

---

## Práctica 2. Maven

### Objetivos

- Aprender a **gestionar un proyecto con Maven**
- Comprender las distintas etapas del ciclo de vida de un proyecto
- Gestionar de manera eficiente las dependencias de los proyectos
- Utilizar plugins que hagan aún más productivo el uso de Maven

En este boletín de prácticas se pretende que el alumno aprende a gestionar los proyectos de una manera estructurada y estandarizada. También aprenderá a manejar dependencias a otros proyectos mediante el uso de repositorios de artefactos.

### Introducción

En nuestros proyectos Java siempre tenemos varias tareas que realizar. La primera suele ser crear una estructura de directorios para nuestro proyecto, la cual incluye carpetas para artefactos tales como: ficheros fuentes (.java), iconos, datos, ficheros de configuración, ficheros compilados (.class), ficheros .jar generados, la documentación javadoc, etc. Otras tareas que realizamos normalmente son compilar los ficheros .java, borrar ficheros .class, generar la documentación en javadoc, empaquetar el proyecto en un jar o incluso podemos generar documentación web para publicar nuestro trabajo. Si nuestro programa es grande, incluso es posible que dependamos de otros ficheros jar externos, como los drivers de base de datos, la librería de JUnit para las clases de test, la librería Log4j para la salida de un log, etc. En este caso, tendremos que copiar todas estas dependencias (librerías .jar externas) en algún sitio de nuestro proyecto e incluirlas.

Es importante automatizar estas tareas repetitivas de manera que supongan el menor esfuerzo para el desarrollador. Para ello, podríamos pensar en crear algunos scripts o ficheros .bat, aunque sería más eficiente utilizar alguna de las herramientas disponibles para definir y ejecutar flujos de tareas como es Ant o incluso mejor utilizar una herramienta como Maven para gestión de proyectos software. En el caso de Ant, no hay tareas predefinidas y el desarrollador debe definir cada tarea por lo que se reescribe el fichero build.xml (define las tareas que se ejecutan con Ant) de un proyecto a otro. También tendremos que copiar al nuevo proyecto los .jar externos de las dependencias. En cambio, Maven es una herramienta para la gestión de proyectos que permite realizar las tareas asociadas a la construcción de un artefacto software de forma rápida y completa. A través de órdenes simples, Maven nos crea una estructura de directorios para nuestro proyecto con los directorios para los archivos fuentes, los iconos, ficheros de configuración y datos, etc. Si a Maven le indicamos qué jar externos necesitamos, es capaz de ir a buscarlos a internet y descargarlos por nosotros. Sin necesidad prácticamente de configurar nada, Maven sabe como borrar los .class, compilar, generar el .jar, generar el javadoc y generar una documentación web con bastantes informes (métricas, código duplicado, etc). Maven se encarga de ejecutar automáticamente nuestros tests de prueba cuando compilamos. Incluso nos genera un zip de distribución en el que van todos los .jar necesarios y ficheros de configuración de nuestro proyecto.

## Parte 1. Instalar Maven

Descargar (<http://maven.apache.org/download.html>) y descomprimir **Maven** (la distribución viene como un fichero zip). Para poder usar Maven desde la línea de comandos será necesario configurar una serie de variables de entorno. A continuación se detallan las mismas:

M2\_HOME : Indica la carpeta con la distribución de Maven  
M2 : Referencia a la carpeta /bin de la distribución de Maven  
M2\_REPO : Indica la carpeta con el repositorio local de Maven

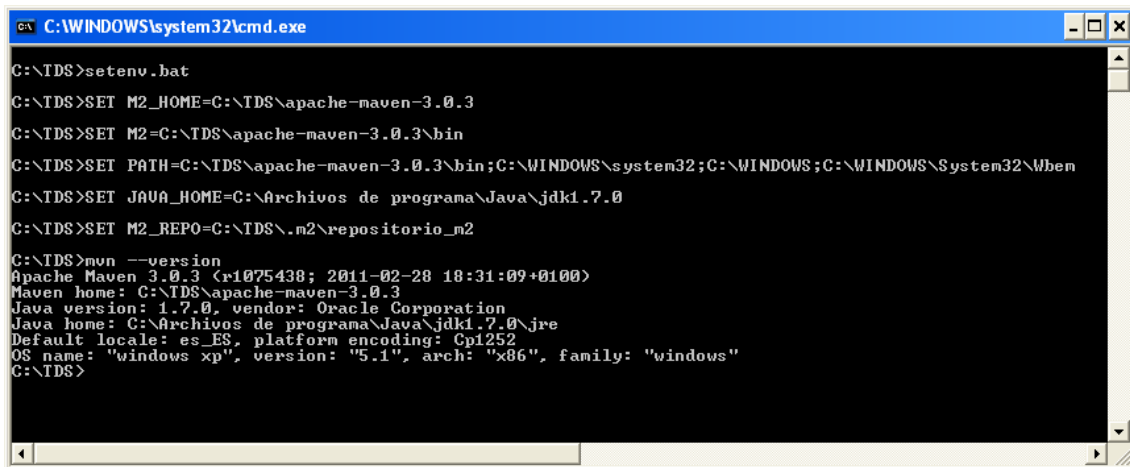
Además de estas variables, debemos configurar correctamente la variable JAVA\_HOME y añadir M2 a la variable PATH de Windows.

Un ejemplo de configuración de las variables sería:

```
SET M2_HOME=C:\TDS\apache-maven-3.0.3
SET M2=C:\TDS\apache-maven-3.0.3\bin
SET PATH=%M2%;%PATH%
SET JAVA_HOME=C:\Archivos de programa\Java\jdk1.7.0
SET M2_REPO=C:\TDS\.m2\repositorio_m2
```

Probaremos ahora que la instalación de Maven se realizó correctamente:

```
mvn --version
```



```
C:\WINDOWS\system32\cmd.exe
C:\TDS>setenv.bat
C:\TDS>SET M2_HOME=C:\TDS\apache-maven-3.0.3
C:\TDS>SET M2=C:\TDS\apache-maven-3.0.3\bin
C:\TDS>SET PATH=C:\TDS\apache-maven-3.0.3\bin;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem
C:\TDS>SET JAVA_HOME=C:\Archivos de programa\Java\jdk1.7.0
C:\TDS>SET M2_REPO=C:\TDS\.m2\repositorio_m2
C:\TDS>mvn --version
Apache Maven 3.0.3 (r1075438; 2011-02-28 18:31:09+0100)
Maven home: C:\TDS\apache-maven-3.0.3
Java version: 1.7.0, vendor: Oracle Corporation
Java home: C:\Archivos de programa\Java\jdk1.7.0\jre
Default locale: es_ES, platform encoding: Cp1252
OS name: "windows xp", version: "5.1", arch: "x86", family: "windows"
C:\TDS>
```

El comando `--version` debe mostrar la información acerca de la instalación de Maven.

Por defecto, la distribución de Maven asume que el repositorio local en Windows se encuentra en la carpeta `\.m2\repository` que hay dentro de la carpeta de nuestro usuario Windows en 'Documents and settings'. Como se puede ver en el anterior ejemplo, el repositorio local referencia a una carpeta personal (crear dicha carpeta). Para hacer esto, modificamos el fichero de configuración de Maven **settings.xml** que encontraremos en la carpeta `/conf` de Maven. Veremos que justo después de la etiqueta raíz `<settings>` del documento xml, aparece comentada la etiqueta para definir el repositorio local. Indicaremos, por ejemplo:

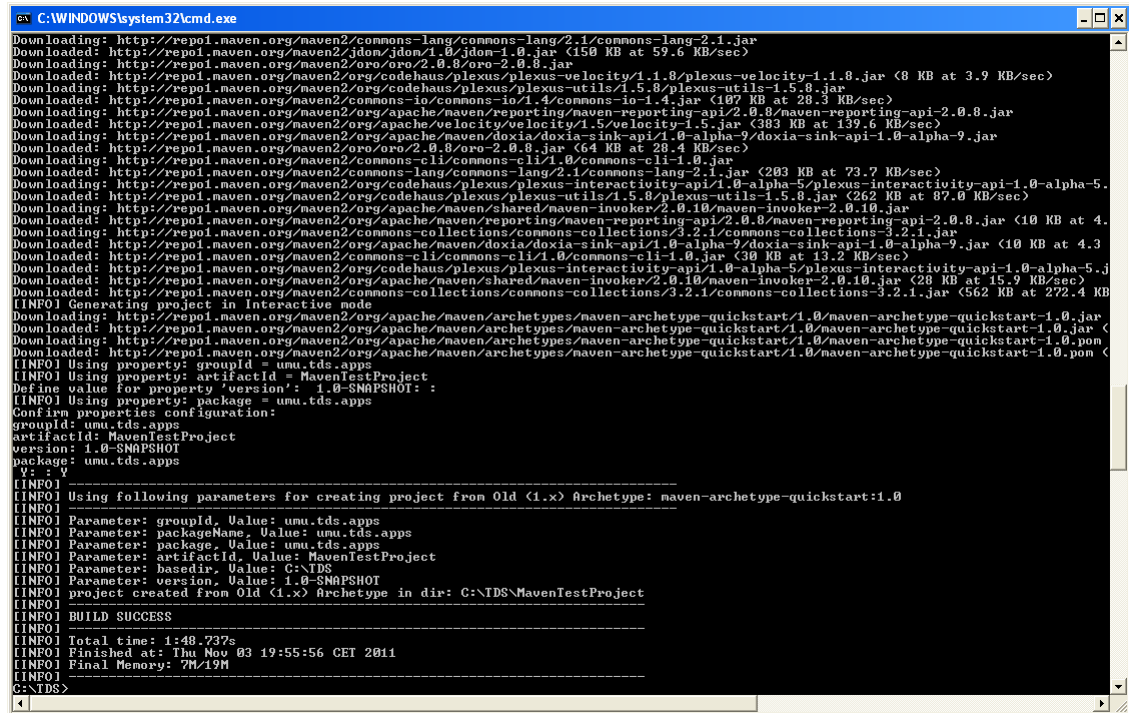
```
<localRepository>C:\TDS\.m2\repositorio_m2</localRepository>
```

Ya tenemos personalizado nuestro repositorio local, aunque actualmente se encuentra vacío.

