
 - Un estandar de ciclo de vida para construir, testear y desplegar los artefactos del proyecto



HISTORIA

- Origen
 - Maven en Yiddish, significa “ACUMULADOR DE CONOCIMIENTO”
- Que es?
 - Inicialmente fue un intento de simplificar los procesos de construccion en el Proyecto de Jakarta Turbine
 - Su antecesor → ANT
 - **Lanzamiento inicial:** 30 de marzo de 2002
- PRINCIPIOS
 - Convencion por sobre Configuracion
 - Ejecucion Declarativa
 - Reusabilidad
 - Organizacion coherente de dependencia
 - Foco en la escritura de las aplicaciones



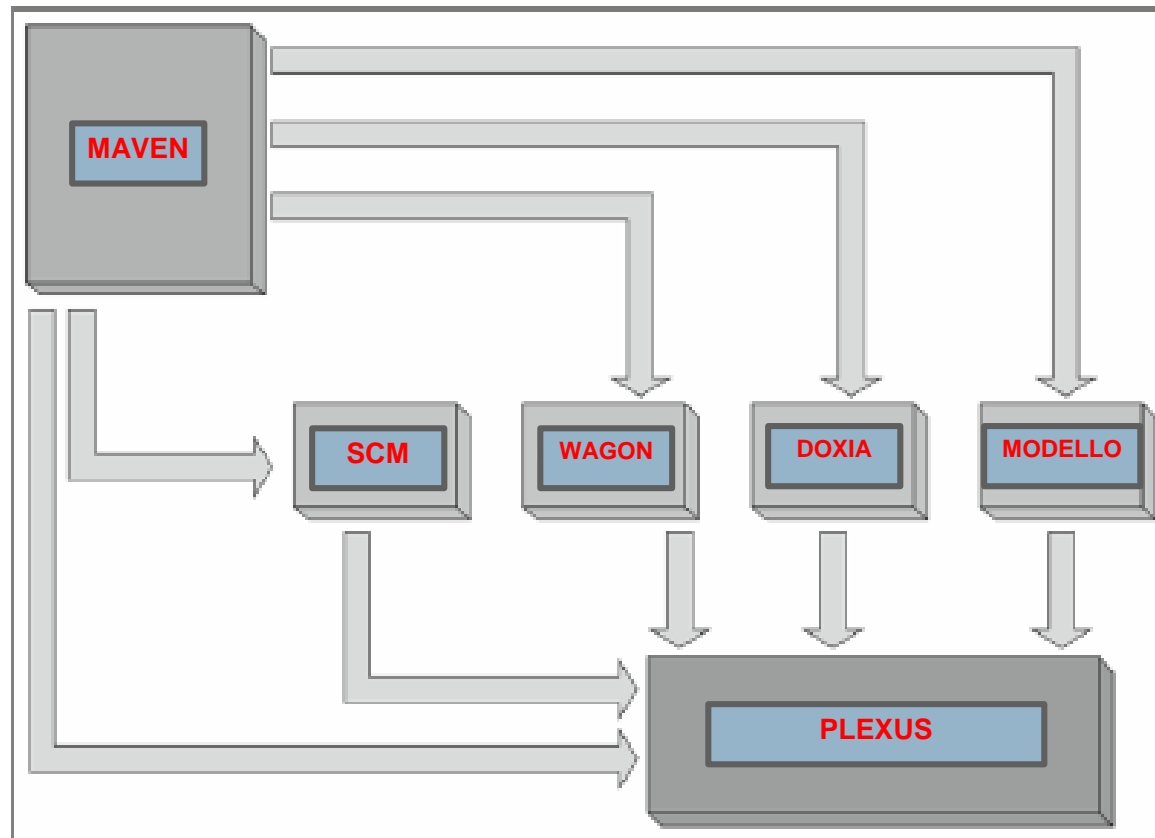
CICLO DE VIDA

Cada uno de estos ciclos de vida de compilación se define mediante una lista diferente de fases de compilación, en la que una fase de compilación representa una etapa del ciclo de vida

- **validate** - validar que el proyecto es correcto y que toda la información necesaria está disponible
- **compile** - compila el código fuente del proyecto
- **test** - probar el código fuente compilado utilizando un marco de prueba unitario adecuado. Estas pruebas no deberían requerir que el código esté empaquetado o implementado
- **package** - tomar el código compilado y empaquetarlo en su formato distribuible, como un JAR.
- **verify** - ejecutar las comprobaciones de los resultados de las pruebas de integración para garantizar que se cumplan los criterios de calidad
- **install** - instale el paquete en el repositorio local, para usarlo como una dependencia en otros proyectos localmente
- **deploy** - hecho en el entorno de compilación, copia el paquete final en el repositorio remoto para compartirlo con otros desarrolladores y proyectos.



ARQUITECTURA DE MAVEN



WAGON

Es una abstraccion de transporte usada en un artefacto Maven y código de manejo del repositorio.

Wagon define una API unificada y actualmente tiene los siguientes proveedores:

- **File**
- **HTTP**
- **HTTP liviano**
- **FTP**
- **SSH/SCP**
- **WebDav**

MAVEN ARQUITECTURA

DOXIA

- es un marco de generación de contenido que proporciona a los usuarios técnicas poderosas para generar contenido estático y dinámico.
- Doxia también se utiliza en un contexto de una publicación basada en web para generar sitios estáticos, además de incorporarse en contenido dinámico de blogs, wikis y sistemas de gestión de contenido.

