

Apache Maven

¿Qué es Apache Maven?



Autor: Ernesto A. Zapata lcart
mail: ezicart@frp.utn.edu.ar

Índice

2

- Introducción: ¿Qué es Maven?
- Ciclo de Vida (Goals)
- Grupos, Artefactos y Arquetipos
- Creación de un proyecto en Maven
- Estructura Básica
- Gestión Declarativa
- Dependencias. ¿Qué son?
- Dependencias. Alcances (*scopes*)

Introducción: ¿Qué es Maven?

3

- Originalmente empezó como un intento de simplificar el proceso de construcción en el proyecto *Jakarta Turbine*.
- Se quería una manera estándar de construir los proyectos, una definición clara de que consistía el proyecto, un medio fácil de publicar información del proyecto y una forma de compartir *jars* entre varios proyectos.
- Maven es un *framework* de gestión de proyectos de software.
- Maven está basado en POM (*Project Object Model*). Cada proyecto tiene la información para su ciclo de vida en un descriptor *xml* (por defecto, el archivo *pom.xml*).
- Maven Proporciona funcionalidades desde la compilación hasta la distribución, despliegue y documentación de los proyectos.
- Maven posee las abstracciones necesarias que animan la reutilización y que ayudan a la estructuración del proyecto.

Introducción: ¿Qué es Maven?

4

- NO ES SÓLO un software para gestionar proyectos de software

“Maven es un sistema de estándares, un repositorio y un software usado para manejar y describir proyectos. Define un ciclo estándar para la construcción, prueba y despliegue de componentes del proyecto. Proporciona un marco que permite la reutilización fácil de la lógica común de la estructura para todos los proyectos que siguen los estándares Maven”

Ciclo de Vida (*Goals*)

5

- *validate* - valida que el proyecto esté correcto y tiene toda la información necesaria para su construcción
- *compile* - compila el código fuente dentro de los artefactos binarios
- *test* - ejecuta los test unitarios, no se precisa que la aplicación esté empaquetada ni desplegada.
- *package* - toma las clases compiladas y recursos y crea un paquete con el proyecto (jar, war, ear)
- *integration-test* - ejecuta test adicionales, que requieren que esté empaquetado
- *verify* - verifica que el empaquetado sea válido
- *install* - instala (copia) el empaquetado dentro de un repositorio local Maven
- *deploy* - despliega el archivo empaquetado en un servidor remoto o en uno local

Grupos, Artefactos y Arquetipos

6

Arquetipo (*archetype*): plantilla del Proyecto. El arquetipo crea la estructura del Proyecto, el contenido del archivo *pom.xml*, la estructura de carpetas y los archivos que incluye por defecto.

Artefacto (*artifact*): Es un proyecto que lo gestiona **Maven** y que incluye un archivo llamado *pom.xml*.

- Son de tipo *jar*, pero podría ser un *war* o un *ear*.
- Pueden tener dependencias entre sí. Si incluimos un artefacto en un proyecto obtendremos sus dependencias.

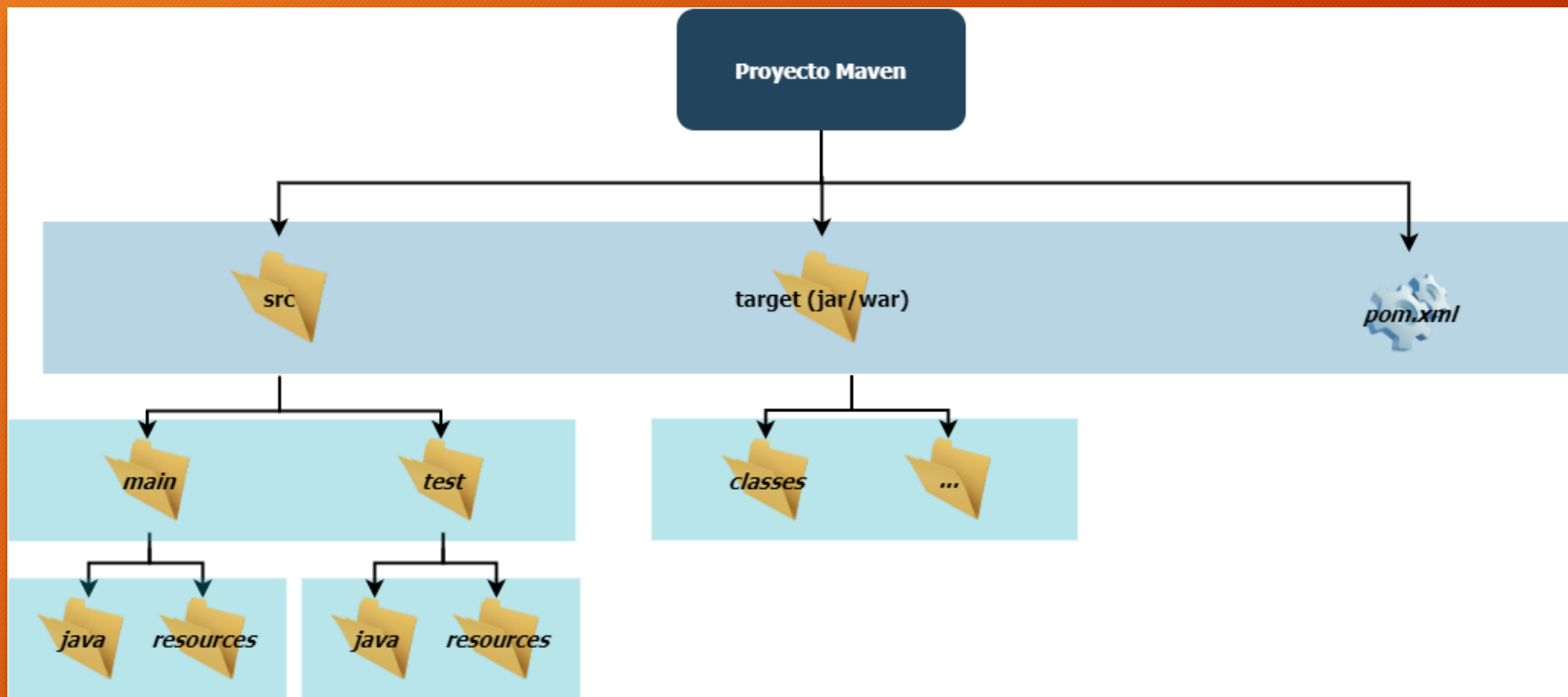
Grupo (*group*): conjunto de artefactos, es una manera de organizarlos.

Creación de un proyecto en Maven

7

Un proyecto en Maven es básicamente una carpeta en la que tenemos un archivo *pom.xml* (descriptor de proyecto). En lugar de crear el proyecto a mano haremos uso de algunas de las plantillas o *archetypes*.

Una de las cosas importantes de la vista de directorio estándar de **Maven**, es que separa los archivos fuentes Java (*.java*) de los archivos de recursos (todo archivo no *.java*).



Estructura básica

Dependiendo del *archetype* elegido, una estructura básica es como la de la imagen...

Gestión declarativa

9

En **Maven** un proyecto se define a partir de su archivo *pom.xml*.

Este archivo se encuentra siempre en la raíz del proyecto y contiene toda información necesaria para el ciclo de vida del proyecto: dependencias, *plugins*, repositorios de donde obtener estos, configuración de los informes, etc.

Dependencias. ¿Qué son?

10

Son aquellos otros componentes que nuestro software necesita en algún momento del ciclo de vida.

Un aspecto interesante de **Maven** es que las dependencias no acompañan al código fuente de nuestro desarrollo.

Según el momento en el que se necesite una dependencia tendrá un ámbito (*scope*) distinto.

Por defecto se entiende que son para compilar. Maven obtiene automáticamente las dependencias del proyecto, bien del repositorio local si ya está, o bien de un repositorio remoto.

Dependencias. Alcances (*scopes*)

11

El “*scope*” sirve para indicar el ámbito o alcance de nuestras dependencias. Pueden ser:

- ***compile***: es el valor por defecto. Indica que la dependencia es necesaria para compilar. Se propaga a los proyectos dependientes.
- ***provided***: como el anterior, pero se espera a que el contenedor (Tomcat, Glassfish, Jboss, etc.) ya tenga esa librería.
- ***runtime***: necesaria para tiempo de ejecución pero no para compilar
- ***test***: sólo necesaria para testing.