## Parte 2. Crear un proyecto

Vamos a crear nuestro primer proyecto con Maven. Debemos recordar que Maven requiere una conexión a Internet para descargar las dependencias y los artefactos que requiere para crear cada tipo de proyecto (las descargas se instalarán en el repositorio local, de donde se tomarán en futuros proyectos)

```
mvn archetype:generate -DgroupId=unu.tds.apps -DartifactId=MavenTestProject -DarchetypeArtifactId=maven-archetype-quickstart
```



```
C:\WINDOWS\system32\cmd.exe
Downloading: http://repo1.maven.org/maven2/commons-lang/commons-lang/2.1/commons-lang-2.1.jar
Downloaded: http://repo1.maven.org/maven2/jdom/jdom/1.0/jdom-1.0.jar (150 KB at 59.6 KB/sec)
Downloading: http://repo1.maven.org/maven2/oro/oro/2.0.8/oro-2.0.8.jar
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-velocity/1.1.8/plexus-velocity-1.1.8.jar (8 KB at 3.9 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-utils/1.5.0/plexus-utils-1.5.0.jar
Downloaded: http://repo1.maven.org/maven2/commons-io/commons-io/1.4/commons-io-1.4.jar (187 KB at 28.3 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/reporting/maven-reporting-api/2.0.8/maven-reporting-api-2.0.8.jar
Downloaded: http://repo1.maven.org/maven2/org/apache/velocity/velocity/1.5/velocity-1.5.jar (383 KB at 139.6 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/doxia/doxia-sink-api/1.0-alpha-9/doxia-sink-api-1.0-alpha-9.jar
Downloaded: http://repo1.maven.org/maven2/oro/oro/2.0.8/oro-2.0.8.jar (64 KB at 28.4 KB/sec)
Downloaded: http://repo1.maven.org/maven2/commons-cli/commons-cli/1.0/commons-cli-1.0.jar
Downloaded: http://repo1.maven.org/maven2/commons-lang/commons-lang/2.1/commons-lang-2.1.jar (283 KB at 73.7 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-interactivity-api/1.0-alpha-5/plexus-interactivity-api-1.0-alpha-5.jar
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-utils/1.5.8/plexus-utils-1.5.8.jar (262 KB at 87.0 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/shared/maven-invoker/2.0.10/maven-invoker-2.0.10.jar
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/reporting/maven-reporting-api/2.0.8/maven-reporting-api-2.0.8.jar (10 KB at 4.
Downloaded: http://repo1.maven.org/maven2/commons-collections/commons-collections/3.2.1/commons-collections-3.2.1.jar (10 KB at 4.3
Downloaded: http://repo1.maven.org/maven2/commons-cli/commons-cli/1.0/commons-cli-1.0.jar (30 KB at 13.2 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-interactivity-api/1.0-alpha-5/plexus-interactivity-api-1.0-alpha-5.j
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/shared/maven-invoker/2.0.10/maven-invoker-2.0.10.jar (28 KB at 15.9 KB/sec)
Downloaded: http://repo1.maven.org/maven2/commons-collections/commons-collections/3.2.1/commons-collections-3.2.1.jar (562 KB at 272.4 KB
[INFO] Generating project in Interactive mode
Downloading: http://repo1.maven.org/maven2/org/apache/maven/archetypes/maven-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.jar
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/archetypes/maven-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.jar (
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/archetypes/maven-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.pom (
[INFO] Using property: groupId = unu.tds.apps
[INFO] Using property: artifactId = MavenTestProject
Define value for property 'version': 1.0-SNAPSHOT:
[INFO] Using property: package = unu.tds.apps
Confirm properties configuration:
groupId: unu.tds.apps
artifactId: MavenTestProject
version: 1.0-SNAPSHOT
package: unu.tds.apps
Y: Y
[INFO]
[INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-archetype-quickstart:1.0
[INFO]
[INFO] Parameter: groupId, Value: unu.tds.apps
[INFO] Parameter: packageName, Value: unu.tds.apps
[INFO] Parameter: package, Value: unu.tds.apps
[INFO] Parameter: artifactId, Value: MavenTestProject
[INFO] Parameter: basedir, Value: C:\TDS
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: C:\TDS\MavenTestProject
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 1:48.737s
[INFO] Finished at: Thu Nov 03 19:55:56 CET 2011
[INFO] Final Memory: 7M/19M
[INFO]
C:\TDS>
```

Se nos preguntará por la versión de nuestro proyecto (en fases iniciales, dejaremos SNAPSHOT 1.0) y que confirmemos la configuración.

Nota: Si el tipo de proyecto (*archetype*) usado tiene mas de una versión, Maven puede solicitarnos que la seleccionemos (ver la captura anterior para el arquetipo *quickstart*)

El comando/plugin *generate* recibe como parámetros los identificadores del grupo de aplicaciones (-DgroupId), el identificador de la propia aplicación (-DartifactId) y el tipo de proyecto a crear (-DarchetypeArtifactId).

Ya tenemos creado nuestro proyecto simple usando *generate*. Comprobaremos que se ha creado el directorio del proyecto. Se ha usado un tipo de proyecto (arquetipo) simple (*quickstart*) por lo que la estructura generada en la carpeta del proyecto (\MavenTestProject) para el mismo es bien sencilla.

El tipo de proyecto 'quickstart' tan solo crea 2 directorios dentro de la carpeta de código fuente (**src**): uno para la aplicación (**main**) y otro para las pruebas (**test**). Crea también un fichero **pom.xml** que es un fichero que contiene datos de configuración de nuestro proyecto, como dependencias con otros .jar, tipos de informes que queremos en la página web de nuestro proyecto, etc. El contenido generado para este fichero es el siguiente:

```

<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/maven-v4_0_0.xsd">

    <modelVersion>4.0.0</modelVersion>
    <groupId>umu.tds.apps</groupId>
    <artifactId>MavenTestProject</artifactId>
    <packaging>jar</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>MavenTestProject</name>
    <url>http://maven.apache.org</url>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

</project>

```

Podemos ver los datos de configuración principales que indicamos al generar el proyecto. Cabe resaltar el apartado de dependencias donde se liga nuestro proyecto con una serie de artefactos (librerías) de las que depende: en nuestro tipo de proyecto, se usa JUnit en el ámbito de las pruebas (test). Por lo tanto aquí es donde se pueden añadir las dependencias que requiera nuestro proyecto (como se verá más adelante).

Como hemos comentado antes, se han creado los subdirectorios **main** y **test** en **src**. Dentro de **main** debemos colocar todos los ficheros fuentes y ficheros de configuración o datos propios del proyecto. En **test** debemos colocar todos nuestros ficheros fuentes de prueba, ficheros de datos o de configuración de pruebas, etc. Es decir, en **main** tendremos los artefactos que constituyen la aplicación y en **test** lo que nos permite probarla. En ambos casos, se ha creado un directorio **java** donde se almacenan los fuentes. En paralelo a estos directorios **java** y si lo necesitamos, debemos crear nosotros manualmente otros directorios. El nombre de estos nuevos directorios es estándar en **Maven**:

- **config** para ficheros de configuración, iconos, etc
- **resources** para ficheros que queramos que formen parte del jar. **Maven** incluirá automáticamente todo lo que haya en este subdirectorio dentro del jar
- **assembly** para la configuración que queramos en nuestro zip de distribución.

Se han generado automáticamente en el directorio **java de main** (se crean carpetas **++++**) un archivo App.java (que simplemente muestra un "HelloWorld") y en el directorio **java de test** un fichero AppTest.java con una prueba JUnit para App.java.

Nota: En <http://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html> se puede encontrar la estructura de directorios propuesta desde Maven.

## Parte 3. Compilar un proyecto

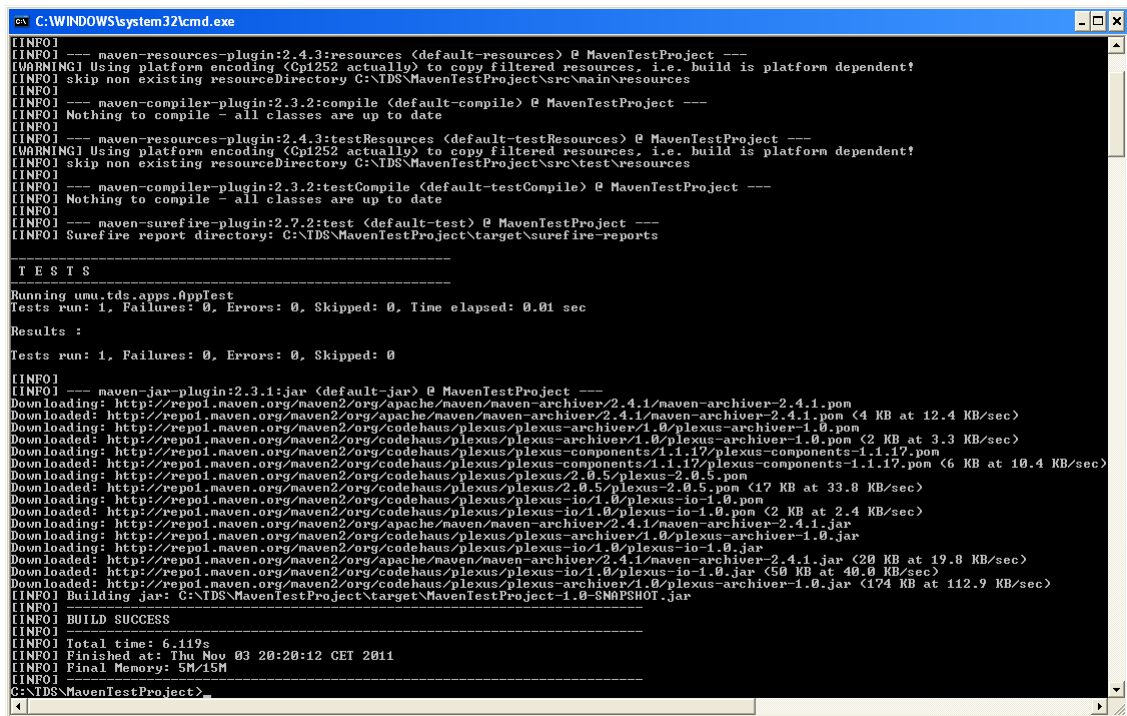
```
mvn compile
```

```
mvn test
```

## Parte 4. Empaquetar un proyecto

Para empaquetar la aplicación basta con ejecutar:

`mvn package`



```
C:\WINDOWS\system32\cmd.exe
[INFO] --- maven-resources-plugin:2.4.3:resources (default-resources) @ MavenTestProject ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\TDS\MavenTestProject\src\main\resources
[INFO] --- maven-compiler-plugin:2.3.2:compile (default-compile) @ MavenTestProject ---
[INFO] Nothing to compile - all classes are up to date
[INFO] --- maven-resources-plugin:2.4.3:testResources (default-testResources) @ MavenTestProject ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\TDS\MavenTestProject\src\test\resources
[INFO] --- maven-compiler-plugin:2.3.2:testCompile (default-testCompile) @ MavenTestProject ---
[INFO] Nothing to compile - all classes are up to date
[INFO] --- maven-surefire-plugin:2.7.2:test (default-test) @ MavenTestProject ---
[INFO] Surefire report directory: C:\TDS\MavenTestProject\target\surefire-reports

T E S T S
Running umu.tds.apps.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.01 sec
Results :
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] --- maven-jar-plugin:2.3.1:jar (default-jar) @ MavenTestProject ---
Downloading: http://repo1.maven.org/maven2/org/apache/maven/maven-archiver/2.4.1/maven-archiver-2.4.1.pom (4 KB at 12.4 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/maven-archiver/2.4.1/maven-archiver-2.4.1.pom (4 KB at 12.4 KB/sec)
Downloading: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-archiver/1.0/plexus-archiver-1.0.pom (2 KB at 3.3 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-archiver/1.0/plexus-archiver-1.0.pom (2 KB at 3.3 KB/sec)
Downloading: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-components/1.1.17/plexus-components-1.1.17.pom (6 KB at 10.4 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-components/1.1.17/plexus-components-1.1.17.pom (6 KB at 10.4 KB/sec)
Downloading: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus/2.0.5/plexus-2.0.5.pom (17 KB at 33.8 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus/2.0.5/plexus-2.0.5.pom (17 KB at 33.8 KB/sec)
Downloading: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-io/1.0/plexus-io-1.0.pom (2 KB at 2.4 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-io/1.0/plexus-io-1.0.pom (2 KB at 2.4 KB/sec)
Downloading: http://repo1.maven.org/maven2/org/apache/maven/maven-archiver/2.4.1/maven-archiver-2.4.1.jar (20 KB at 19.8 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/maven-archiver/2.4.1/maven-archiver-2.4.1.jar (20 KB at 19.8 KB/sec)
Downloading: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-archiver/1.0/plexus-archiver-1.0.jar (50 KB at 40.0 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-archiver/1.0/plexus-archiver-1.0.jar (50 KB at 40.0 KB/sec)
[INFO] Building jar: C:\TDS\MavenTestProject\target\MavenTestProject-1.0-SNAPSHOT.jar
[INFO] BUILD SUCCESS
[INFO] Total time: 6.119s
[INFO] Finished at: Thu Nov 03 20:20:12 CET 2011
[INFO] Final Memory: 5M/15M
[INFO] C:\TDS\MavenTestProject>
```