MODELLO

- El componente Modello en Maven se puede utilizar para generar los siguientes tipos de artefactos en el momento de la construcción con referencia al modelo de datos:
 - Java POJOs del data model (POJO = PLAN OLD JAVA OBJECTS)
 - Java POJOs a XML
 - XML ta Java POJOs
 - Xdoc documentacon del data model
 - XML schema para validar que el contenido XML matchee con el data model

POJO = BEAN = VALUE OBJECT

PLEXUS

Plexus es un contenedor **IOC** que proporciona programación orientada a componentes para construir componentes modulares y reutilizables que se pueden ensamblar y reutilizar fácilmente.

Algunos de las características admitidas son las siguientes:

- Ciclos de vida de los componentes
- Estrategias de creación de instancias de componentes
- Contenedores anidados
- Configuración de componentes
- Cableado automático
- Dependencias de componentes
- Varias técnicas de **inyección de dependencia**, incluida la inyección de constructor, inyección de setters e inyección de campo privado

IOC = inversión of control

MAVEN SCM

SCM → source code management, prove un API para las operaciones de código.

Los siguientes tipos de scm son soportados por Maven:

- Bazaar
- CVS
- Git
- Jazz
- Mercurial
- Perforce
- StarTeam
- Subversion
- CM energy



Topics

Configuración de Maven

- Artefact_id
- Group_id
- Version
- Scopes

El pom.xml

Repositorios

- Locales
- Remotos

Comandos de maven

- Clean
- Compile
- Install
- Build
- Deploy
- Test

Artefactos

- Que son Artefactos
- Para que sirven los Artefactos

Arquetipos

- Que son Arquetipos
- Construcción de Arquetipos

Plugins

- Que son Plugins
- Creación de Plugins
- Ejecución de plugins



Introducción

Maven es una herramienta de gestión de proyectos. **Se basa en un fichero central, pom.xml, donde se define todo lo que necesita tu proyecto.** Maven maneja las dependencias del proyecto, compila, empaqueta y ejecuta los test. **Mediante plugins, permite hacer mucho mas, como por ejemplo generar los mapas de Hibernate a partir de una base de datos, desplegar la aplicación, etc...**

“Lo mas útil de Maven en mi opinión, es el manejo de las dependencias. Aun recuerdo cuando empezaba a trabajar en Java. Tenías que ir bajándote manualmente los jar que necesitabas en tu proyecto y copiarlos manualmente en el classpath. Era muy tedioso. Con Maven esto se acabó. Solo necesitas definir en tu pom.xml las dependencias que necesitas y maven las descarga y las añade al classpath.”

Instalando Maven

Maven normalmente ya viene instalado en cualquier distribución popular de linux o al menos, está disponible en los repositorios oficiales de la distribución. En otros sistemas, como MacOSX, tambien esta instalado por defecto.

Para usuarios de Windows, hay que descomprimir el fichero zip donde mejor les sea conveniente. Luego necesitamos que crear la variable de entorno M2_HOME apuntando al directorio donde descomprimimos maven y añadir M2_HOME al path del sistema para poder ejecutar maven desde la consola.

Maven puede ser usado desde la consola, aunque también existe un plugin para eclipse (m2Eclipse)

La primera vez que ejecutemos maven, creará un repositorio local en tu disco duro. En concreto, creará la carpeta *.m2* en la carpeta *home* del usuario. En ella se guardarán todos los artefactos que maneje maven.

Apache Maven. Instalación

- Tener instalado el JDK de Java a utilizar.
- Descargar la versión de Maven (archivos binarios en zip) desde <http://maven.apache.org/download.cgi>.
- Descomprimir en la ruta deseada por el usuario.
- Configurar variables de entorno M2_HOME y MAVEN_HOME a la carpeta descomprimida.
- Actualizar variable de entorno PATH a la carpeta bin.

Apache Maven. Instalación

Nueva variable del sistema

Nombre de la variable:

Valor de la variable:

Nueva variable del sistema

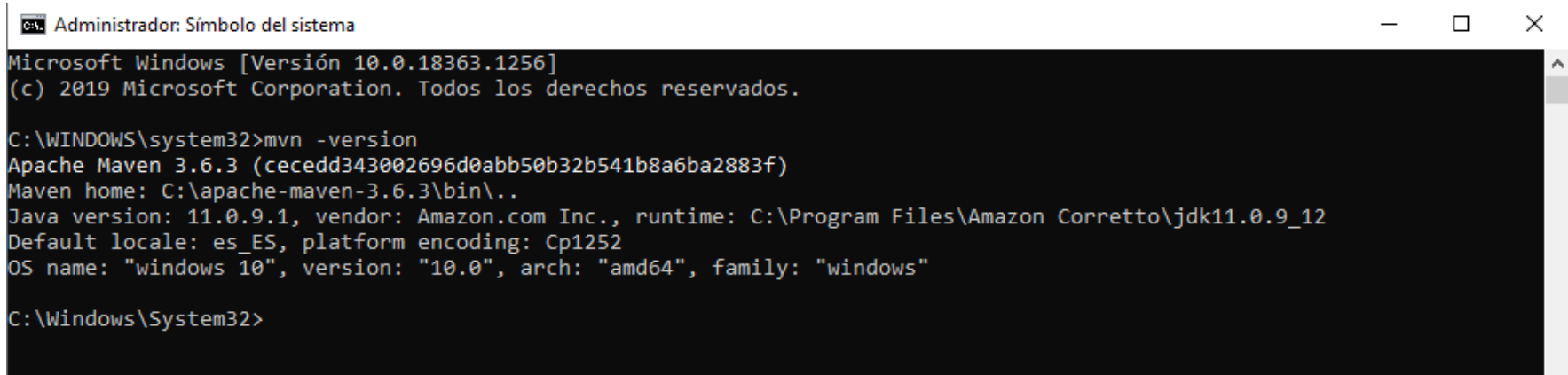
Nombre de la variable:

Valor de la variable:

Editar variable de entorno

Apache Maven. Instalación

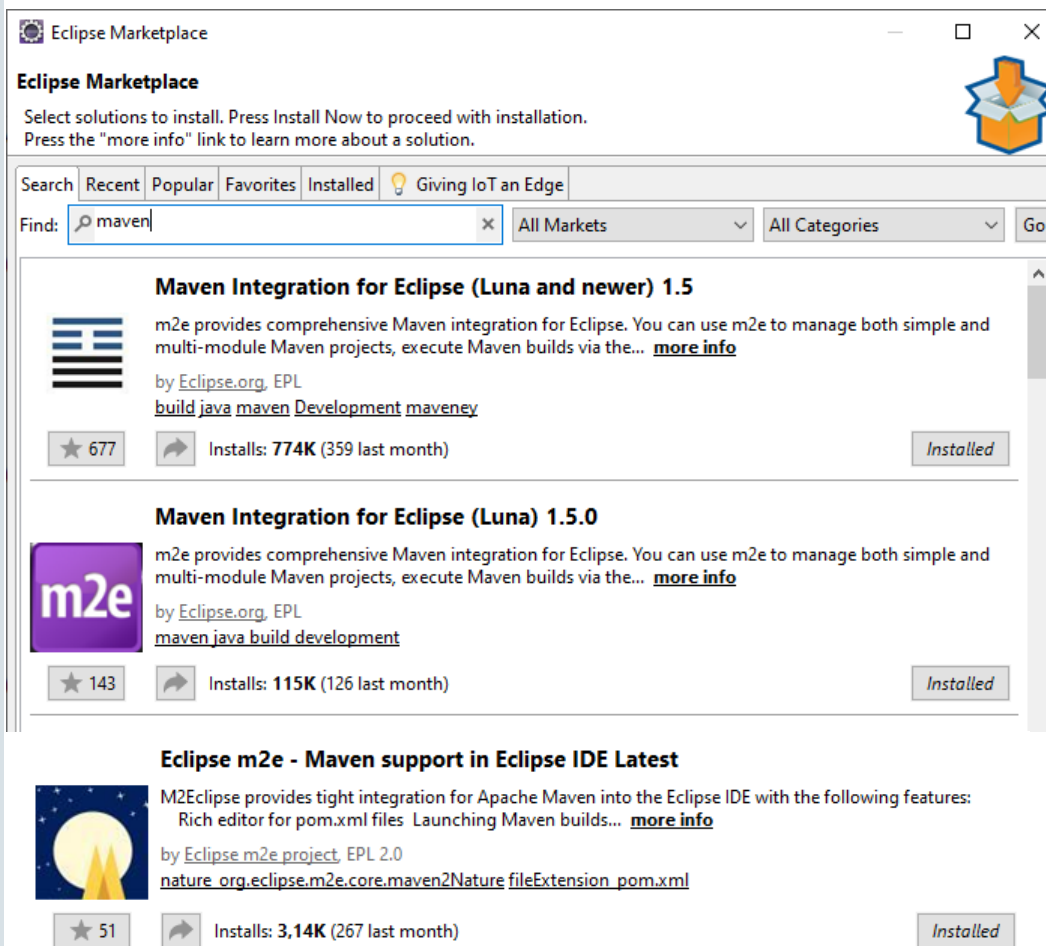
- **Comprobar instalación de Maven en terminal (ventana de comandos)**
 - **mvn -version**



```
Administrador: Símbolo del sistema
Microsoft Windows [Versión 10.0.18363.1256]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\WINDOWS\system32>mvn -version
Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)
Maven home: C:\apache-maven-3.6.3\bin\..
Java version: 11.0.9.1, vendor: Amazon.com Inc., runtime: C:\Program Files\Amazon Corretto\jdk11.0.9_12
Default locale: es_ES, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"

C:\Windows\System32>
```



