De normal, cuando realizamos un **proyecto Java** si es pequeño nosotros mismos podemos hacer las actividades típicas de la construcción de software.

* Instalar las librerías necesarias para nuestro proyecto en el classpath
* Correr los casos de test
* Crear la documentación del código fuente
* Compilar código fuente
* Empacar el código compilado en JAR

Pero, ¿qué sucede conforme nuestro proyecto va creciendo y se va agregando más gente a él? Pues que podemos perder el control del mismo y es más plausible cometer errores humanos en la construcción del programa. Para eso está **Maven**.

**¿Qué es Maven?**

**Maven**es una herramienta para la gestión de proyectos, que nos permitirá administrar completamente el ciclo de vida de la misma. Comúnmente se conoce a **Maven**como un gestor de dependencias Y YA. Y eso no es del todo cierto. Podemos considerarlo como una herramienta de construcción con muchas características, que nos ayuda a gestionar las dependencias, test, documentación, compilaciones, distribuciones e incluso los *mailing list*.

Si quieres una explicación más profesional de lo que es **Maven**puedes seguir el enlace hacía [Maven en Wikipedia](http://es.wikipedia.org/wiki/Maven).

Bien, centrémonos. ¿Qué es, entonces, lo que **Maven**puede hacer por nosotros? Así, en muy resumidas cuentas, lo siguiente:

* Build. Construye nuestro código fuente del proyecto
* Test. Ejecuta los casos de prueba
* Gestiona las dependencias de nuestro proyecto. Es cierto que no es sólo un gestor, pero para mi es de lo mejor que tiene **Maven**. Importa las librerías automáticamente desde un repositorio remoto. Nos podemos olvidar de hacer esa tarea manualmente.
* Permite la creación y descarga de plantillas de proyectos para tener la estructura ya creada. Por ejemplo si nos vamos a dedicar a hacer páginas web, o conectores, nos interesará para ahorrarnos el trabajo de tener que empezar desde cero una y otra vez
* Crea una web con la documentación del proyecto
* Desplega el proyecto (AKA **artefacto**) en servidor
* Y por supuesto, perfecta integración con **Git**, **SVN**, **Jira**...

Imagina hacer todo esto manualmente, y más de una persona que trabaje en nuestro proyecto. ¡Es para volverse loco!

Un par de cosas que conviene conocer antes de empezar:

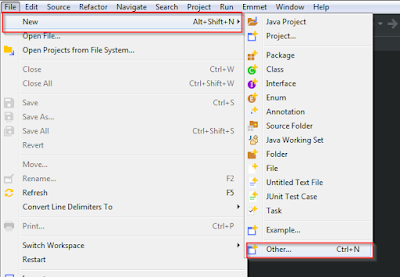
* **artefacto**. Es un proyecto que lo gestiona **Maven**y que incluye un fichero llamado **pom.xml**.
* **POM**. Son las siglas de **Project Object Model**. Es un fichero **XML**que contiene la configuración del artefacto. Más adelante trabajaremos con él. Si has programado en **Android**anteriormente, un símil adecuado sería el **AndroidManifest**.
* **groupId**. El identificador único para crear nuestro artefacto. Se suele poner el mismo que en un paquete java. *com.programandoapasitos*

**Maven en Eclipse**

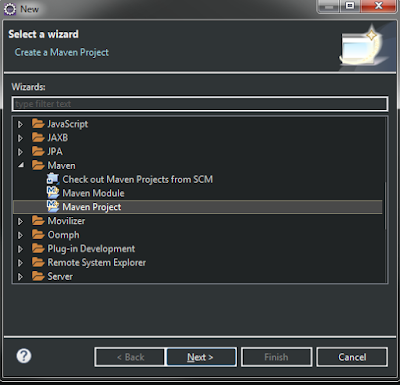
Tenéis que saber que Maven es perfectamente posible usarlo desde la consola de comandos. Si es el caso que estáis interesados en esa forma os recomiendo [éste artículo de Jarroba.com](https://jarroba.com/maven/) que lo explica perfectamente.

Pero en nuestro caso vamos a usar la comodidad que nos da Eclipse con las herramientas que ya vienen integradas por defecto en su versión [Java EE Developers](https://www.eclipse.org/downloads/) que es la que incluye el plugin de **Maven**, aunque ojo que puede no ser la última versión de **Maven**disponible.