Esto primero compilará si es necesario, pasará las clases de test de JUnit y si no hay fallos, incluirá en el directorio **target** nuestro fichero .jar con la aplicación empaquetada (solo la aplicación, sin el contenido de test), que por defecto será:

`MavenTestProject-1.0-SNAPSHOT.jar`

**MavenTestProject** es el nombre dado a nuestra aplicación según indicamos en el **artifactId**. **Maven** añade un 1.0 para indicar que es la versión 1.0 de nuestro proyecto. Este número aparece en el fichero **pom.xml** (se puede modificar). **-SNAPSHOT** indica que esta versión está en construcción, que no es definitiva. **Maven** irá guardando todas las versiones del .jar que generemos e irá sustituyendo **-SNAPSHOT** por la fecha y hora de la construcción. La etiqueta **-SNAPSHOT** también aparece en el **pom.xml** y podemos quitarla cuando creamos que tenemos la versión definitiva.

Dentro del .jar se incluirán, también automáticamente, todos los ficheros que tengamos debajo del directorio **resources**

Si en cualquier momento queremos eliminar los archivos generados por Maven en un proyecto y ‘limpiar’ el mismo, podemos escribir:

`mvn clean`



```
C:\WINDOWS\system32\cmd.exe

C:\TDS\MavenTestProject>mvn clean
[INFO] Scanning for projects...
[INFO]
[INFO] Building MavenTestProject 1.0-SNAPSHOT
[INFO]
[INFO] --- maven-clean-plugin:2.4.1:clean (default-clean) @ MavenTestProject ---
[INFO] Deleting C:\TDS\MavenTestProject\target
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 0.341s
[INFO] Finished at: Thu Nov 03 20:31:17 CET 2011
[INFO] Final Memory: 3M/15M
[INFO]
C:\TDS\MavenTestProject>
```

## Parte 5. Instalar un proyecto en el repositorio

Para instalar el proyecto en nuestro repositorio basta con ejecutar:

```
mvn install
```

```
C:\WINDOWS\system32\cmd.exe

[INFO] Building MavenTestProject 1.0-SNAPSHOT
[INFO]
[INFO] --- maven-resources-plugin:2.4.3:resources (default-resources) @ MavenTestProject ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\TDS\MavenTestProject\src\main\resources
[INFO]
[INFO] --- maven-compiler-plugin:2.3.2:compile (default-compile) @ MavenTestProject ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.4.3:testResources (default-testResources) @ MavenTestProject ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\TDS\MavenTestProject\src\test\resources
[INFO]
[INFO] --- maven-compiler-plugin:2.3.2:testCompile (default-testCompile) @ MavenTestProject ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.7.2:test (default-test) @ MavenTestProject ---
[INFO] Surefire report directory: C:\TDS\MavenTestProject\target\surefire-reports

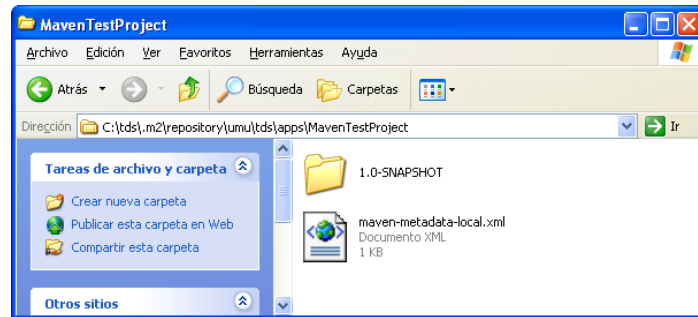
T E S T S
Running umu.tds.apps.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.02 sec

Results :
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] --- maven-jar-plugin:2.3.1:jar (default-jar) @ MavenTestProject ---
[INFO]
[INFO] --- maven-install-plugin:2.3.1:install (default-install) @ MavenTestProject ---
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-digest/1.0/plexus-digest-1.0.pom (2 KB at 2.4 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-components/1.1.7/plexus-components-1.1.7.pom (5 KB at 9.9 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-components/1.1.7/plexus-components-1.1.7.pom (5 KB at 9.9 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus/1.0.8/plexus-1.0.8.pom (8 KB at 14.4 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-container-default/1.0-alpha-8/plexus-container-default-1.0-alpha-8.pom (12 KB at 23.6 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-digest/1.0/plexus-digest-1.0.jar (12 KB at 23.6 KB/sec)
[INFO] Installing C:\TDS\MavenTestProject\target\MavenTestProject-1.0-SNAPSHOT.jar to C:\TDS\.m2\repository\umu\tds\apps\MavenTestProject\1.0-SNAPSHOT\MavenTestProject-1.0-SNAPSHOT.jar
[INFO] Installing C:\TDS\MavenTestProject\pom.xml to C:\TDS\.m2\repository\umu\tds\apps\MavenTestProject\1.0-SNAPSHOT\MavenTestProject-1.0-SNAPSHOT.pom
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 3.966s
[INFO] Finished at: Thu Nov 03 20:25:40 CET 2011
[INFO] Final Memory: 4M/15M
[INFO]
C:\TDS\MavenTestProject>
```

Esto compilará si son necesarios nuestros archivos fuentes, les pasará los test, generará el jar y lo copiará en nuestro repositorio local. Esta operación hace que ese jar esté disponible para otros proyectos **maven** que tengamos en nuestro ordenador. Es útil, por tanto, para proyectos **maven** que sean librerías que queramos usar en varios proyectos.

Podemos comprobar ahora que en nuestro repositorio local se encuentra la aplicación del proyecto, dentro del paquete en el que la definimos



Si hemos configurado un repositorio en red (no local) común a varios programadores en varios ordenadores, el comando a usar es:

```
mvn deploy
```

Esto hace todo lo que hace **install** (incluido el propio install) y luego pone nuestro jar en ese repositorio común en red, lo que permite que esté a disposición de todos los programadores del equipo.

El **-SNAPSHOT** en la versión tiene aquí su importancia. Si dependemos de un jar que tenga **-SNAPSHOT**, cada vez que compilemos, aunque ese jar esté en nuestro repositorio local, **maven** ira a buscarlos a los repositorios comunes o de internet, para ver si hay una versión de fecha más moderna. Si la hay, se la bajará. Por tanto, suele ser útil en un equipo de trabajo mantener la etiqueta de la versión en **-SNAPSHOT** en los jar que todavía están en desarrollo y sufren cambios frecuentes.

Si no ponemos **-SNAPSHOT**, una vez bajado el jar a nuestro repositorio local, **maven** no se preocupará de buscar si hay versiones más modernas en los repositorios remotos.

## Parte 6. Generar documentación

Para generar la documentación debemos editar el fichero **pom.xml** para indicarle que queremos este tipo de documentación (a continuación de **</dependencies>**):

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



Luego ejecutamos el plugin (de nuevo deberá descargarlo):

```
mvn javadoc:javadoc
```

```
C:\WINDOWS\system32\cmd.exe
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-velocity/1.1.7/plexus-velocity-1.1.7.jar (8 KB at 3.0 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/doxia/doxia-module-xdoc/1.0/doxia-module-xdoc-1.0.jar
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/doxia/doxia-decoration-model/1.0/doxia-decoration-model-1.0.jar (48 KB at 19.2 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/doxia/doxia-module-xhtml/1.0/doxia-module-xhtml-1.0.jar
Downloaded: http://repo1.maven.org/maven2/commons-collections/commons-collections/3.2/commons-collections-3.2.jar (558 KB at 184.5 KB/sec)
Downloaded: http://repo1.maven.org/maven2/commons-lang/commons-lang/2.4/commons-lang-2.4.jar
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/doxia/doxia-module-apt/1.0/doxia-module-apt-1.0.jar (46 KB at 20.5 KB/sec)
Downloaded: http://repo1.maven.org/maven2/commons-httpclient/commons-httpclient/3.1/commons-httpclient-3.1.jar
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/doxia/doxia-module-fnl/1.0/doxia-module-fnl-1.0.jar (19 KB at 8.0 KB/sec)
Downloaded: http://repo1.maven.org/maven2/commons-codec/commons-codec/1.2/commons-codec-1.2.jar
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/doxia/doxia-module-xdoc/1.0/doxia-module-xdoc-1.0.jar (28 KB at 10.9 KB/sec)
Downloaded: http://repo1.maven.org/maven2/commons-logging/commons-logging/1.1/commons-logging-1.1.jar
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/doxia/doxia-module-xhtml/1.0/doxia-module-xhtml-1.0.jar (22 KB at 8.5 KB/sec)
Downloaded: http://repo1.maven.org/maven2/log4j/log4j/1.2.14/log4j-1.2.14.jar
Downloaded: http://repo1.maven.org/maven2/commons-lang/commons-lang/2.4/commons-lang-2.4.jar (256 KB at 86.0 KB/sec)
Downloaded: http://repo1.maven.org/maven2/com/thoughtworks/qdox/qdox/1.9.2/qdox-1.9.2.jar
Downloaded: http://repo1.maven.org/maven2/commons-codec/commons-codec/1.2/commons-codec-1.2.jar (30 KB at 12.9 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-archiver/1.0-alpha-9/plexus-archiver-1.0-alpha-9.jar
Downloaded: http://repo1.maven.org/maven2/commons-httpclient/commons-httpclient/3.1/commons-httpclient-3.1.jar (298 KB at 108.2 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-io/1.0-alpha-1/plexus-io-1.0-alpha-1.jar
Downloaded: http://repo1.maven.org/maven2/commons-logging/commons-logging/1.1/commons-logging-1.1.jar (60 KB at 26.3 KB/sec)
Downloaded: http://repo1.maven.org/maven2/log4j/log4j/1.2.14/log4j-1.2.14.jar (359 KB at 130.8 KB/sec)
Downloaded: http://repo1.maven.org/maven2/com/thoughtworks/qdox/qdox/1.9.2/qdox-1.9.2.jar (167 KB at 83.3 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-io/1.0-alpha-1/plexus-io-1.0-alpha-1.jar (42 KB at 6.9 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-archiver/1.0-alpha-9/plexus-archiver-1.0-alpha-9.jar (154 KB at 69.3 KB/sec)
[WARNING] Source files encoding has not been set, using platform encoding Cp1252, i.e. build is platform dependent!
[INFO]
[INFO] Loading source files for package umu.tds.apps...
[INFO] Constructing Javadoc information...
[INFO] Standard Doclet version 1.7.0
[INFO] Building tree for all the packages and classes...
[INFO] Generating C:\TDS\MavenTestProject\target\site\apidocs\umu.tds.apps\App.html...
[INFO] Generating C:\TDS\MavenTestProject\target\site\apidocs\umu.tds.apps\package-frame.html...
[INFO] Generating C:\TDS\MavenTestProject\target\site\apidocs\umu.tds.apps\package-summary.html...
[INFO] Generating C:\TDS\MavenTestProject\target\site\apidocs\umu.tds.apps\package-tree.html...
[INFO] Generating C:\TDS\MavenTestProject\target\site\apidocs\umu.tds.apps\constant-values.html...
[INFO] Generating C:\TDS\MavenTestProject\target\site\apidocs\umu.tds.apps\class-use/App.html...
[INFO] Generating C:\TDS\MavenTestProject\target\site\apidocs\umu.tds.apps\package-use.html...
[INFO] Building index for all the packages and classes...
[INFO] Generating C:\TDS\MavenTestProject\target\site\apidocs\overview-tree.html...
[INFO] Generating C:\TDS\MavenTestProject\target\site\apidocs\index-all.html...
[INFO] Generating C:\TDS\MavenTestProject\target\site\apidocs\deprecated-list.html...
[INFO] Building index for all classes...
[INFO] Generating C:\TDS\MavenTestProject\target\site\apidocs\allclasses-frame.html...
[INFO] Generating C:\TDS\MavenTestProject\target\site\apidocs\allclasses-noframe.html...
[INFO] Generating C:\TDS\MavenTestProject\target\site\apidocs\index.html...
[INFO] Generating C:\TDS\MavenTestProject\target\site\apidocs\help-doc.html...
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 49.854s
[INFO] Finished at: Thu Nov 03 20:39:49 CET 2011
[INFO] Final Memory: 8M/21M
[INFO]
C:\TDS\MavenTestProject>
```