Apache Maven y Eclipse

Plugin de Eclipse para Apache Maven

maven

IN



2ª PARTE

Autor: Ernesto A. Zapata lcart
mail: ezicart@frp.utn.edu.ar

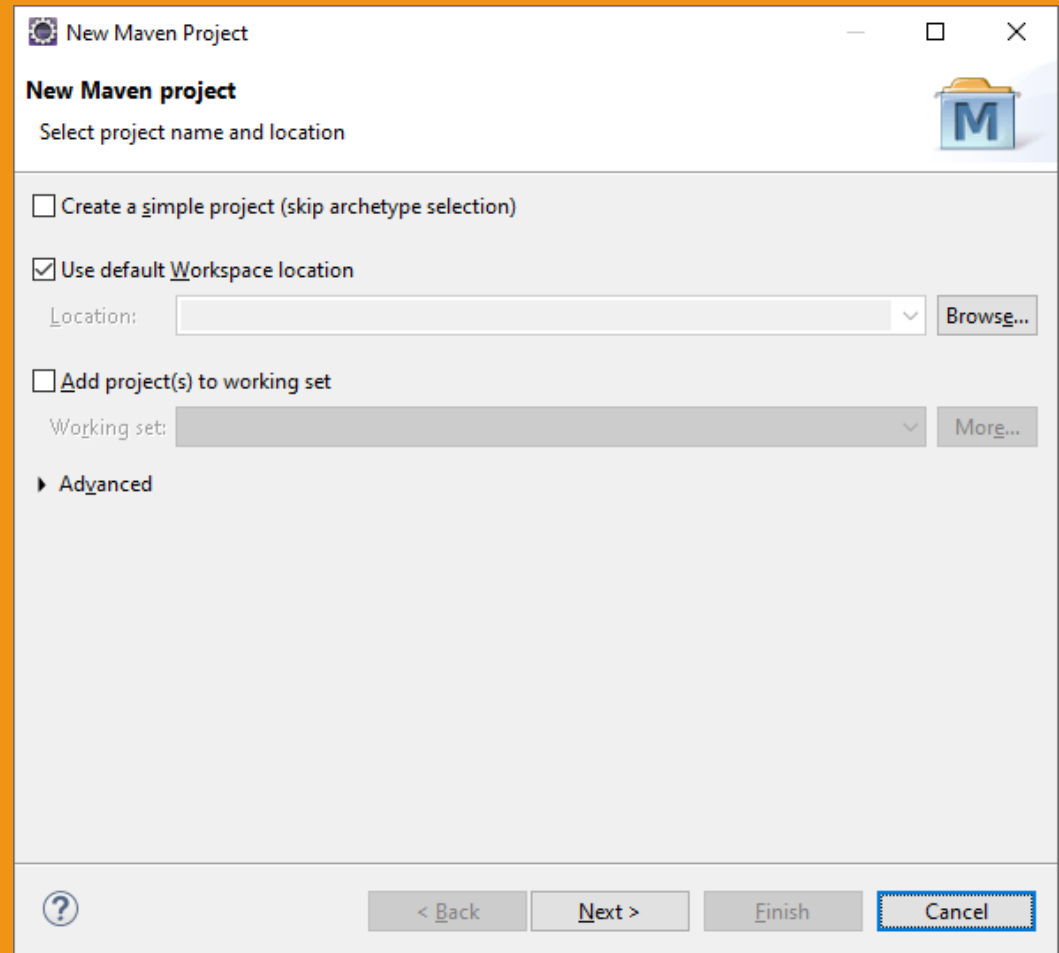


1. Integración con Eclipse

Maven y todas sus herramientas vienen integradas por defecto en la versión para desarrolladores JEE que es la que incluye el plugin de Maven (**M2Eclipse**); aunque es posible que no sea la última versión disponible.

Creación del proyecto Maven

Desde el menú principal del Eclipse seleccionamos *File -> New...Maven Project*.

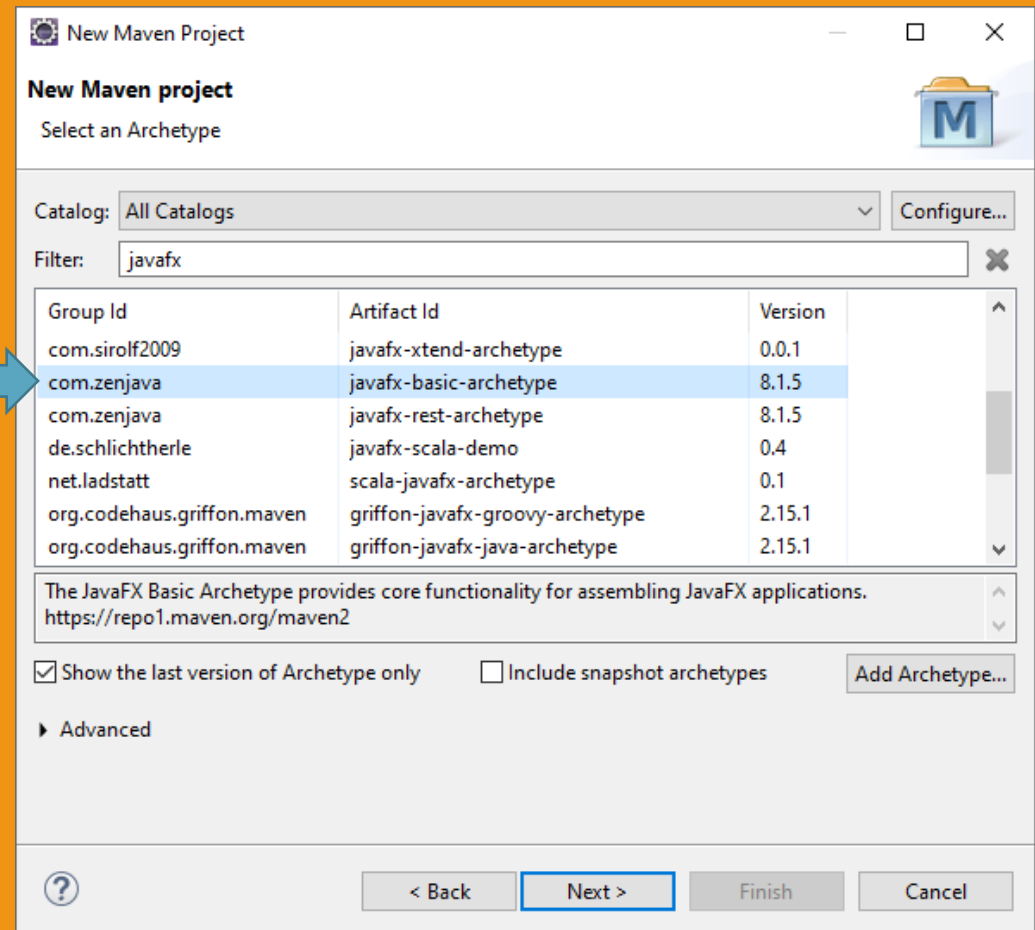


Hacemos “click” en *Next...*

Creación del proyecto Maven

2. Integración con Eclipse

En la opción “*Filter*” ponemos ‘*javafx*’ y elegimos el arquetipo que está marcado en azul.



Hacemos “*click*” en *Next...*

3. Integración con Eclipse

Repasar conceptos de *Group Id*
y *Artifact Id*

Creación del proyecto Maven

Y ahora, por último paso del asistente, tendremos que indicar nuestro propio *Group Id* y el *Artifact Id*.

New Maven Project

New Maven project
Specify Archetype parameters

Group Id:

Artifact Id:

Version:

Package:

Properties available from archetype:

Name	Value
organizationName	UTN

► Advanced

Hacemos “click” en *Finish*

4. Integración con Eclipse

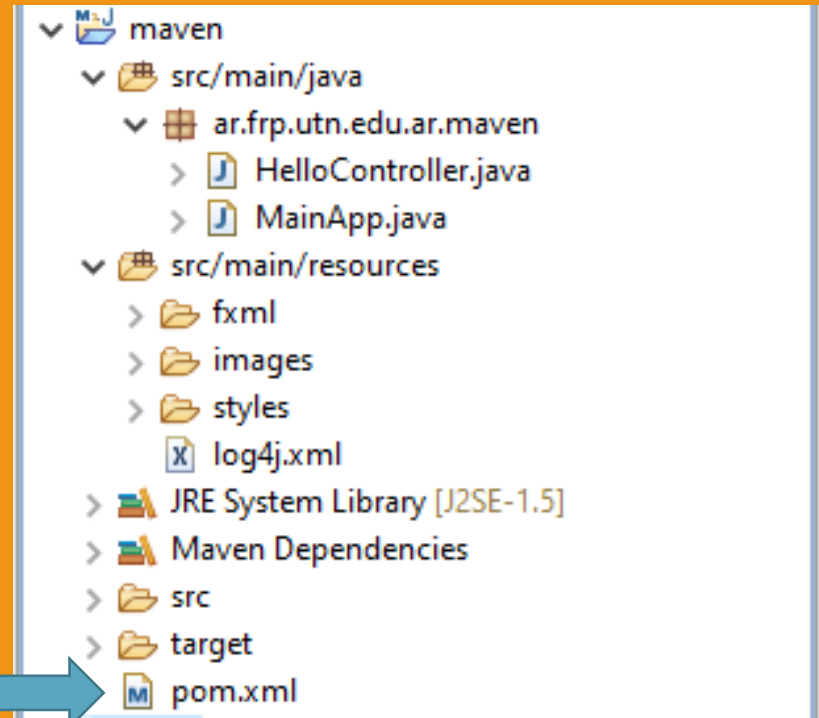
Debieran notar que el *Group Id* es un paquete y el *Artefact Id* es el nombre del proyecto; revisar la teoría al respecto.

Como elegimos un “*archetype*” para JavaFX, nos creó automáticamente dos programas a tal efecto...pero no crea el patron MVC; hay que hacerlo a mano.