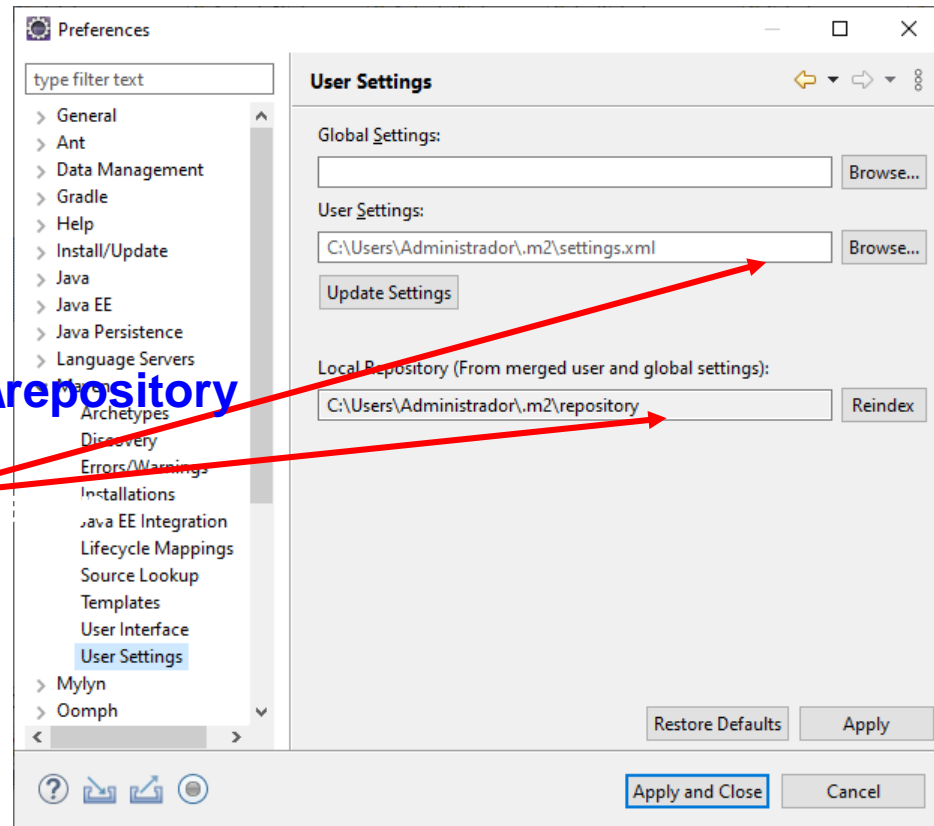
INSTALACION

Chequear en Eclipse, en el marketplace, accediendo desde el menu Ayuda:



Para cambiar el PATH del repositorio del MAVEN embebido al que instalamos ir a

C:\Users\Administrador\.m2\repository



Grupos y artefactos



Grupos y Artefactos

Un artefacto es un componente de software que podemos incluir en un proyecto como dependencia. **Normalmente será un jar, pero podría ser de otro tipo, como un war por ejemplo. Los artefactos pueden tener dependencias entre sí, por lo tanto, si incluimos un artefacto en un proyecto, también obtendremos sus dependencias.**

Un grupo es un conjunto de artefactos. **Es una manera de organizarlos. Así por ejemplo todos los artefactos de Spring Framework se encuentran en el grupo org.springframework.**

Esta es la manera de declarar una dependencia de nuestro proyecto con un artefacto. Se indica el identificador de grupo, el identificador del artefacto y la versión. **EJ:**

```
<dependency>  
    <groupid>org.springframework</groupid>  
    <artifactid>spring-orm</artifactid>  
    <version>3.0.5.RELEASE</version>  
    <scope>runtime</scope>  
</dependency>
```

Scope (alcance)



Scope (Alcance)

El *scope* sirve para indicar el alcance de nuestra dependencia y su transitividad. Hay 6 tipos:

compile: es la que tenemos por defecto sino especificamos *scope*. Indica que la dependencia es necesaria para compilar. La dependencia además se propaga en los proyectos dependientes.

provided: Es como la anterior, pero esperas que el contenedor ya tenga esa librería. Un claro ejemplo es cuando desplegamos en un servidor de aplicaciones, que por defecto, tiene bastantes librerías que utilizaremos en el proyecto, así que no necesitamos desplegar la dependencia.

runtime: La dependencia es necesaria en tiempo de ejecución pero no es necesaria para compilar.

test: La dependencia es solo para testing que es una de las fases de compilación con maven. JUnit es un claro ejemplo de esto.

system: Es como provided pero tienes que incluir la dependencia explícitamente. Maven no buscará este artefacto en tu repositorio local. Habrá que especificar la ruta de la dependencia mediante la etiqueta `<systemPath>`

import: este solo se usa en la sección *dependencyManagement*.

Goals



Goals

¿Qué es un Goal?

Un Goal no es mas que un comando que recibe maven como parámetro **para que haga algo.**

La sintaxis sería:

mvn plugin:comando

Goals

Maven tiene una arquitectura de plugins, para poder ampliar su funcionalidad, aparte de los que ya trae por defecto.

Ejemplos de goals serían:

mvn clean:clean (o mvn clean): limpia todas las clases compiladas del proyecto.

mvn compile: compila el proyecto

mvn package: empaqueta el proyecto (si es un proyecto java simple, genera un jar, si es un proyecto web, un war, etc...)

mvn install: instala el artefacto en el repositorio local (/Users/home/.m2)

Un ejemplo de un goal de un plugin externo, sería el plugin de hibernate 3.X:

mvn hibernate3:hbm2hbmxml En este caso el goal es hbm2hbmxml, que genera los mapas de hibernate a partir de una base de datos dada.

Los plugins son artefactos, y se incluyen en el pom como <plugin>, pero esto es materia para otro curso.

Archetypes



Un archetype, traducido arquetipo, es, una plantilla. **Cuando creamos un proyecto, como ahora veremos, tenemos que especificar uno. Un arquetipo crea la estructura del proyecto, el contenido del pom.xml, la estructura de carpetas y los ficheros que incluye por defecto.**

ARQUETIPOS

QUE SON ?

Cuando creamos un proyecto de Java, necesitamos estructurarlo de diferentes maneras en función de la tipo de proyecto.

Si es una aplicación web Java EE, entonces necesitamos tener un WEB-INF directorio y un archivo web.xml.

Si es un proyecto de complemento de Maven, necesitamos tener un Mojo clase que se extiende desde `org.apache.maven.plugin.AbstractMojo`.

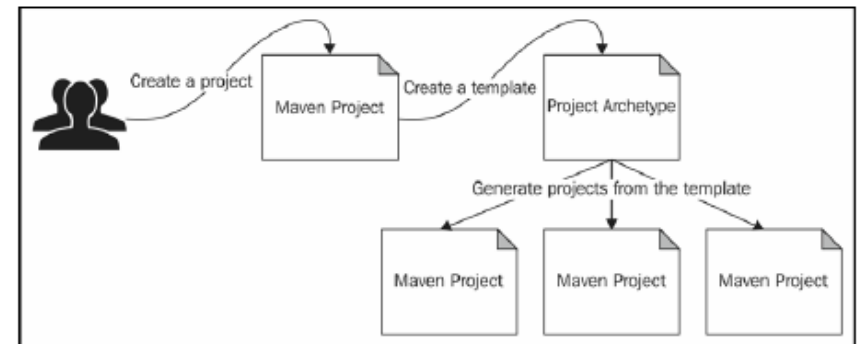
Como cada tipo de proyecto tiene su propia estructura predefinida,

¿por qué todos tendrían que construir el mismo estructura una y otra vez?