...y **maven** nos generará en target un directorio **target\site\apidocs** y dentro de él metera el javadoc.

Si ejecutamos

```
mvn site:site
```

```
C:\WINDOWS\system32\cmd.exe
[WARNING]
[INFO]
[INFO] Building MavenTestProject 1.0-SNAPSHOT
[INFO]
[INFO] --- maven-site-plugin:2.0.1:site (default-site) @ MavenTestProject ---
[INFO]
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/shared/maven-doxia-tools/1.0.2/maven-doxia-tools-1.0.2.pom (6 KB at 12.7 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/shared/maven-shared-components/11/maven-shared-components-11.pom
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/shared/maven-shared-components/11/maven-shared-components-11.pom (9 KB at 16.9 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/wagon/wagon-provider-api/1.0-beta-4/wagon-provider-api-1.0-beta-4.pom (906 B at 1.0 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/wagon/wagon/1.0-beta-4/wagon-1.0-beta-4.pom
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/wagon/wagon/1.0-beta-4/wagon-1.0-beta-4.pom (11 KB at 21.3 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-archiver/1.0-alpha-7/plexus-archiver-1.0-alpha-7.pom (2 KB at 2.0 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-components/1.1.6/plexus-components-1.1.6.pom
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-components/1.1.6/plexus-components-1.1.6.pom (2 KB at 3.8 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-utils/1.2/plexus-utils-1.2.pom (767 B at 1.5 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus/1.0.5/plexus-1.0.5.pom
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus/1.0.5/plexus-1.0.5.pom (6 KB at 11.6 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/mortbay/jetty/jetty-6.1.5/jetty-6.1.5.pom (5 KB at 8.2 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/mortbay/jetty/project/6.1.5/project-6.1.5.pom (13 KB at 26.5 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/mortbay/jetty/jetty-util/6.1.5/jetty-util-6.1.5.pom (4 KB at 7.7 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/mortbay/jetty/jetty-util/6.1.5/jetty-util-6.1.5.pom (4 KB at 7.7 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/mortbay/jetty/servlet-api-2.5/6.1.5/servlet-api-2.5-6.1.5.pom (4 KB at 8.0 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/mortbay/jetty/servlet-api-2.5/6.1.5/servlet-api-2.5-6.1.5.pom (4 KB at 8.0 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/mortbay/jetty/jetty-util/6.1.5/jetty-util-6.1.5.jar (135 KB at 54.4 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/mortbay/jetty/jetty-6.1.5/jetty-6.1.5.jar (475 KB at 174.8 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/mortbay/jetty/servlet-api-2.5/6.1.5/servlet-api-2.5-6.1.5.jar (130 KB at 87.9 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/skins/maven-default-skin/maven-metadata.xml (341 B at 1.1 KB/sec)
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/skins/maven-default-skin/1.0/maven-default-skin-1.0.jar (8 KB at 15.9 KB/sec)
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 10.595s
[INFO] Finished at: Thu Nov 03 20:42:58 CET 2011
[INFO] Final Memory: 7M/17M
[INFO]
C:\TDS\MavenTestProject>
```

generará en **target\site** una documentación web por defecto, incluyendo el **javadoc**. En el **pom.xml** y en algunos ficheros adicionales de configuración se podría personalizar el aspecto de la documentación.

## Parte 7. Añadir dependencias

Podemos añadir dependencias a nuestro proyecto en forma de artefactos o librerías .jar. Basta modificar el archivo principal **pom.xml** e indicarlo en el apartado de dependencias.

```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

Vemos que hay un apartado **dependencies** y que dentro tiene un **dependency** de JUnit. Para esta dependencia de JUnit hay que indicar el **groupId** (junit), el **artifactId** (junit otra vez), la **versión** que deseamos (3.8.1) y cuándo la necesitamos (en los test)

Vamos a añadir, por ejemplo, a nuestro proyecto la librería de Log4J. Debemos incluir entonces en nuestro pom.xml lo siguiente:

```
<dependency>
  <groupId>log4j</groupId>
  <artifactId>log4j</artifactId>
  <version>1.2.16</version>
  <scope>compile</scope>
</dependency>
```

Para poder localizar los identificadores adecuados de la dependencia, iremos al repositorio oficial de Maven con los plugins (<http://mvnrepository.com/plugins.html>) y buscaremos el plugin, en este caso log4j. Entonces seleccionaremos la implementación y la versión, en nuestro caso la de apache y la 1.2.16, y obtendremos el código XML a añadir al POM.

Esta vez se ha puesto "compile" en vez de "test" para indicar que lo necesitamos al compilar nuestro proyecto y no al hacerle el test. Ya podemos usar en nuestro código fuente la librería log4j. Cuando compilemos la primera vez, **Maven** mirará si ya tenemos log4j en nuestro repositorio local y si no lo tenemos, se irá a buscarlo a su repositorio remoto y lo bajará. El log4j quedará entonces instalado en nuestro repositorio local.

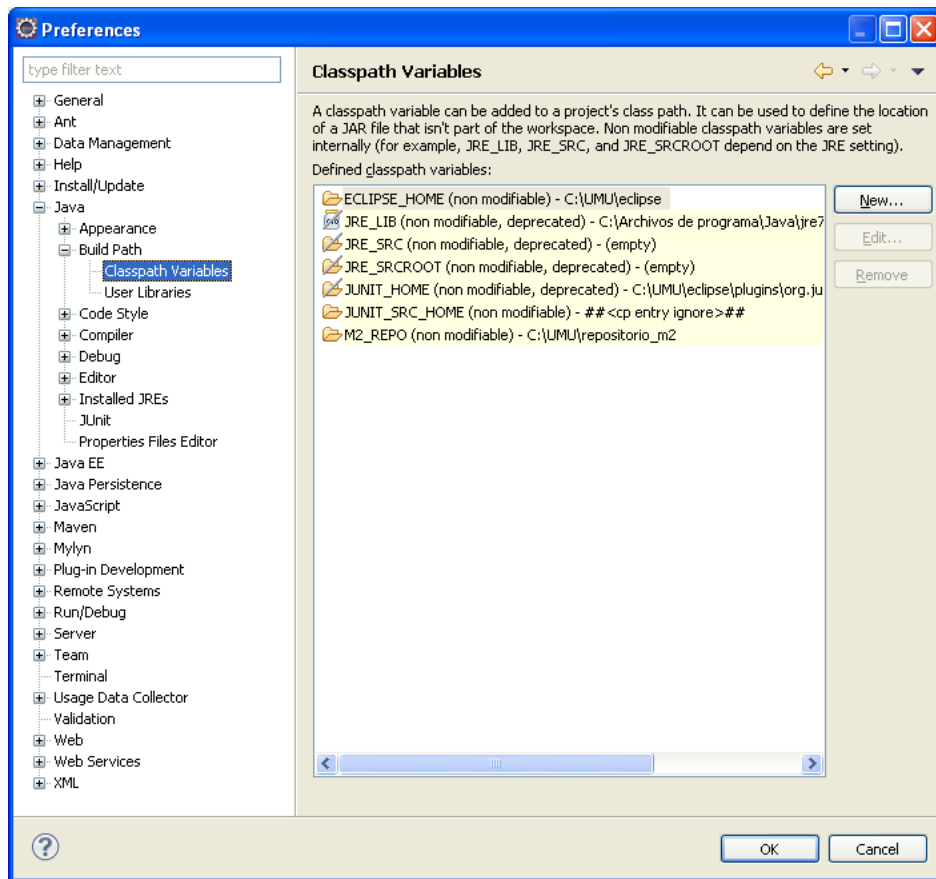
## Parte 8. Integración con Eclipse

Vamos a ver ahora como integrar Maven con nuestro entorno de desarrollo Eclipse. Se puede realizar de dos formas: con el **propio JDT de Eclipse** o con un **plugin especial de Eclipse**.

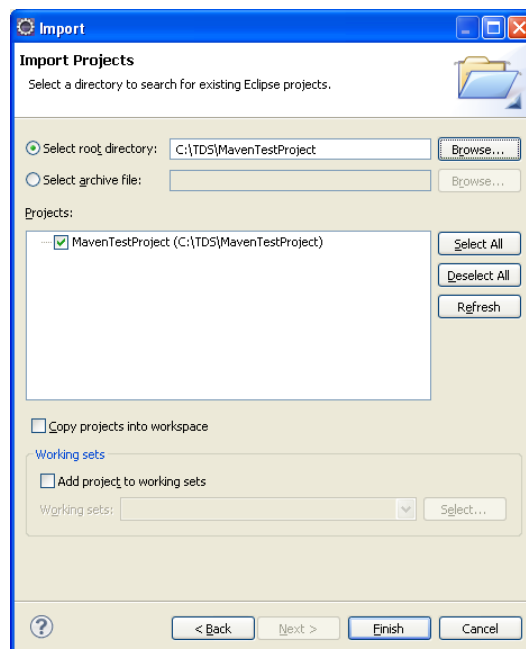
En ambos casos, podemos generar los ficheros del proyecto Eclipse a partir del POM ejecutando:

```
mvn eclipse:eclipse
```



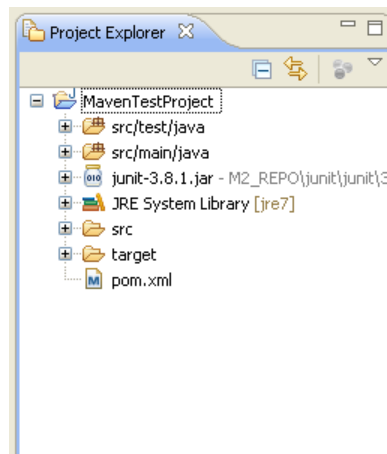


Sólo nos queda importar nuestro proyecto: File → Import → General → Existing projects into workspace

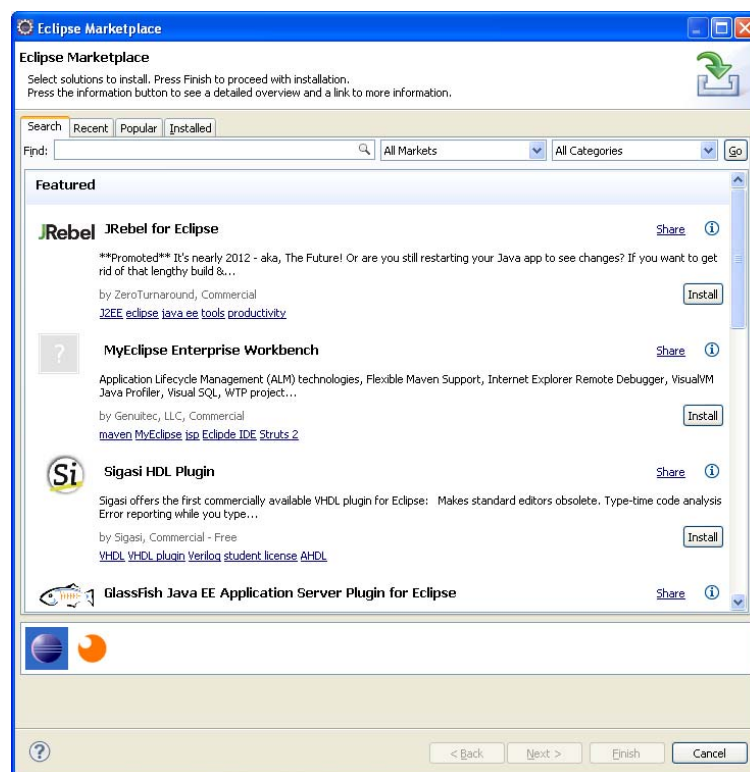


Nota: Importante, no marcar 'Copy projects into workspace' para que el proyecto fuese quede en el directorio que estamos usando con Maven

Podemos comprobar en el 'Project Explorer' que las dependencias con JUnit se ligaron correctamente:

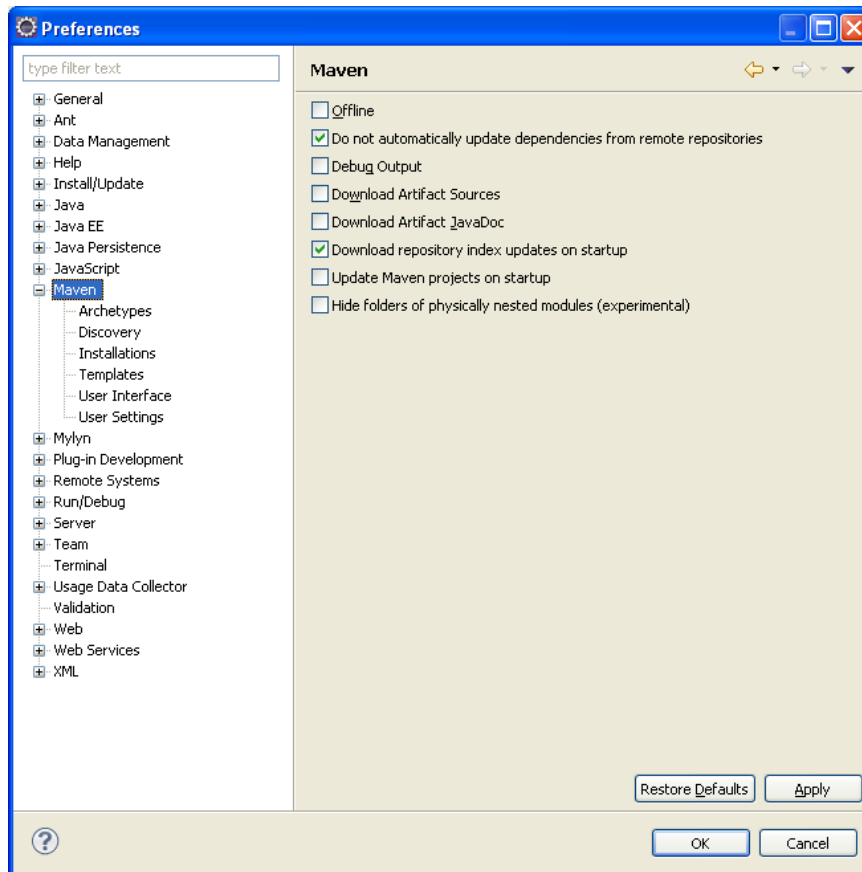


Ahora sería posible compaginar las tareas de programación en Eclipse con las de gestión de proyectos Maven desde la línea de comandos. Sin embargo, vamos a **utilizar un plugin específico de Eclipse para Maven**. Existen varios y nosotros utilizaremos el plugin M2Eclipse que lo instalaremos desde el Marketplace de Eclipse: Help → Eclipse Marketplace ...

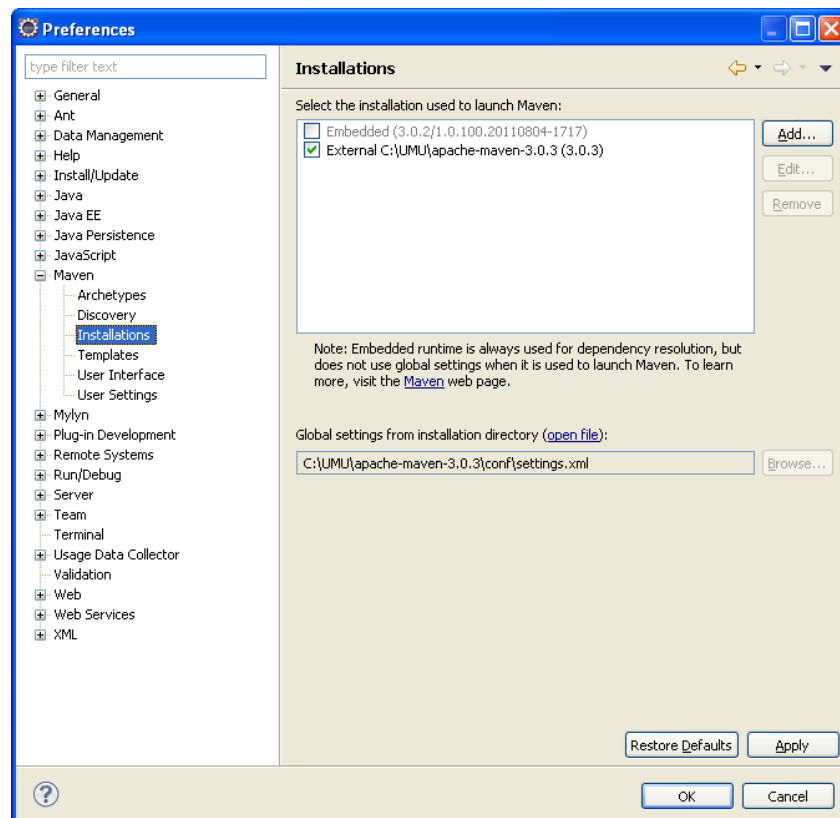


En la entrada 'Find' escribiremos 'maven' y de la lista de plugins de Eclipse a instalar seleccionaremos: 'Maven Integration for Eclipse' y haremos click en el botón 'install' a su derecha. Una vez instalado (tardará tiempo en calcular dependencias y descargar), deberemos reiniciar Eclipse.

Para configurar el plugin M2Eclipse, nos vamos a Windows → Preferences

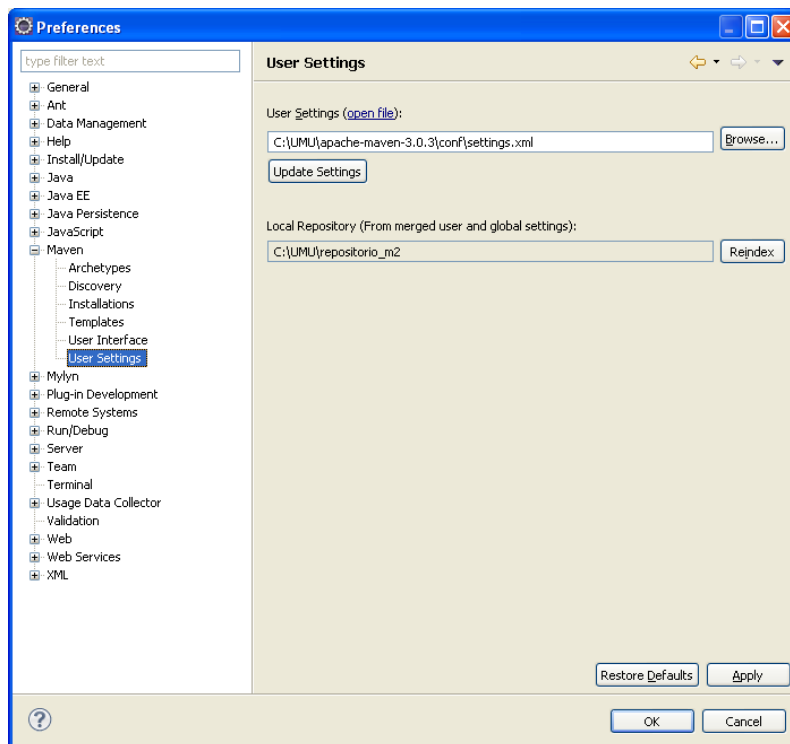


Debemos configurar las opciones de la ventana 'Installations' para que referencie a nuestra instalación de Maven:

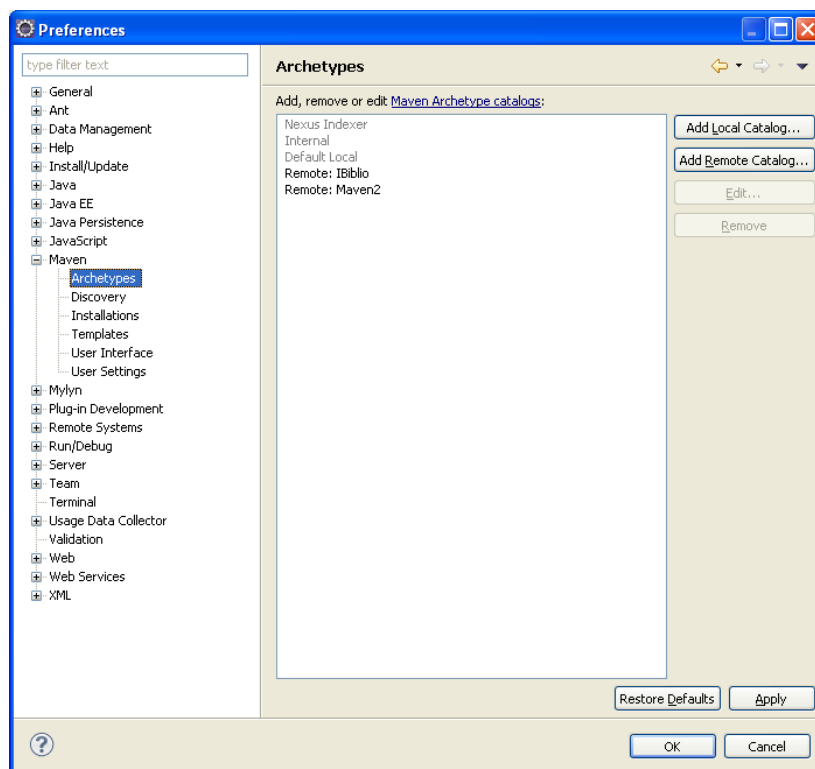




...también la de 'User settings':

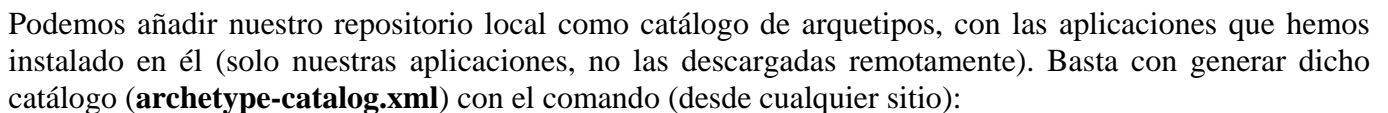


Y opcionalmente podemos añadir catálogos de arquetipos (tipos de proyectos) en la ventana 'Archetypes':



Se han añadido por ejemplo, dos repositorios de Maven:

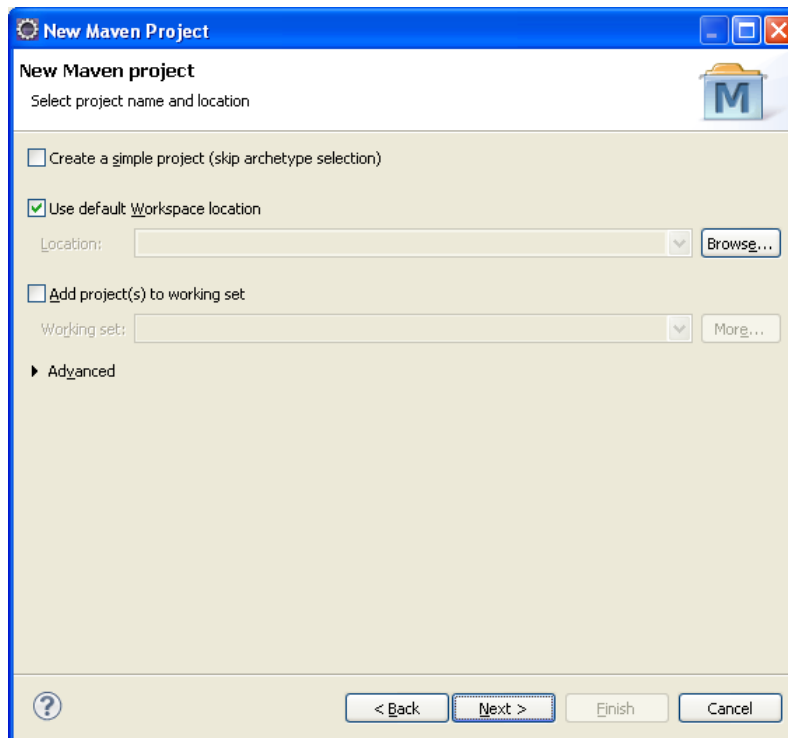
<http://repo1.maven.org/maven2/archetype-catalog.xml>