Vean también el archivo *pom.xml* en la raíz del proyecto.

Creación del proyecto Maven

Debiera quedar el proyecto conformado tal como se muestra en la figura

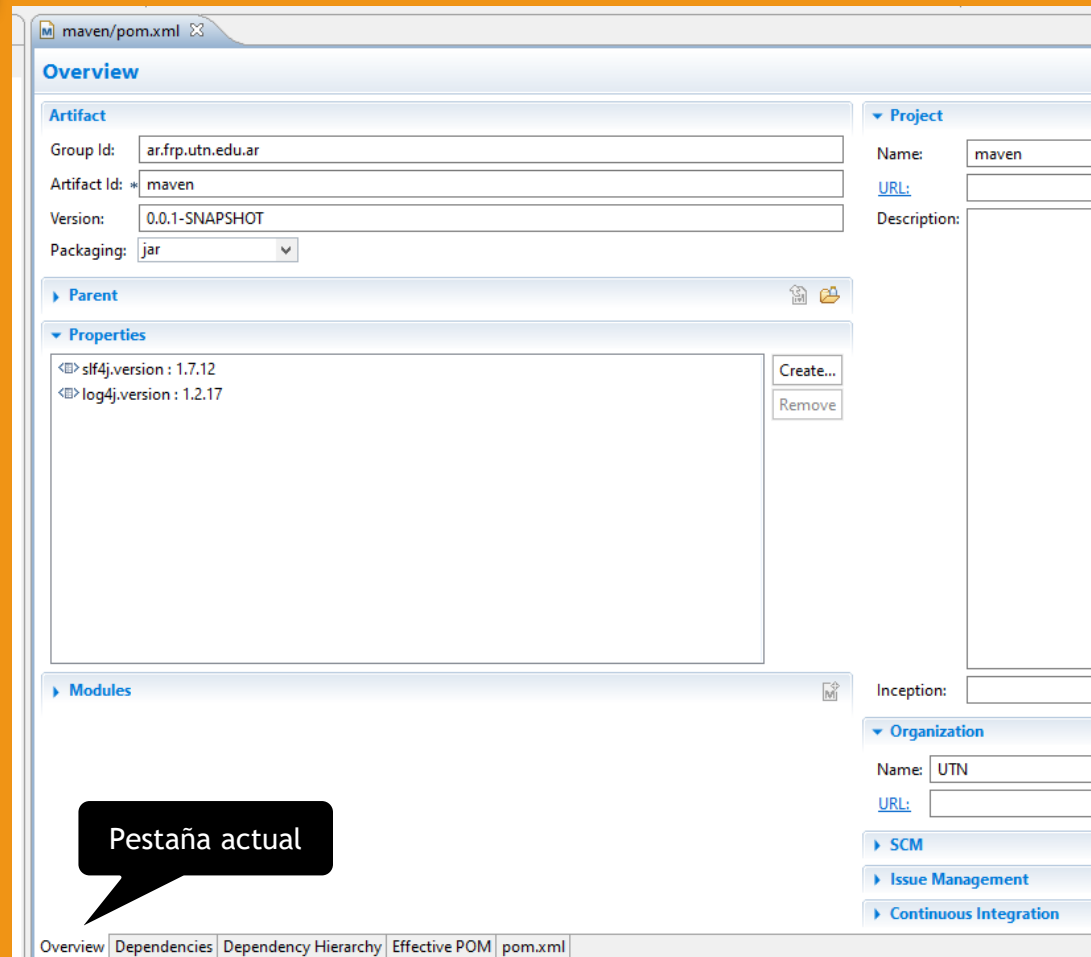


5. Integración con Eclipse

Abajo podemos ver las pestañas de trabajo del archivo *pom.xml*, la de la figura es “Overview”.

Maven en acción

Pulsando <F3> o doble “click” sobre el archivo *pom.xml* debiéramos ver la siguiente figura

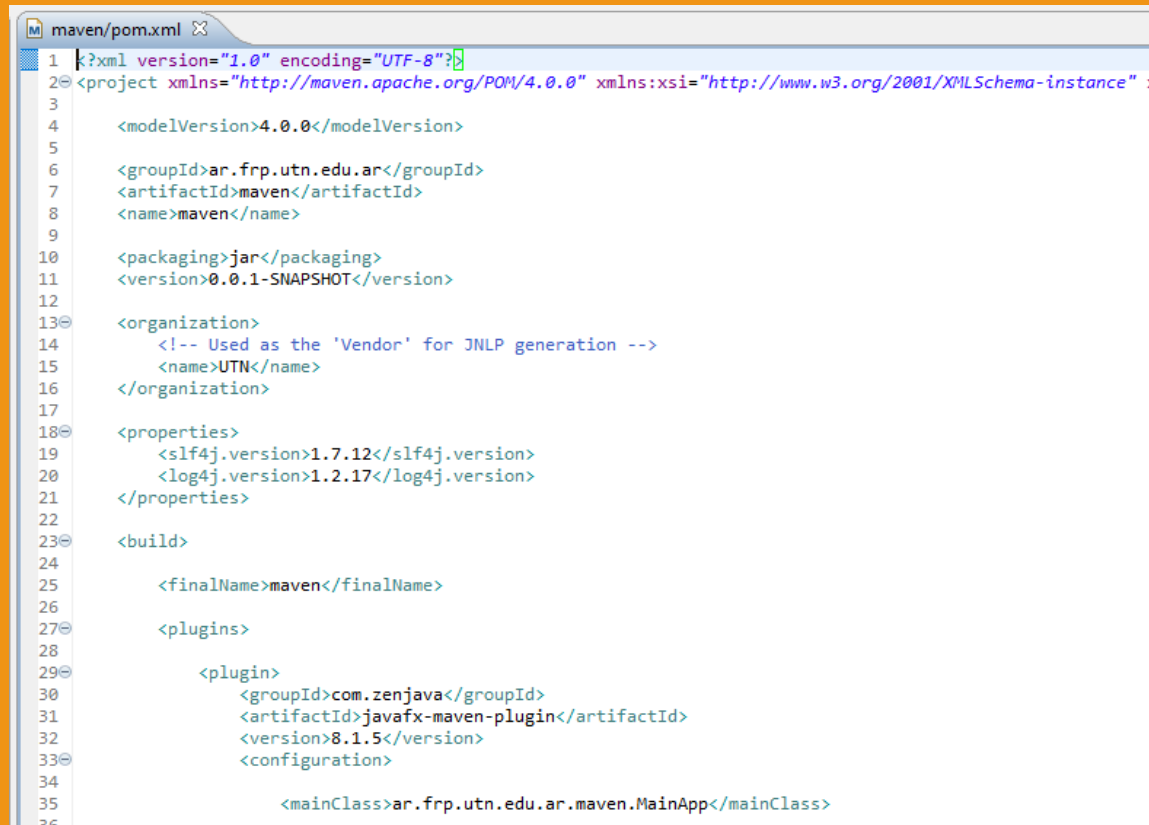


6. Integración con Eclipse

De este código hay trozos que podemos intuir bastante bien con lo mencionado anteriormente. Del resto, el tag de `properties` indicar á las propiedades de nuestro proyecto.

Maven en acción

Haciendo “*click*” sobre la pestaña “*pom.xml*” debiéramos ver la siguiente figura (parcial)

A screenshot of a code editor showing a Maven pom.xml file. The file is titled 'maven/pom.xml' and contains XML code for a Maven project. The code includes tags for modelVersion, groupId, artifactId, name, packaging, version, organization, properties, build, finalName, plugins, and a mainClass. The properties section defines slf4j.version and log4j.version. The build section defines finalName and a plugin for com.zenjava:javaafx-maven-plugin. The mainClass is ar.frp.utn.edu.ar.maven.MainApp.