¿Por qué no empezar con una plantilla?

Cada proyecto puede tener su propia plantilla, y los desarrolladores pueden ampliar la plantilla para adaptarse a sus requisitos.

Los arquetipos de Maven abordan esta preocupación. Cada arquetipo es una plantilla de proyecto.



A list of Maven archetypes can be found at
<http://maven-repository.com/archetypes>.

ARQUETIPOS

1. Un archetype de maven es un plugin en si mismo
2. El objetivo del comando GENERATE, es justamente crear un Proyecto Maven desde un arquetipo

```
mvn archetype:generate
```

- groupId=com.curso.ejemplos
- artifactId=ejemplos
- version=1.0.0
- interactiveMode=false



Diseño con Arquetipos

New Maven Project

New Maven project

Select project name and location

☐ Create a simple project (skip archetype selection)

☒ Use default Workspace location

New Maven Project

New Maven project

Specify Archetype parameters

Group Id:

Artifact Id:

Version:

Package:

Properties available from archetype:

Name	Value
name	app2
description	null
shaded	true

Advanced

Back Next

NO CLICKEAR

New Maven Project

New Maven project

Select an Archetype

Catalog:

Filter:

Group Id	Artifact Id	Version
am.ik.archetype	graalvm-springmvc-blank-archetype	0.1.2
am.ik.archetype	mvc-1.0-blank-archetype	1.0.0-m02
co.ntier	spring-mvc-archetype	1.0.2
com.bernardomg.maven.archetypes	spring-mvc-angular-archetype	1.0.2
com.bernardomg.maven.archetypes	spring-mvc-react-archetype	1.2.6
com.bernardomg.maven.archetypes	spring-mvc-thymeleaf-archetype	1.2.5
com.commercetools.maven-archetypes	commercetools-spring-mvc-quickstart	0.4.0

A Maven Archetype for projects using Spring MVC, and Angular.
<https://repo1.maven.org/maven2>

☒ Show the last version of Archetype only ☐ Include snapshot archetypes

Advanced

Back Next Finish Cancel

Creando un proyecto con Maven...



Creando un proyecto con Maven...

Vamos a crear nuestro primer proyecto Maven.

Si quisieramos hacerlo desde consola:

```
mvn archetype:generate -DgroupId=com.empresa.escuelita -  
DartifactId=maven-jar-sample
```

**En este caso vamos a ver un ejemplo desde el plugin de eclipse
(m2Eclipse)**

Estructura de proyectos



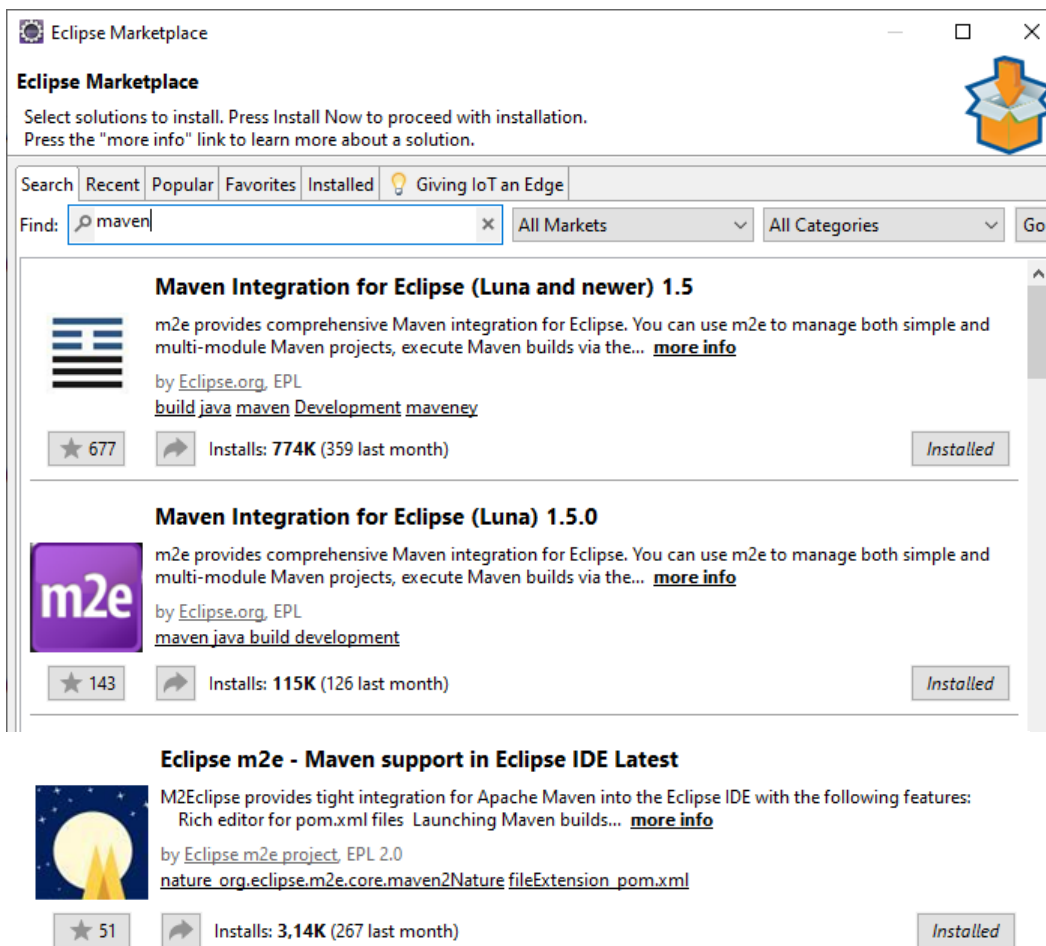
Estructura de proyectos Maven

Dependerá del tipo de proyecto con el que trabajemos, pero tienen cosas en común. Echémosle un vistazo:



Tenemos nuestro pom.xml, donde se configura el proyecto maven y una carpeta src donde se colocan los fuentes. Dentro de src, tenemos dos carpetas: main, donde va todo el código de nuestro proyecto, y test, donde va el código de test. En este caso, como es un proyecto java simple, dentro de main solo tenemos la carpeta java. En ella ya podemos colocar nuestros paquetes y clases java.

En caso de utilizar otros arquetipos, como maven-archetype-webapp, tendríamos otras carpetas además de la de java, la de resources y web.



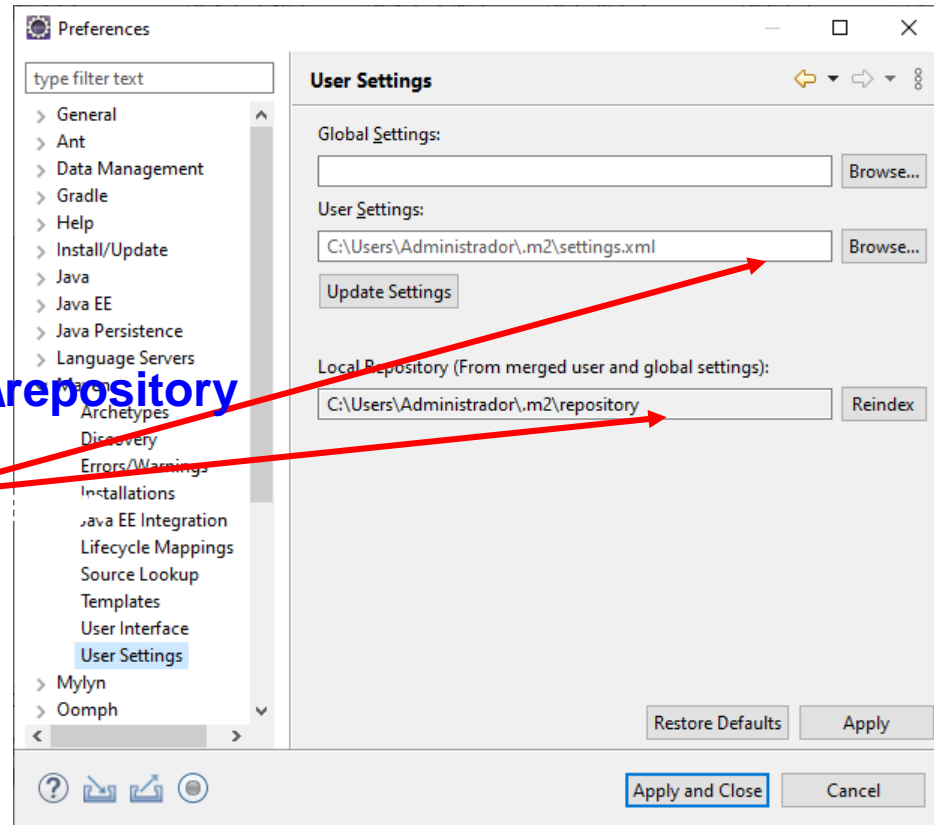
INSTALACION

Chequear en Eclipse, en el marketplace, accediendo desde el menu Ayuda:



Para cambiar el PATH del repositorio del MAVEN embebido al que instalamos ir a

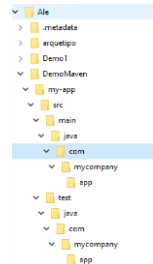
C:\Users\Administrador\.m2\repository



1er Proyecto Maven – por fuera de Eclipse

1. Ir a una carpeta donde quedara generado el proyecto
2. Ejecutar el siguiente comando en CMD

```
mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app -  
DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=1.4 -  
DinteractiveMode=false
```



Crea el proyecto
(en versiones anteriores era **CREATE**)

Estructura del proyecto

Otro ejemplo:

```
mvn archetype:generate -DarchetypeGroupId=org.apache.maven.archetypes -  
DarchetypeArtifactId=maven-archetype-webapp -DarchetypeVersion=1.4
```