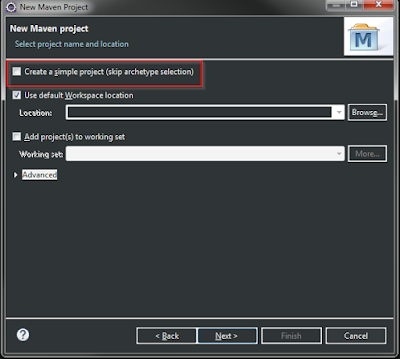
Lo primero será crear un nuevo proyecto **Maven**yendo a *File → New → Other*.

[](https://1.bp.blogspot.com/-G3WKomFik-I/WXrn7xmoV2I/AAAAAAAAIDQ/uC1tUTT5eqwmlmLuiByXbgqTM951zHotgCLcBGAs/s1600/1.png)

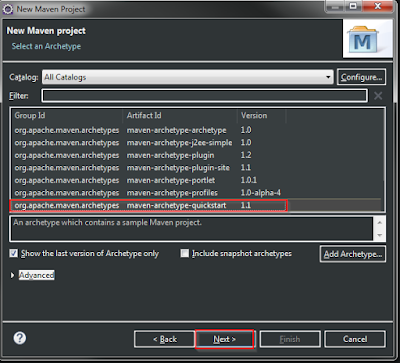
Elegimos ***Maven****→****Maven Project***y pulsamos Next.

[](https://1.bp.blogspot.com/-8kkwSjCJS7A/WXroaWlL2iI/AAAAAAAAIDU/DFFOq2RPhXkwSKGwgMYZHWprZQkTdyaXwCLcBGAs/s1600/2.png)

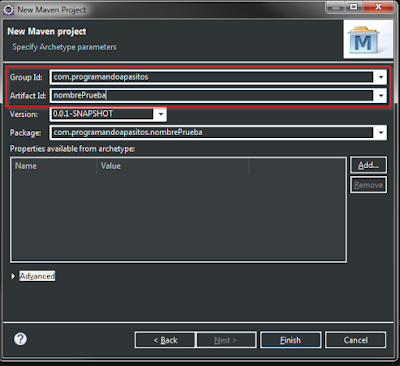
Ahora deberemos elegir la configuración de nuestro proyecto Maven. En la siguiente ventana podemos marcar el check de *Create a simple Project (skip archetype selection)*, lo que nos creará un proyecto completamente simple sin más configuración, o bien podemos desmarcarla, que será nuestro caso, y elegiremos un arquetipo de las opciones que nos muestre el asistente.

[](https://4.bp.blogspot.com/-EZDDhSkjKQI/WXrqQ16mIJI/AAAAAAAAIDg/dE2DYI7UXgcUkaqVtji2ivwT8z9AvuOwACLcBGAs/s1600/3.png)

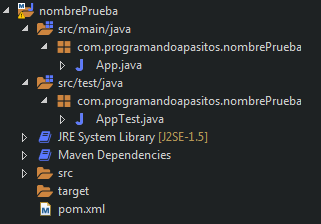
En la nueva ventana puedes seleccionar un arquetipo del listado o cargar uno desde una ubicación. En nuestro caso seleccionaremos el de *groupId* *org.apache.maven.archetypes* y **Artifact**id *maven-archetype-quickstart*y pulsamos Next.

[](https://3.bp.blogspot.com/-eyuu0QFnS28/WXrzqShEo3I/AAAAAAAAIDw/8yQgU0hQY6cjHSJ761JFWvS4NRtkL7fuwCLcBGAs/s1600/4.png)

Y ahora, por último paso del asistente, tendremos que indicar nuestro propio **groupId**y el **artifactId**. Esto se hace porque los que vimos en la anterior pantalla eran los que usaron los creadores del artefacto. Sobrescribiéndolos lo establecemos como queremos y aparte de la personalización, si algún día se convierte en arquetipo lo podrán buscar con nuestros datos, y reemplazarlos con los suyos.