```
C:\WINDOWS\system32\cmd.exe
```

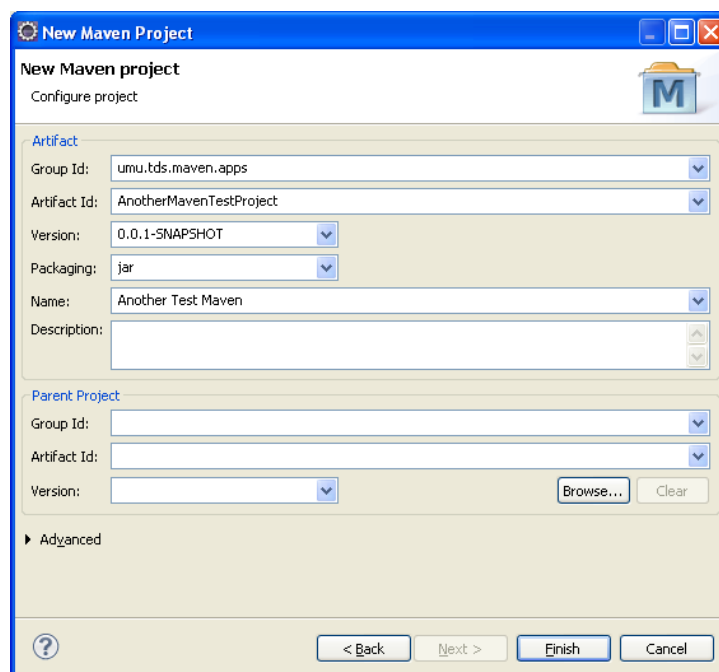
```
[INFO] Scanning C:\YDS..m2\repository\org\apache\maven\plugins\maven-site-plugin-2.0.1\maven-site-plugin-2.0.1.jar
[INFO] Scanning C:\YDS..m2\repository\org\apache\maven\plugins\maven-surefire-plugin-2.7.2\maven-surefire-plugin-2.7.2.jar
[INFO] Scanning C:\YDS..m2\repository\org\apache\maven\reporting\maven-reporting-api-2.0.6\maven-reporting-api-2.0.6.jar
[INFO] Scanning C:\YDS..m2\repository\org\apache\maven\reporting\maven-reporting-api-2.0.8\maven-reporting-api-2.0.8.jar
[INFO] Scanning C:\YDS..m2\repository\org\apache\maven\reporting\maven-reporting-api-2.0.9\maven-reporting-api-2.0.9.jar
[INFO] Scanning C:\YDS..m2\repository\org\apache\maven\reporting\maven-reporting-api-3.0\maven-reporting-api-3.0.jar
[INFO] Scanning C:\YDS..m2\repository\org\apache\maven\shared\maven-common-artifact-filters-1.3\maven-common-artifact-filters-1.3.jar
[INFO] Scanning C:\YDS..m2\repository\org\apache\maven\shared\maven-doxia-tools-1.0.2\maven-doxia-tools-1.0.2.jar
[INFO] Scanning C:\YDS..m2\repository\org\apache\maven\shared\maven-filtering-1.0-beta-4\maven-filtering-1.0-beta-4.jar
[INFO] Scanning C:\YDS..m2\repository\org\apache\maven\shared\maven-invoker-2.0.10\maven-invoker-2.0.10.jar
[INFO] Scanning C:\YDS..m2\repository\org\apache\maven\shared\maven-invoker-2.0.9\maven-invoker-2.0.9.jar
[INFO] Scanning C:\YDS..m2\repository\org\apache\maven\shared\maven-osgi-0.2.0\maven-osgi-0.2.0.jar
[INFO] Scanning C:\YDS..m2\repository\org\apache\maven\skins\maven-default-skin-1.0\maven-default-skin-1.0.jar
[INFO] Scanning C:\YDS..m2\repository\org\apache\maven\surefire\maven-surefire-common-2.7.2\maven-surefire-common-2.7.2.jar
[INFO] Scanning C:\YDS..m2\repository\org\apache\maven\surefire\surefire-api-2.7.2\maven-surefire-api-2.7.2.jar
[INFO] Scanning C:\YDS..m2\repository\org\apache\maven\surefire\surefire-booter-2.7.2\maven-surefire-booter-2.7.2.jar
[INFO] Scanning C:\YDS..m2\repository\org\apache\maven\surefire\surefire-junit3-2.7.2\maven-surefire-junit3-2.7.2.jar
[INFO] Scanning C:\YDS..m2\repository\org\apache\velocity\velocity-1.5\velocity-1.5.jar
[INFO] Scanning C:\YDS..m2\repository\org.codehaus\plexus\plexus-archiver-1.0-alpha-7\plexus-archiver-1.0-alpha-7.jar
[INFO] Scanning C:\YDS..m2\repository\org.codehaus\plexus\plexus-archiver-1.0-alpha-7\plexus-archiver-1.0-alpha-7.jar
[INFO] Scanning C:\YDS..m2\repository\org.codehaus\plexus\plexus-archiver-1.0-alpha-7\plexus-archiver-1.0-alpha-7.jar
[INFO] Scanning C:\YDS..m2\repository\org.codehaus\plexus\plexus-compiler-api-1.8.1\plexus-compiler-api-1.8.1.jar