https://maven.apache.org/guides/getting-started/index.html#How_do_I_make_my_first_Maven_project

```
Administrador: Símbolo del sistema
Microsoft Windows [Versión 10.0.19042.1415]
(c) Microsoft Corporation. Todos los derechos reservados.

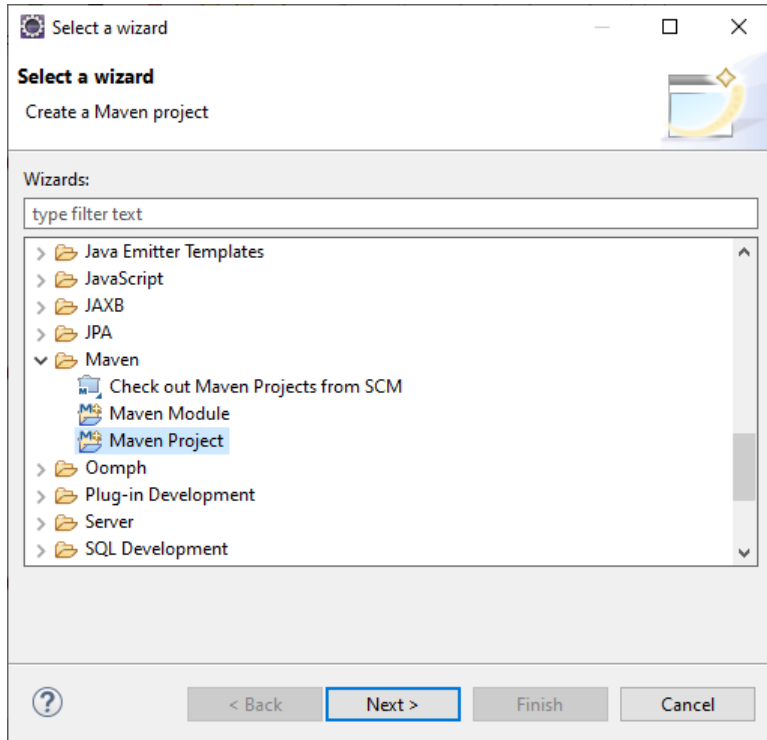
C:\Users\Administrador>cd ..

C:\Users>cd ..

C:\>mvn -B archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app -DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=1.4 -DinteractiveMode=false
[INFO] Scanning for projects...
[INFO] -----< org.apache.maven:standalone-pom >-----
[INFO] Building Maven Stub Project (No POM) 1
[INFO] -----[ pom ]-----
[INFO]
[INFO] >>> maven-archetype-plugin:3.2.0:generate (default-cli) > generate-sources @ standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:3.2.0:generate (default-cli) < generate-sources @ standalone-pom <<<
[INFO]
[INFO] --- maven-archetype-plugin:3.2.0:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Batch mode
[INFO] -----
[INFO] Using following parameters for creating project from Archetype: maven-archetype-quickstart:1.4
[INFO] -----
[INFO] Parameter: groupId, Value: com.mycompany.app
[INFO] Parameter: artifactId, Value: my-app
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: package, Value: com.mycompany.app
[INFO] Parameter: packageInPathFormat, Value: com/mycompany/app
[INFO] Parameter: package, Value: com.mycompany.app
[INFO] Parameter: groupId, Value: com.mycompany.app
[INFO] Parameter: artifactId, Value: my-app
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Project created from Archetype in dir: C:\my-app
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 12.045 s
[INFO] Finished at: 2021-12-23T09:01:48-03:00
[INFO] -----

C:\>
```


1er Proyecto Maven – Con Eclipse



New Maven Project

New Maven project

Select project name and location

☒ Create a simple project (skip archetype selection)

☒ Use default Workspace location

New Maven Project

New Maven project

⊗ Enter a group id for the artifact.

Artifact

Group Id:

Artifact Id:

Version: 0.0.1-SNAPSHOT

Packaging: jar

Name:

Description:

Parent Project

Group Id:

Artifact Id:

Version:

Advanced

New Maven Project

New Maven project

Select project name and location

☐ Create a simple project (skip archetype selection)

☒ Use default Workspace location

New Maven Project

New Maven project

Select an Archetype

Catalog: All Catalogs

Filter:

Group Id	Artifact Id	Version
----------	-------------	---------

☒ Show the last version of Archetype only ☐ Include snapshot archetypes

Advanced

POM . XML

PROJECT OBJECT MODEL:

Es un archivo XML que contiene información sobre el proyecto y los detalles de configuración utilizados por Maven para construir el proyecto.

POM mínimo

Los requisitos mínimos para un POM son los siguientes:

- `project` raíz
- `modelVersion` - debe establecerse en 4.0.0
- `groupId` - la identificación del grupo del proyecto.
- `artifactId` - la identificación del artefacto (proyecto)
- `version` - la versión del artefacto en el grupo especificado

He aquí un ejemplo:

```
1. <project>
2.   <modelVersion> 4.0.0 </modelVersion>
3.
4.   <groupId> com.miempresa.app </groupId>
5.   <artifactId> mi-aplicación </artifactId>
6.   <version> 1 </version>
7. </project>
```



Compilando un proyecto...



Bueno, ya tenemos el proyecto, ahora nos toca compilar. El archetype *maven-archetype-quickstart*, por defecto incluye un paquete y una clase, tanto en el código del proyecto como en el de test. Para el paquete utiliza la unión del groupId más artifactId que hemos especificado al crear el proyecto:

`src/main/java/com/empresa/escuelita/App.java:`

Compilando el proyecto...

```
/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Hello World!" );
    }
}
```

Compilando el proyecto...

```
public class AppTest extends TestCase
{
    /**
     * Create the test case
     *
     * param testName name of the test case
     */
    public AppTest( String testName )
    {
        super( testName );
    }
    /**
     * return the suite of tests being tested
     */
    public static Test suite()
    {
        return new TestSuite( AppTest.class );
    }
    /**
     * Rigorous Test :-)
     */
    public void testApp()
    {
        assertTrue( true );
    }
}
```

Compilando el proyecto

Como podemos ver, el código del ejemplo es bastante trivial.

Ahora vamos a compilar:

```
mvn package
```

Este goal hace dos cosas, compila y empaqueta. En este caso, crea un jar. Si solo quisiéramos compilar usaríamos *compile* en vez de *package*.

Si ahora navegamos dentro del proyecto, verán que tenemos una nueva carpeta llamada `target`. En ella es donde maven genera los `.class` y en este caso, el `jar`.