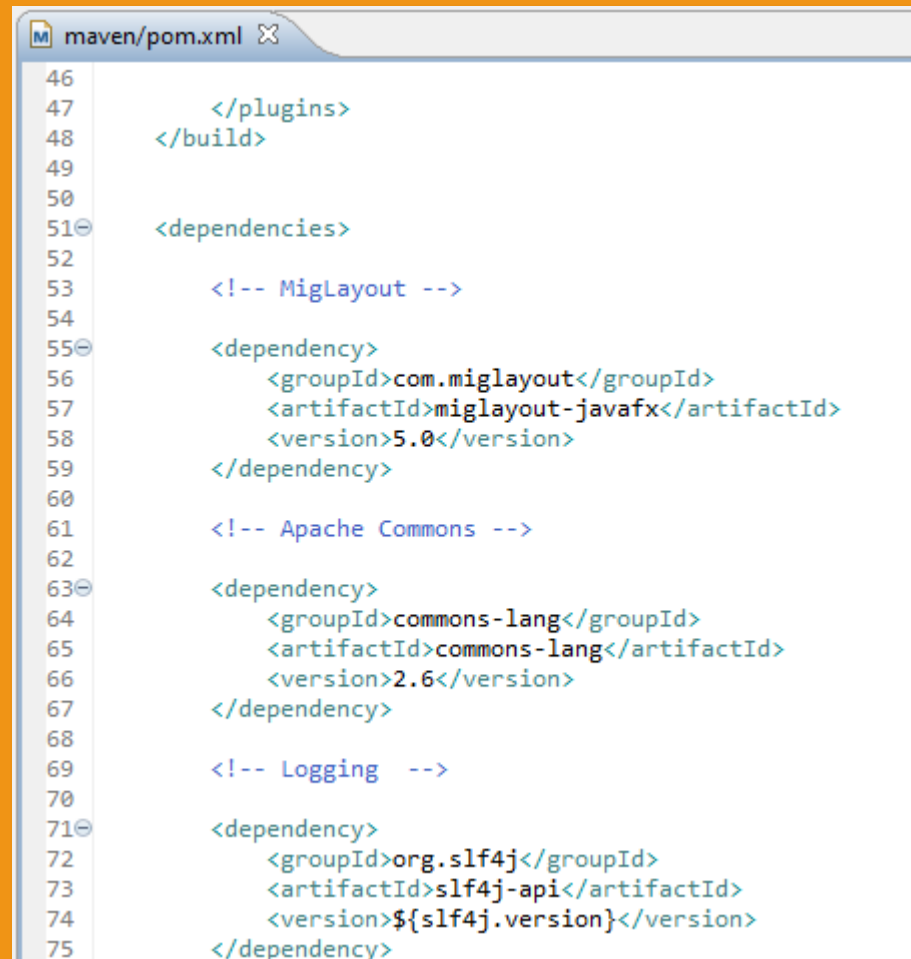
```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3
4     <modelVersion>4.0.0</modelVersion>
5
6     <groupId>ar.frp.utn.edu.ar</groupId>
7     <artifactId>maven</artifactId>
8     <name>maven</name>
9
10    <packaging>jar</packaging>
11    <version>0.0.1-SNAPSHOT</version>
12
13    <organization>
14        <!-- Used as the 'Vendor' for JNLP generation -->
15        <name>UTN</name>
16    </organization>
17
18    <properties>
19        <slf4j.version>1.7.12</slf4j.version>
20        <log4j.version>1.2.17</log4j.version>
21    </properties>
22
23    <build>
24
25        <finalName>maven</finalName>
26
27        <plugins>
28
29            <plugin>
30                <groupId>com.zenjava</groupId>
31                <artifactId>javaafx-maven-plugin</artifactId>
32                <version>8.1.5</version>
33                <configuration>
34
35                    <mainClass>ar.frp.utn.edu.ar.maven.MainApp</mainClass>
36                </configuration>
37            </plugin>
38        </plugins>
39    </build>
40</project>
```

7. Integración con Eclipse

Y luego
tenemos **dependencies**,
que es la sección donde
gestionaremos nuestras
dependencias

Maven en acción

Un poco más abajo sobre el archivo “*pom.xml*”
debiéramos ver la siguiente figura (parcial)



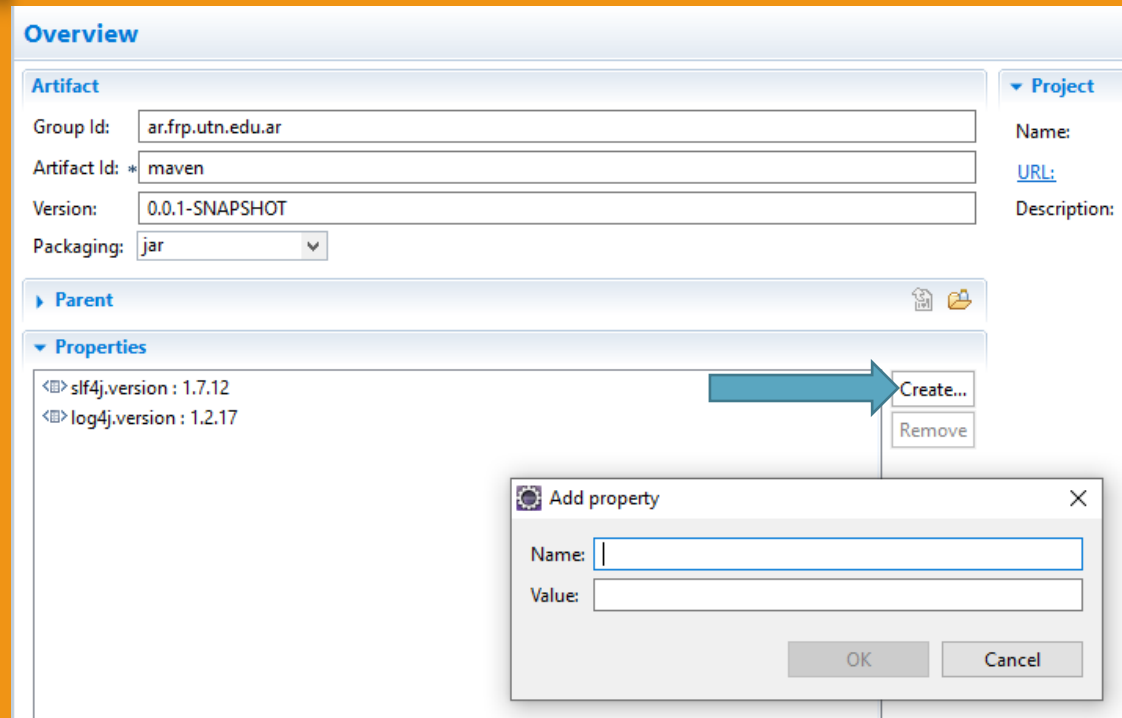
```
maven/pom.xml X
46     </plugins>
47 </build>
48
49
50
51 <dependencies>
52
53     <!-- MigLayout -->
54
55     <dependency>
56         <groupId>com.miglayout</groupId>
57         <artifactId>miglayout-javafx</artifactId>
58         <version>5.0</version>
59     </dependency>
60
61     <!-- Apache Commons -->
62
63     <dependency>
64         <groupId>commons-lang</groupId>
65         <artifactId>commons-lang</artifactId>
66         <version>2.6</version>
67     </dependency>
68
69     <!-- Logging -->
70
71     <dependency>
72         <groupId>org.slf4j</groupId>
73         <artifactId>slf4j-api</artifactId>
74         <version>${slf4j.version}</version>
75     </dependency>
```

8. Integración con Eclipse

De momento lo haremos de manera “sencilla”.

Maven en acción

Vamos a “decirle” a Maven que compilaremos y ejecutaremos con Java 8. Vamos a la pestaña “Overview” y hacemos “click” en el botón “Create” del acordeón (objeto) “*Properties*”.



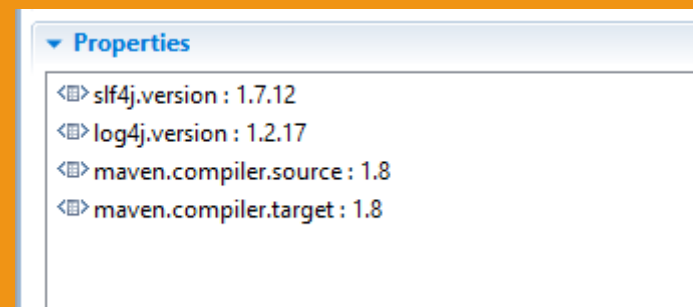
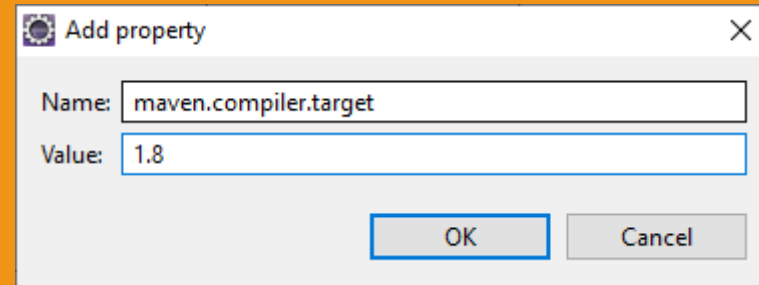
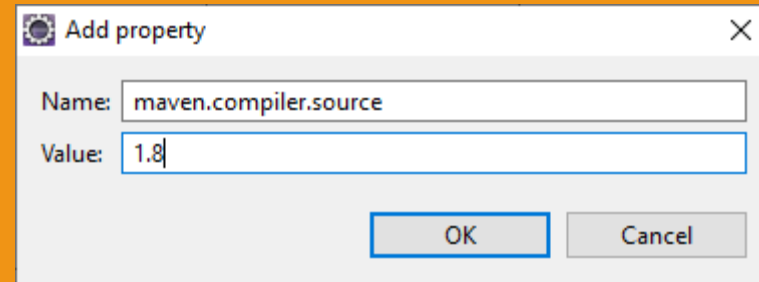
9. Integración con Eclipse

Agreguemos esas dos propiedades con el valor 1.8, que es la versión de Java que queremos que use *Maven*.