[INFO] Scanning C:\YDS..m2\repository\org.codehaus\plexus\plexus-compiler-javac-1.8.1\plexus-compiler-javac-1.8.1.jar
[INFO] Scanning C:\YDS..m2\repository\org.codehaus\plexus\plexus-compiler-manager-1.8.1\plexus-compiler-manager-1.8.1.jar
[INFO] Scanning C:\YDS..m2\repository\org.codehaus\plexus\plexus-digest-1.0\plexus-digest-1.0.jar
[INFO] Scanning C:\YDS..m2\repository\org.codehaus\plexus\plexus-ibsn-1.0-beta-7\plexus-ibsn-1.0-beta-7.jar
[INFO] Scanning C:\YDS..m2\repository\org.codehaus\plexus\plexus-interactivity-api-1.0-alpha-4\plexus-interactivity-api-1.0-alpha-4.jar
[INFO] Scanning C:\YDS..m2\repository\org.codehaus\plexus\plexus-interactivity-api-1.0-alpha-5\plexus-interactivity-api-1.0-alpha-5.jar
[INFO] Scanning C:\YDS..m2\repository\org.codehaus\plexus\plexus-interactivity-jline-1.0-alpha-5\plexus-interactivity-jline-1.0-alpha-5.jar
[INFO] Scanning C:\YDS..m2\repository\org.codehaus\plexus\plexus-interpolation-1.13\plexus-interpolation-1.13.jar
[INFO] Scanning C:\YDS..m2\repository\org.codehaus\plexus\plexus-io-1.0-alpha-1\plexus-io-1.0-alpha-1.jar
[INFO] Scanning C:\YDS..m2\repository\org.codehaus\plexus\plexus-io-1.0-alpha-1\plexus-io-1.0-alpha-1.jar
[INFO] Scanning C:\YDS..m2\repository\org.codehaus\plexus\plexus-resources-1.0-alpha-7\plexus-resources-1.0-alpha-7.jar
[INFO] Scanning C:\YDS..m2\repository\org.codehaus\plexus\plexus-utils-1.5\plexus-utils-1.5.1.jar
[INFO] Scanning C:\YDS..m2\repository\org.codehaus\plexus\plexus-utils-1.5.6\plexus-utils-1.5.6.jar
[INFO] Scanning C:\YDS..m2\repository\org.codehaus\plexus\plexus-utils-1.5.8\plexus-utils-1.5.8.jar
[INFO] Scanning C:\YDS..m2\repository\org.codehaus\plexus\plexus-utils-2.0.5\plexus-utils-2.0.5.jar
[INFO] Scanning C:\YDS..m2\repository\org.codehaus\plexus\plexus-velocity-1.7\plexus-velocity-1.7.jar
[INFO] Scanning C:\YDS..m2\repository\org.codehaus\plexus\plexus-velocity-1.8\plexus-velocity-1.8.jar
[INFO] Scanning C:\YDS..m2\repository\org.eclipse\core\resources-3.3.0-v20070604\resources-3.3.0-v20070604.jar
[INFO] Scanning C:\YDS..m2\repository\org.mortbay.jetty\jetty-6.1.5\jetty-6.1.5.jar
[INFO] Scanning C:\YDS..m2\repository\org.mortbay.jetty\jetty-util-6.1.5\jetty-util-6.1.5.jar
[INFO] Scanning C:\YDS..m2\repository\org.mortbay.servlet-api-2.5.6-1.5\servlet-api-2.5-6-1.5.jar
[INFO] Scanning C:\YDS..m2\repository\org.sonatype\plexus\plexus-build-api-0.0.4\plexus-build-api-0.0.4.jar
[INFO] Scanning C:\YDS..m2\repository\org.snowy\c-0.0\c-0.0.jar
[INFO] Scanning C:\YDS..m2\repository\org.springframework.test\spring-test-1.0-SNAPSHOT\MavenTestProject-1.0-SNAPSHOT.jar