Instalando el Artefacto



Instalando el artefacto

Por último nos queda instalar nuestro artefacto en el repositorio local de maven:

```
mvn install
```

Veamos la salida:

```
[INFO] --- maven-install-plugin:2.3.1:install (default-install) @ maven-jar-sample [INFO] Installing /Users/afichera/workspaces/excuelita/maven-jar-sample/target/maven-jar-sample-1.0-SNAPSHOT.jar to /Users/afichera/.m2/repository/com/empresa/escuelita/maven-jar-sample/1.0-SNAPSHOT/maven-jar-sample-1.0-SNAPSHOT.jar [INFO] Installing /Users/afichera/workspaces/escuelita/maven-jar-sample/pom.xml to /Users/afichera/.m2/repository/com/empresa/escuelita/maven-jar-sample/1.0-SNAPSHOT/maven-jar-sample-1.0-SNAPSHOT.pom
```

Este goal lo que hace es copiar tanto el jar como el pom a nuestro repositorio.

La ruta coincide con el grupo que especificamos al crear el proyecto, concatenando el nombre del artefacto y la versión.

¿De que nos sirve esto?

La gran utilidad es que ahora podemos incluir este proyecto como dependencia en otro proyecto maven, mejorando la potabilidad y modularidad de nuestros proyectos

Artifactory... ¿que es?

Artifactory, ¿Que es?

Artifactory es una herramienta que nos permite crear y gestionar repositorios maven en la red.

Muy útil para cualquier equipo de desarrollo, ya que distintos programadores pueden acceder a todos los artefactos creados en una empresa de desarrollo e incluir lo que necesiten en sus proyectos.

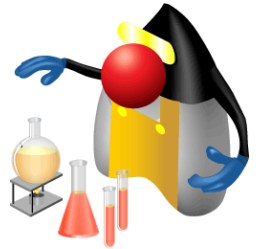
ARTEFACTOS



Que son? **groupid → es el nombre del dominio de la empresa**

- A. Un artefacto es un archivo, generalmente un JAR, que se implementa en un repositorio de Maven.
- B. Una construcción de Maven produce uno o más artefactos, como un JAR compilado y un JAR de "fuentes".
- C. Cada artefacto tiene un ID de grupo (generalmente un nombre de dominio invertido, como com.example.foo), un ID de artefacto (solo un nombre) y una cadena de versión. Los tres juntos identifican de forma única el artefacto.
- D. Las dependencias de un proyecto se especifican como artefactos.
- E. En términos generales de software, un " [artifact](#) " es algo producido por el proceso de desarrollo de software, ya sea una documentación relacionada con el software o un archivo ejecutable.
- F. En la terminología de Maven, el artefacto es el resultado resultante de la compilación de Maven, generalmente un jar o war u otro archivo ejecutable. Los artefactos en maven se identifican mediante un sistema de coordenadas de groupid, artifactid y version. Maven usa groupid, artifactid y version para identificar las dependencias (generalmente otros archivos jar) necesarias para construir y ejecutar su código.

artifactid → es el nombre del proyecto en eclipse



Un repositorio en Maven contiene artefactos de compilación y dependencias de diversos tipos.

1. LOCAL

el repositorio local es un directorio en la computadora donde se ejecuta Maven. Almacena en caché las descargas remotas y contiene artefactos de compilación temporales que aún no ha publicado

2. REMOTO

refieren a cualquier otro tipo de repositorio, al que se accede mediante una variedad de protocolos como file://y https://.

```
1. <project>
2.   ...
3.   <repositories>
4.     <repository>
5.       <id>my-internal-site</id>
6.       <url>https://myserver/repo</url>
7.     </repository>
8.   </repositories>
9.   ...
10. </project>
```

PLUGINS

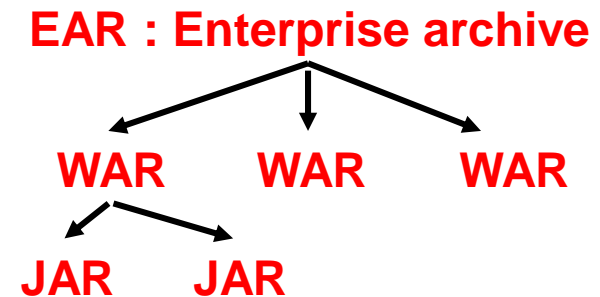
Maven es en realidad un marco de ejecución de complementos donde cada tarea se realiza realmente mediante complementos. Los plugins de Maven se utilizan generalmente para

1. crear archivo jar (java archive)
2. crear archivo war (web archive)
3. compilar archivos de código
4. pruebas unitarias de código
5. crear documentación del proyecto (javadoc)
6. crear informes de proyecto (testNG, genera reportes de testeo)

JAR: java archiveCon el comando `jar nombreProyecto`

WAR: web archive

EAR: Enterprise archive



PLUGINS

Un plugin generalmente proporciona un conjunto de objetivos, que se pueden ejecutar utilizando la siguiente sintaxis: crear archivo jar

```
mvn [plugin-name]:[goal-name]
```

Por ejemplo, un proyecto Java se puede compilar con el objetivo de compilación del maven-compiler-plugin ejecutando el siguiente comando

```
mvn compiler:compile
```



Integración con el IDE



Los proyectos maven, se pueden importar en los IDE mas conocidos del mercado. En eclipse para hacerlo hay que ejecutar el siguiente comando dentro de la carpeta del proyecto

mvn eclipse:eclipse

Este Goal crea los ficheros necesarios para que Eclipse pueda importar el proyecto. Lo único que tenemos que hacer es File -> import -> Existing projects into workspace y seleccionar la carpeta correspondiente a nuestro proyecto. Si tenemos instalado el plugin m2eclipse, no hace falta ejecutar ningún Goal de maven. Simplemente abrimos File -> import -> Existing maven projects y ya lo tenemos en nuestro IDE.