Maven en acción

Vamos a agregar dos propiedades:

- *maven.compiler.source*
- *maven.compiler.target*

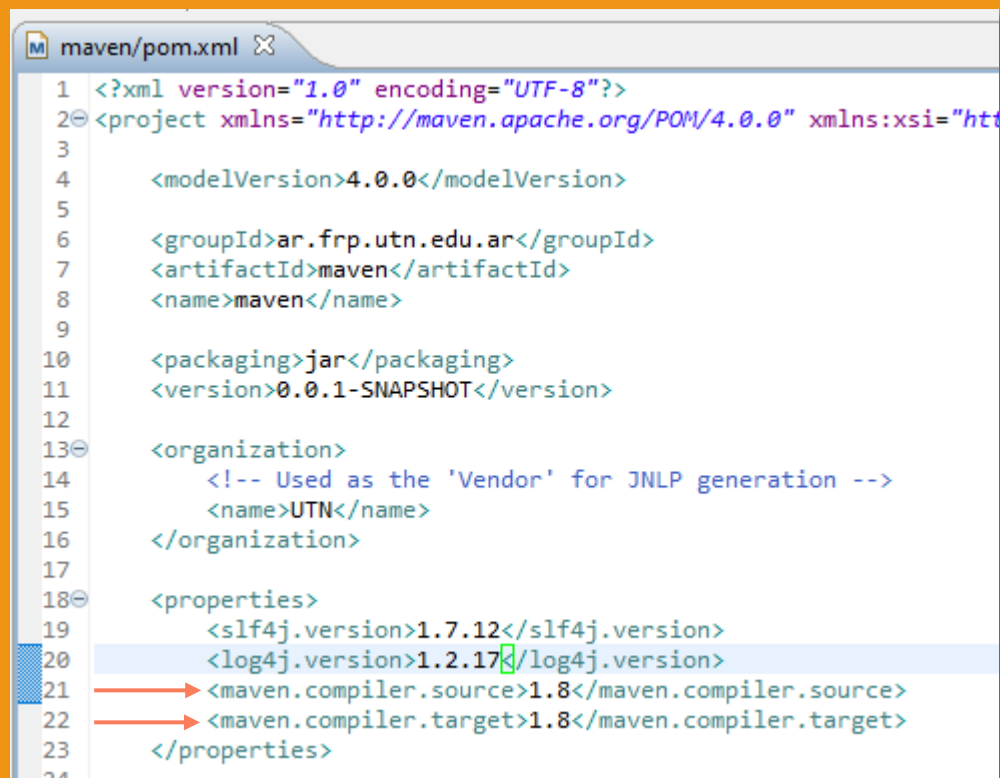


10. Integración con Eclipse

Estas propiedades, como también las dependencias se pueden agregar “a mano”.

Maven en acción

Si vamos a la pestaña “*pom.xml*” debiéramos ver las dos propiedades agregadas



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://maven.apache.org/xsi"
3
4     <modelVersion>4.0.0</modelVersion>
5
6     <groupId>ar.frp.utn.edu.ar</groupId>
7     <artifactId>maven</artifactId>
8     <name>maven</name>
9
10    <packaging>jar</packaging>
11    <version>0.0.1-SNAPSHOT</version>
12
13    <organization>
14        <!-- Used as the 'Vendor' for JNLP generation -->
15        <name>UTN</name>
16    </organization>
17
18    <properties>
19        <slf4j.version>1.7.12</slf4j.version>
20        <log4j.version>1.2.17</log4j.version>
21        <maven.compiler.source>1.8</maven.compiler.source>
22        <maven.compiler.target>1.8</maven.compiler.target>
23    </properties>
24
```

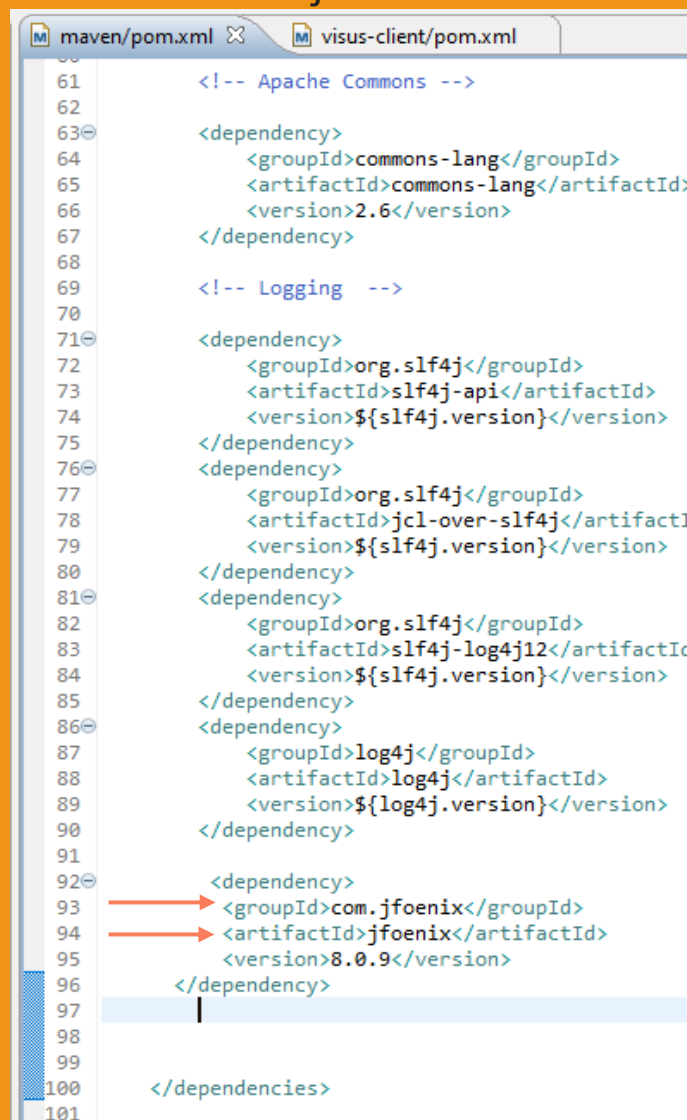
11. Integración con Eclipse

Las dependencias siempre tienen el formato:

```
<dependency>  
  <groupId> [nombre_grupo</groupId>  
<artifactId>[nombre_artefacto]/artifactId>  
  <version>[versión]</version>  
</dependency>
```

Maven en acción

Ahora vamos a agregar una dependencia de nuestra librería de objetos JFoenix...



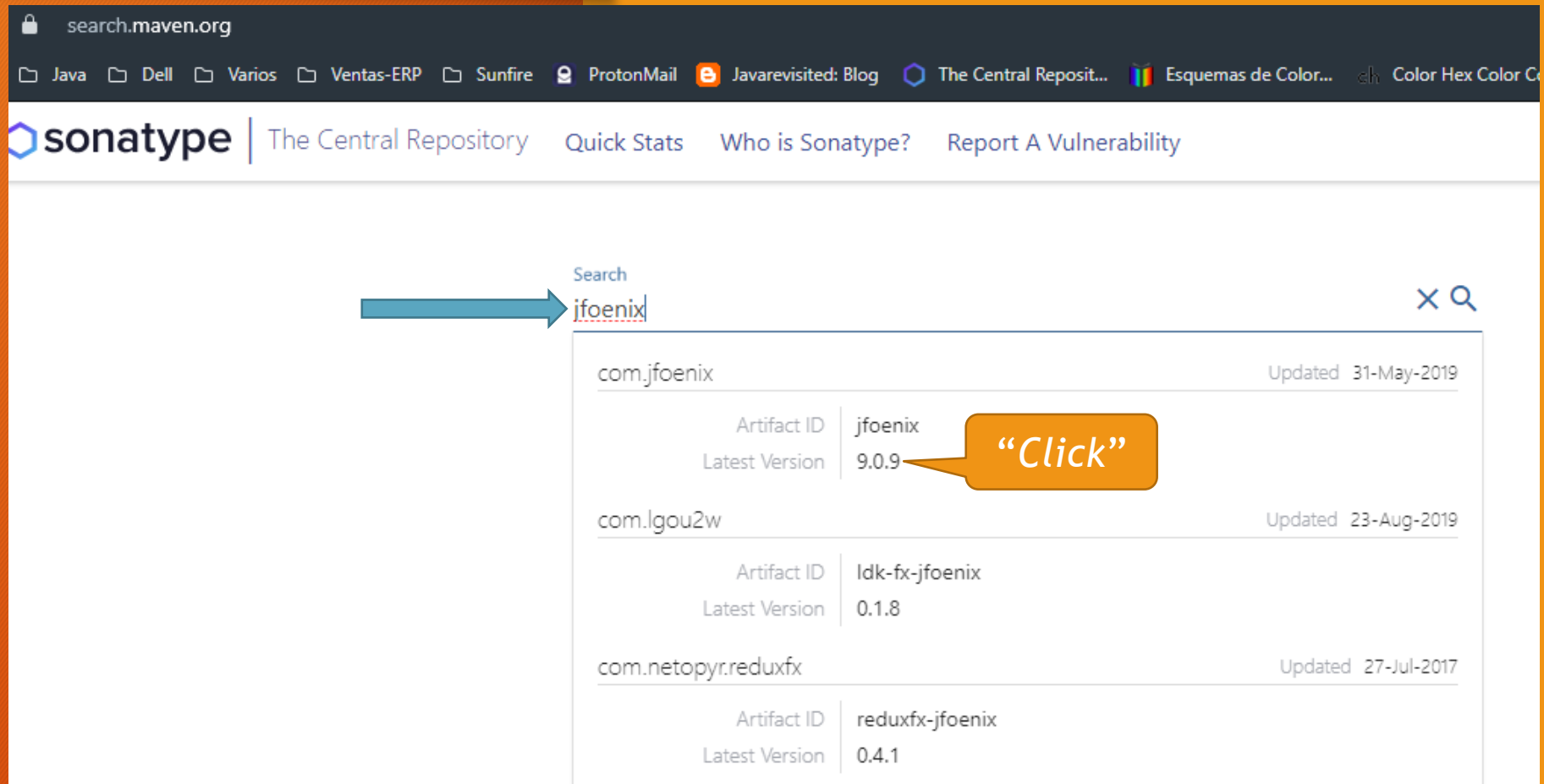
The screenshot shows a code editor with two tabs: 'maven/pom.xml' and 'visus-client/pom.xml'. The 'visus-client/pom.xml' tab is active, displaying XML code for a Maven POM file. The code includes several dependencies for Apache Commons, SLF4J, and Log4J. A new dependency for 'com.jfoenix:jfoenix' is being added at the bottom, with red arrows pointing to the new XML elements. The code is as follows:

```
61      <!-- Apache Commons -->  
62  
63      <dependency>  
64          <groupId>commons-lang</groupId>  
65          <artifactId>commons-lang</artifactId>  
66          <version>2.6</version>  
67      </dependency>  
68  
69      <!-- Logging -->  
70  
71      <dependency>  
72          <groupId>org.slf4j</groupId>  
73          <artifactId>slf4j-api</artifactId>  
74          <version>${slf4j.version}</version>  
75      </dependency>  
76      <dependency>  
77          <groupId>org.slf4j</groupId>  
78          <artifactId>jcl-over-slf4j</artifactId>  
79          <version>${slf4j.version}</version>  
80      </dependency>  
81      <dependency>  
82          <groupId>org.slf4j</groupId>  
83          <artifactId>slf4j-log4j12</artifactId>  
84          <version>${slf4j.version}</version>  
85      </dependency>  
86      <dependency>  
87          <groupId>log4j</groupId>  
88          <artifactId>log4j</artifactId>  
89          <version>${log4j.version}</version>  
90      </dependency>  
91  
92      <dependency>  
93          <groupId>com.jfoenix</groupId>  
94          <artifactId>jfoenix</artifactId>  
95          <version>8.0.9</version>  
96      </dependency>  
97  
98  
99  
100  </dependencies>  
101
```

Maven en acción

12. Integración con Eclipse

Las dependencias siempre se buscan en el repositorio de Maven Central
<https://search.maven.org>



The screenshot shows the Maven Central search interface. The browser address bar displays `search.maven.org`. The page header includes the Sonatype logo and navigation links: "The Central Repository", "Quick Stats", "Who is Sonatype?", and "Report A Vulnerability". A search bar at the top contains the text "jfoenix", with a blue arrow pointing to it from the left. To the right of the search bar is a magnifying glass icon. Below the search bar, three search results are displayed as cards:

- com.jfoenix** (Updated 31-May-2019)

Artifact ID	jfoenix
Latest Version	9.0.9

An orange callout bubble with the text "Click" points to the version number 9.0.9.
- com.lgou2w** (Updated 23-Aug-2019)

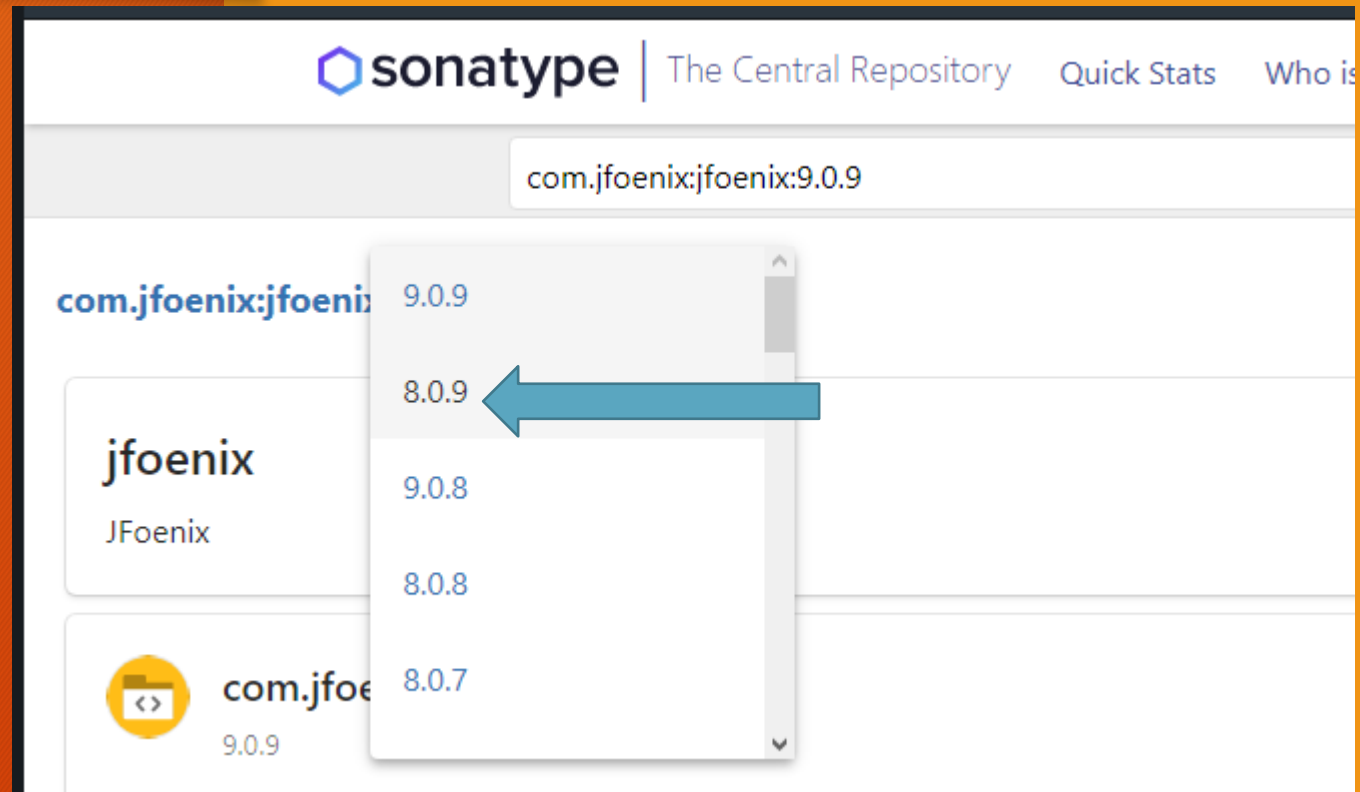
Artifact ID	ldk-fx-jfoenix
Latest Version	0.1.8
- com.netopyr.reduxfx** (Updated 27-Jul-2017)

Artifact ID	reduxfx-jfoenix
Latest Version	0.4.1

13. Integración con Eclipse

Maven en acción

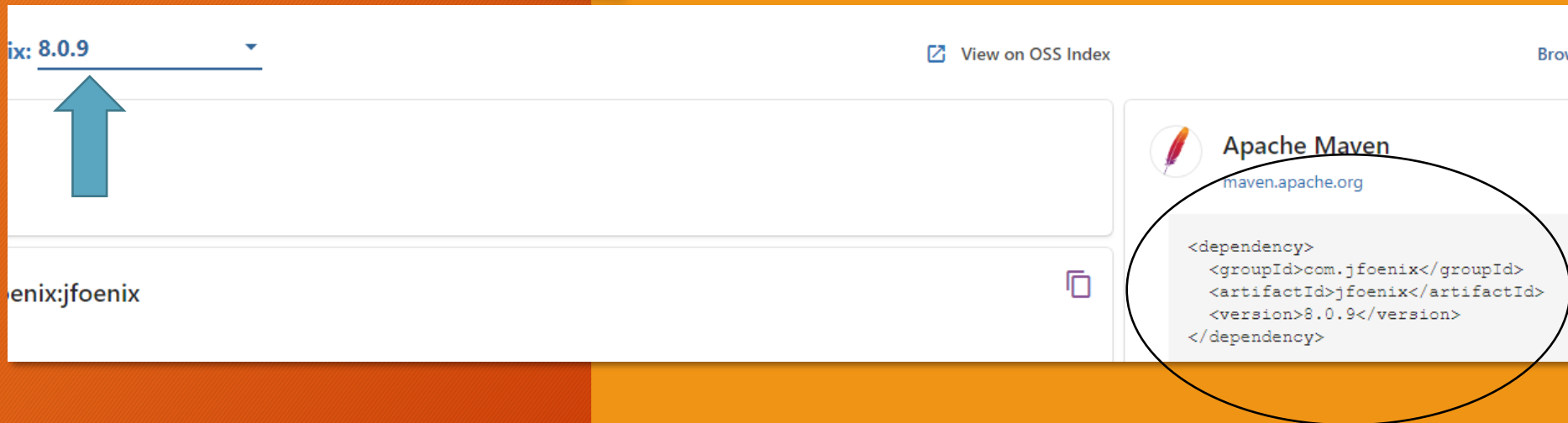
Abrimos el combo y seleccionamos la que nos interesa, la 8.09...



14. Integración con Eclipse

Maven en acción

Fíjense como ya tenemos a la derecha nuestro texto de dependencia listo para “copiar y pegar” en nuestro “pom.xml”.



The screenshot shows the Apache Maven website. On the left, the version '8.0.9' is highlighted with a blue arrow. On the right, the 'Apache Maven' logo and the URL 'maven.apache.org' are visible. Below this, a code block contains the following XML snippet, which is circled in black:

```
<dependency>
  <groupId>com.jfoenix</groupId>
  <artifactId>jfoenix</artifactId>
  <version>8.0.9</version>
</dependency>
```

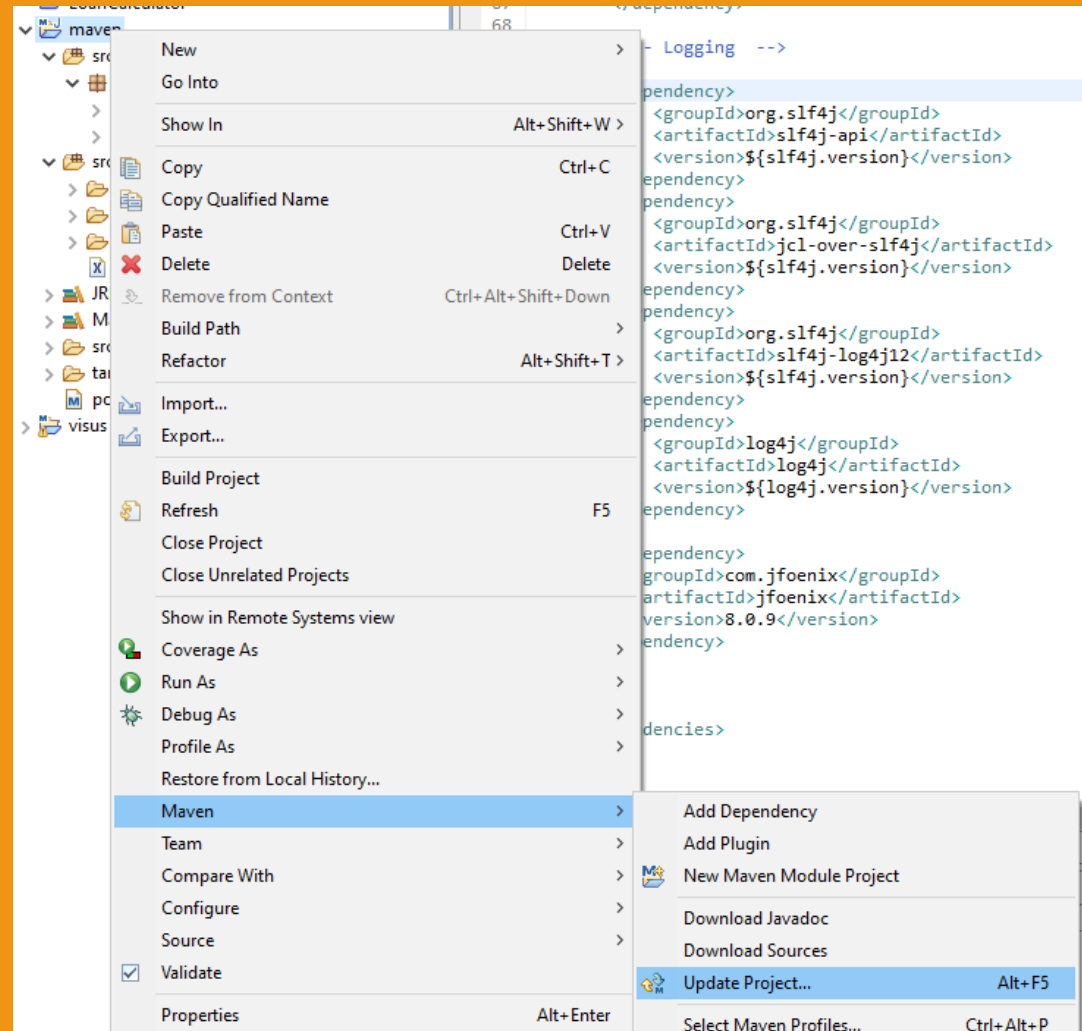
Así se hace con todas las dependencias que debamos usar.

Maven en acción

Actualizamos el proyecto Maven

15. Integración con Eclipse

Siempre que modifiquemos el archivo “*pom.xml*” debemos actualizar el proyecto Maven, para eso pulsamos <Alt + F5> o vamos al proyecto Maven y pulsamos botón derecho → Maven → Update Project...

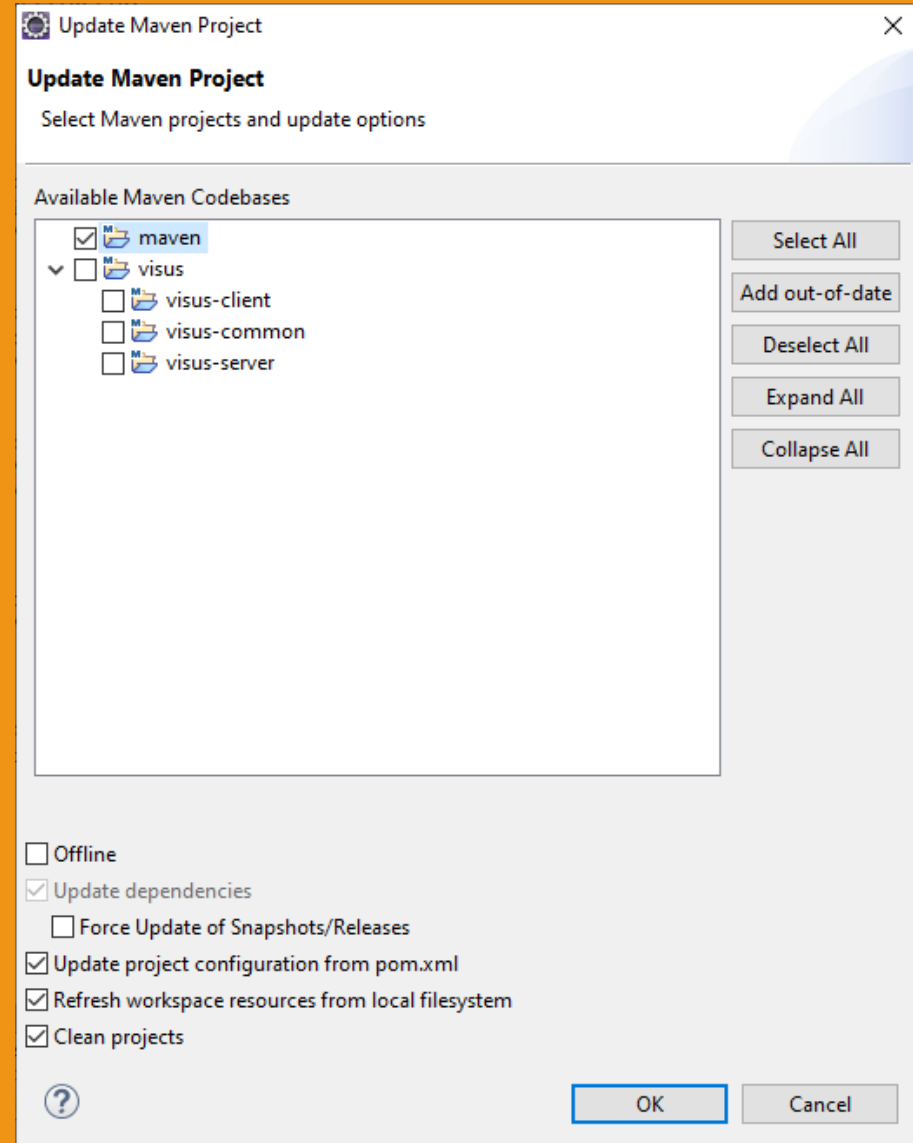


Maven en acción

16. Integración con Eclipse

Las otras opciones se dejan como están y damos “Ok”

Marcamos con un “check” nuestro proyecto:



17. Integración con Eclipse

Maven build → Compila el código del proyecto

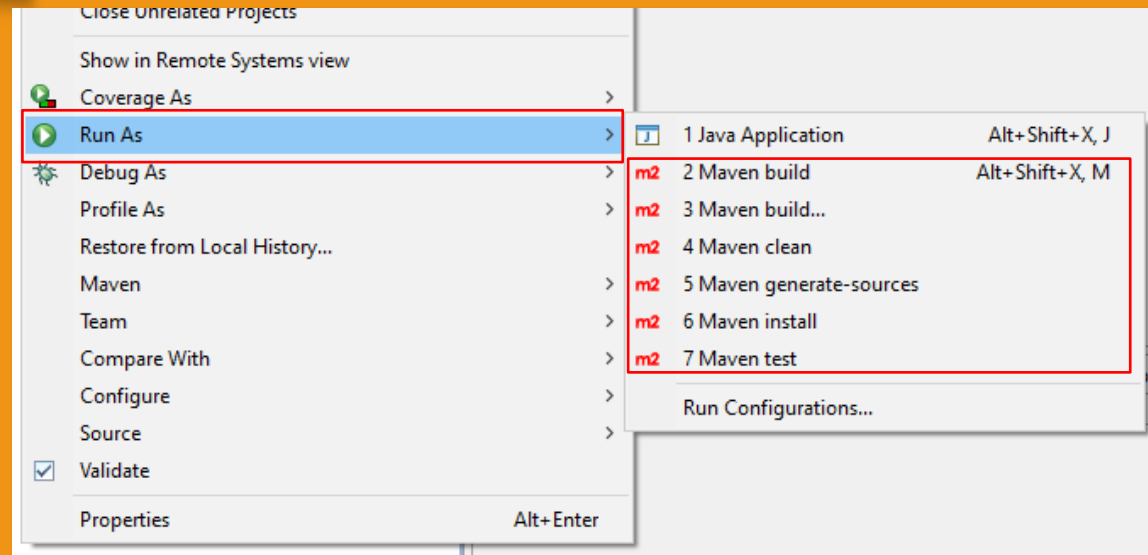
Maven clean → Elimina todos los archivos hechos por los *builds* anteriores

Maven generate-sources → Genera código para incluirlo en la compilación

Maven install → Instala los paquetes de la biblioteca en un repositorio local, compila el proyecto y lo comprueba.

Maven en acción

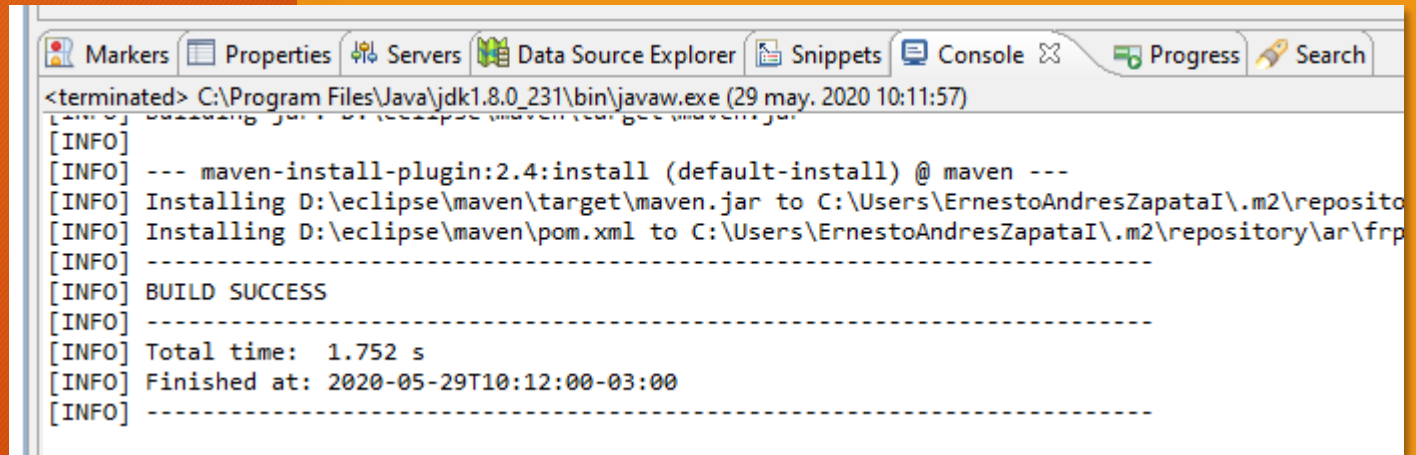