[INFO] Scanning C:\YDS..xml-apix\xml-apix-1.0-h2.xml-apix-1.0-h2.jar
[INFO] BUILD SUCCESS
[INFO] Total time: 1.322s
[INFO] Finished at: Thu Nov 03 21:21:34 CET 2011
[INFO] Final Memory: 49M/15M
C:\YDS>
```

Estamos en disposición de crear nuevos proyectos con Maven, usando arquetipos, sin necesidad de emplear la línea de comandos. Basta con seleccionar File → New → Maven → Maven Project

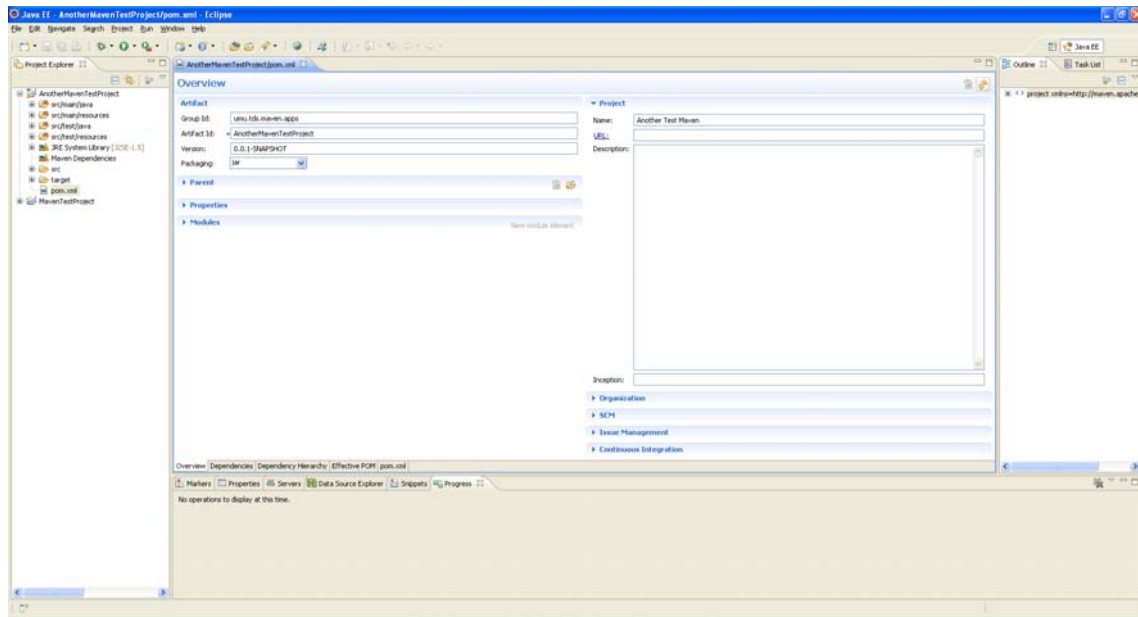


Nota: Se pueden crear módulos en Maven y proyectos de equipo para gestionar con herramientas como CVS.

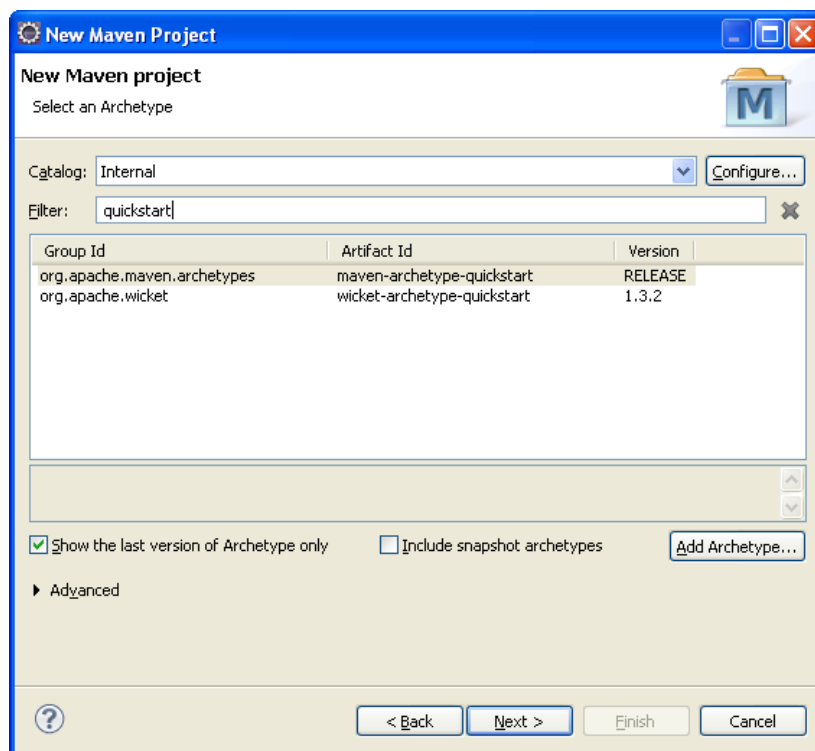
Al crear el proyecto con Maven, podemos indicar que queremos crear un proyecto simple y no partir de ningún arquetipo (marcar la casilla superior entonces):



Se creará entonces un proyecto sin dependencias y con la estructura básica (carpetas *java* y *resources* en cada una de las carpetas **main** y **test**). Se generará el pom.xml que podremos editar con el editor especial de POM que incorpora el plugin.

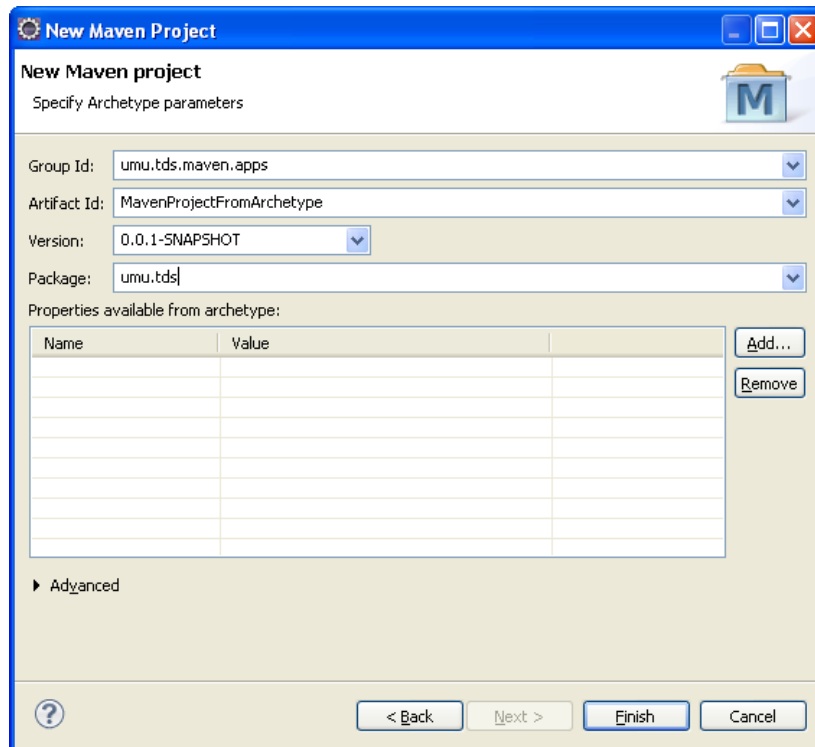


Vamos a crear un proyecto partiendo de un arquetipo. Seleccionaremos el mismo arquetipo que usamos desde la línea de comandos sobre el catálogo 'Internal':

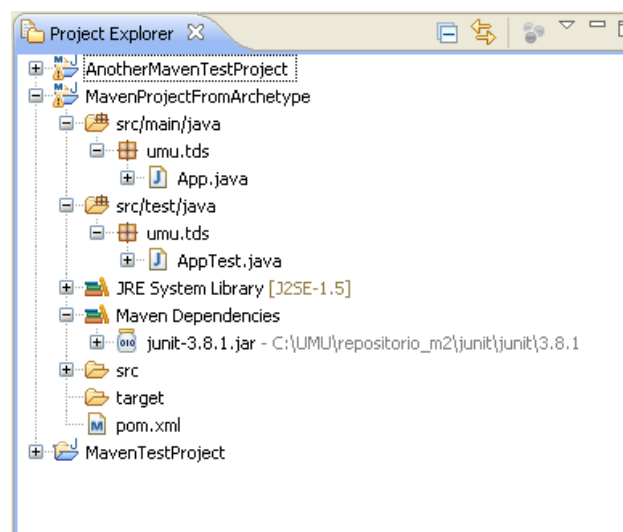


Nota: Los arquetipos de los diversos catálogos que se configuraron anteriormente quedan disponibles ahora para crear nuestros proyectos.

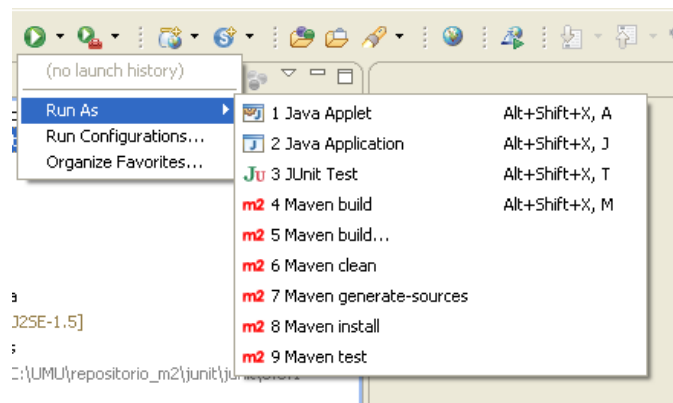
Completamos los datos:



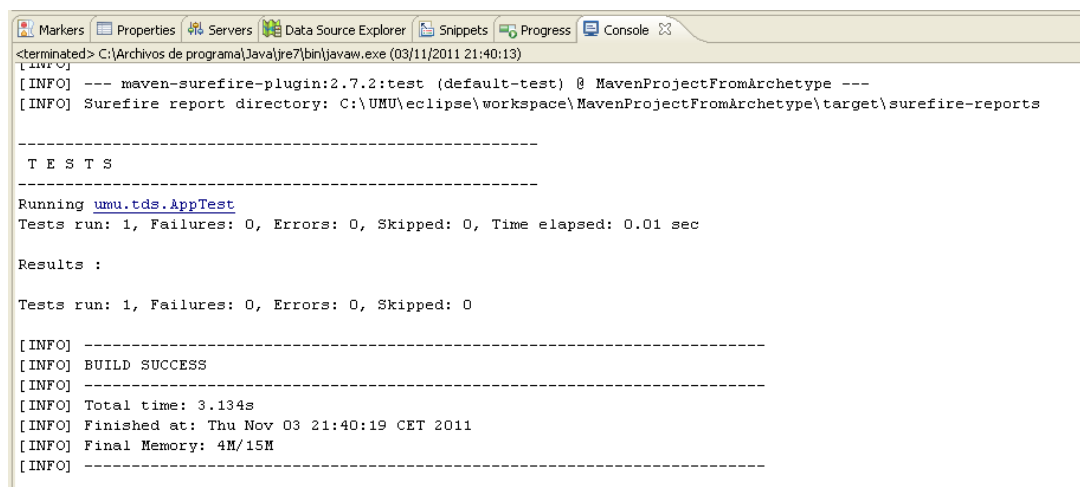
Vemos ahora que el proyecto creado sigue la misma estructura que cuando se creo desde la línea de comandos:



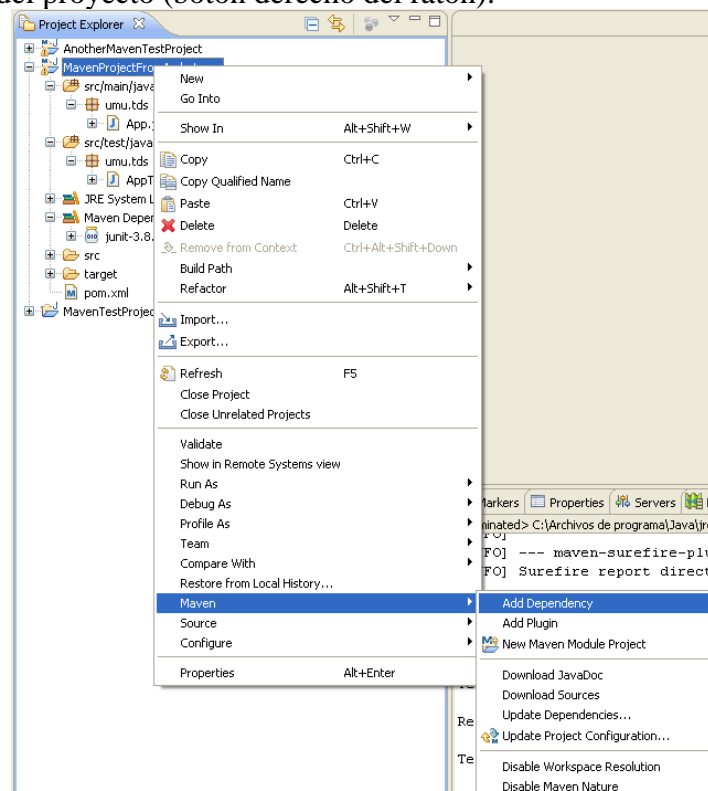
Ahora con el plugin, si seleccionamos el proyecto y hacemos click en 'Run' (de la botonera superior), podremos ejecutar los plugins de Maven para gestionar el ciclo de vida del proyecto:



Por ejemplo ‘Maven test’:

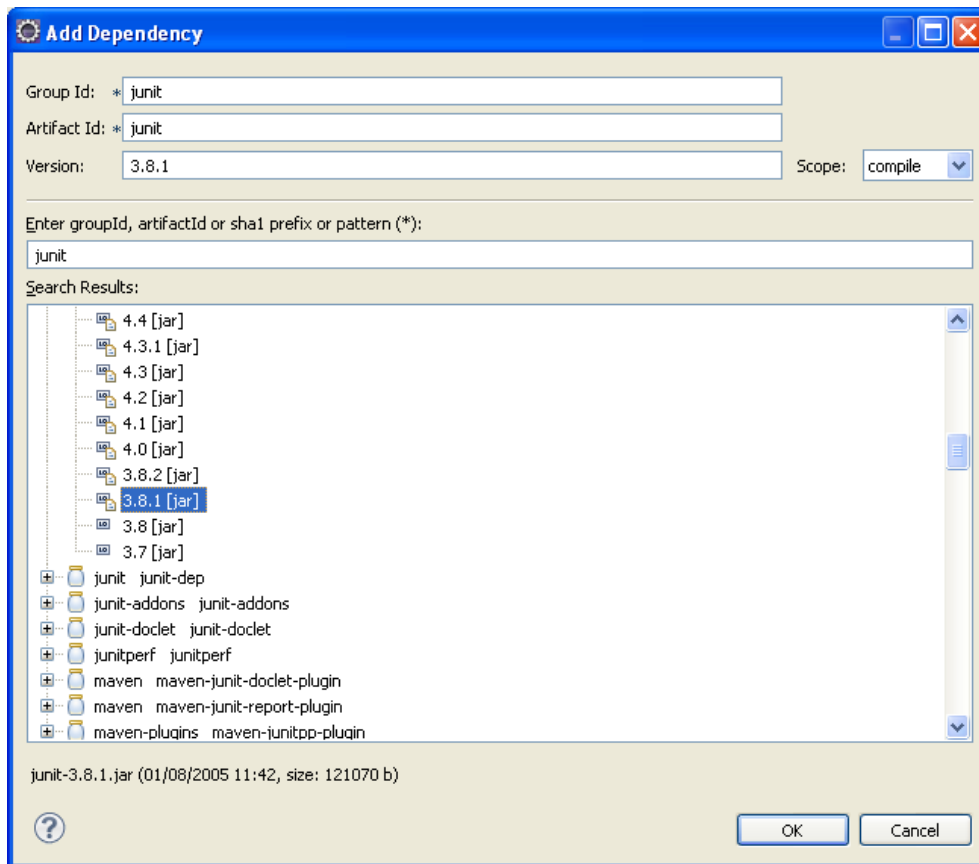


Con el menú contextual del proyecto (boton derecho del ratón):



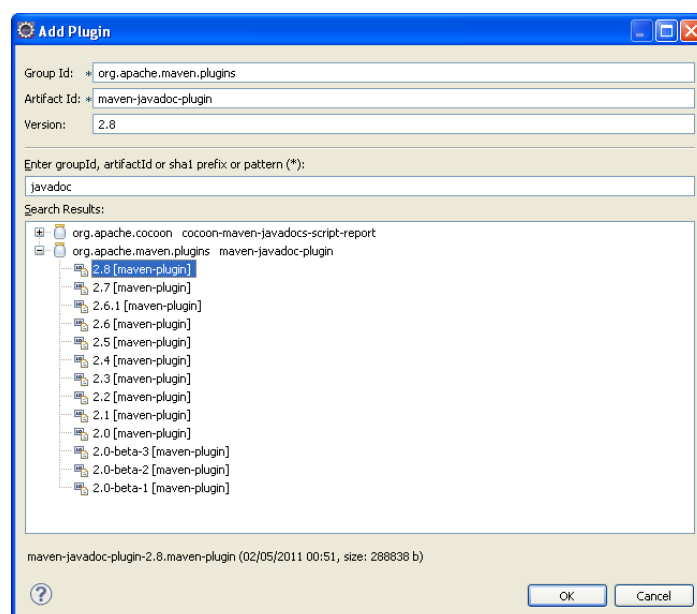


Podremos añadir dependencias, con la gran ventaja de que las búsquedas de la dependencia se hacen automáticamente:

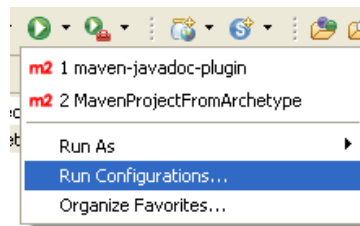


...y ya no es necesario modificar manualmente el **pom.xml**

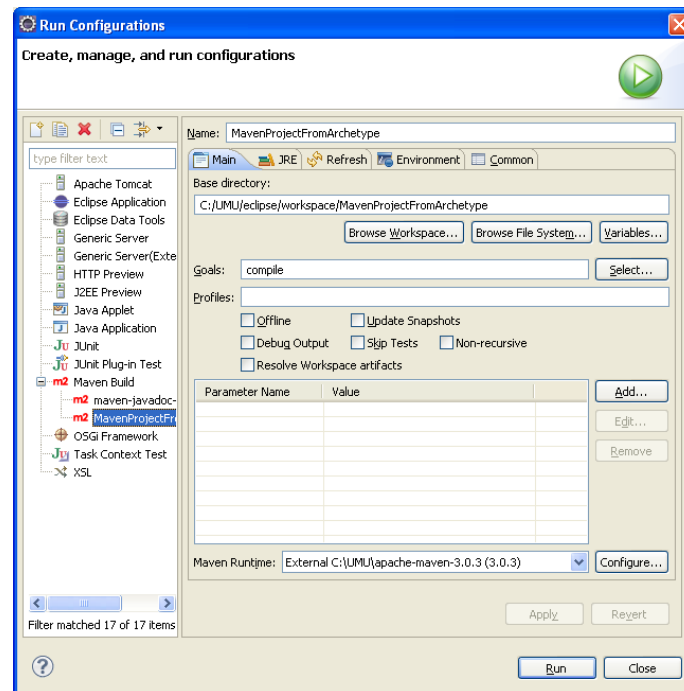
De igual modo podemos añadir un plugin:



Se pueden definir configuraciones de 'Run' para los distintos 'goals' de Maven.



Por ejemplo, 'compile':



Nota: Recordad tener definida una JVM del tipo JDK en Eclipse (y no JRE). Podéis comprobarlo desde Windows → Preferences → Java