Vamos a correr por primera vez nuestro proyecto Maven y aprovecharemos para ver los ciclos de vida (*goals*) posibles que tiene.



Maven en acción

18. Integración con Eclipse

Elegiremos la opción *Maven install*. Aparece la consola con el proceso actual de dicha acción.



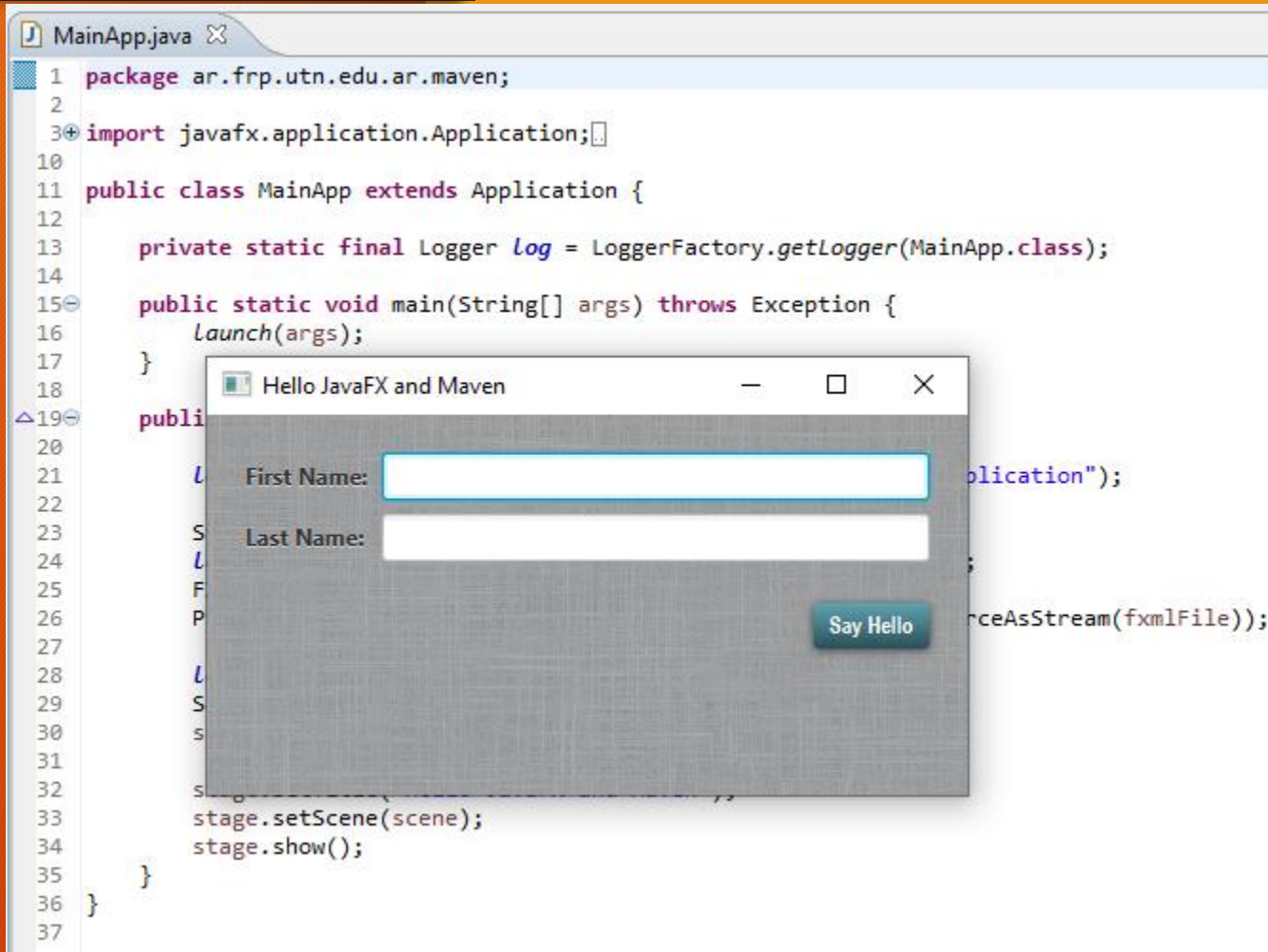
```
<terminated> C:\Program Files\Java\jdk1.8.0_231\bin\javaw.exe (29 may. 2020 10:11:57)
[INFO] Building jar: D:\eclipse\maven\target\maven.jar
[INFO] --- maven-install-plugin:2.4:install (default-install) @ maven ---
[INFO] Installing D:\eclipse\maven\target\maven.jar to C:\Users\ErnestoAndresZapataI\.m2\repository\org\apache\maven\maven-install-plugin\2.4\maven-install-plugin-2.4.jar
[INFO] Installing D:\eclipse\maven\pom.xml to C:\Users\ErnestoAndresZapataI\.m2\repository\ar\frp\pom.xml
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 1.752 s
[INFO] Finished at: 2020-05-29T10:12:00-03:00
[INFO] -----
```

Si vemos un mensaje de **BUILD SUCCESS** como en la imagen de arriba significa que el proceso se ha completado con éxito.

Maven en acción

19. Integración con Eclipse

Una vez finalizado corremos el programa como ya sabemos hacerlo...si todo está bien, debiéramos ver



¿Preguntas?