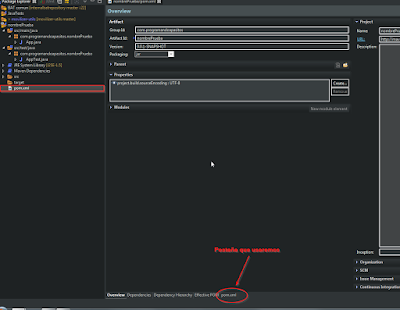
[](https://2.bp.blogspot.com/-yG1FtEtJWh0/WXr1xRIUPsI/AAAAAAAAID8/n7m4QYG6UOYv1IYsuyhYe4g88kkY8rEAQCLcBGAs/s1600/5.png)

Pulsamos *Finish*y ya tendremos nuestro proyecto. La estructura quedará tal que así

[](https://2.bp.blogspot.com/-iXzngN3bluU/WXr3JtMQSgI/AAAAAAAAIEE/VL1AKpLQNu8zLA5Q9oO_a2yIPswtsX48ACLcBGAs/s1600/6.png)

**Usando Maven**

Ahora que ya tenemos nuestro **proyecto Maven** creado es hora de configurar el **POM** para hacerlo funcionar. Para ello hacemos doble click en el archivo **pom.xml** y en la pestaña inferior elegimos la sección **pom.xml**para poder ver el código **XML** "a pelo".

[](https://3.bp.blogspot.com/-4F0Yv_UWz8M/WXr4jTPEKsI/AAAAAAAAIEM/euuviHCiMIA2oo29IL2Vh1arsdGpvme-wCLcBGAs/s1600/7.png)

Cuando entremos veremos un **archivo XML** similar a este

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.programandoapasitos</groupId>

<artifactId>nombrePrueba</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>nombrePrueba</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

De este código hay trozos que podemos intuir bastante bien con lo mencionado anteriormente. Del resto, el tag de **properties** indicará las propiedades de nuestro proyecto. Por ejemplo la codificación de los ficheros en **UTF-8**como vemos en esa línea.

Y luego tenemos **dependencies**, que es la sección donde gestionaremos... ¿adivinas qué? Exacto, nuestras dependencias. Pero eso lo retomaremos más adelante.

De momento lo que tenemos que conseguir es que **Maven** se encargue por nosotros de gestionar el JRE. Esto se hace incluyendo un nuevo plugin en este archivo. Así que nos ponemos debajo de **<dependencies>** y escribimos lo siguiente:

<build>

<pluginManagement>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</pluginManagement>

</build>

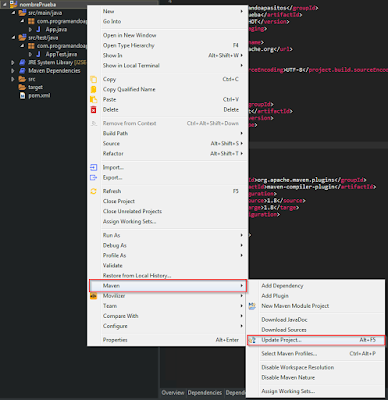
¿Qué hemos hecho?

Dentro de **build** hemos creado la etiqueta **pluginManagement** para gestionar los **plugins** de **Maven**. Y dentro de eso hemos creado una llamada **plugins** que será la que los contendrá.

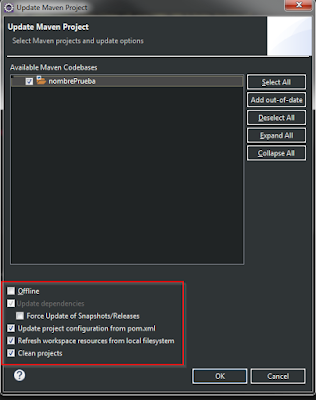
Después hemos procedido a añadir un nuevo **plugin**, en este caso **maven-compiler-plugin**. Y luego le hemos indicado, en su configuración, que el **source** y el **target** apunten a la versión que tengamos del **JRE** en nuestro dispositivo. En mi caso, el 1.8.

Ahora guardamos el fichero y no te preocupes si ves un error en el explorador de proyectos, en el siguiente paso se solucionará.

Vamos al proyecto **Maven** y pulsamos **botón derecho → Maven → Update Project...**

[](https://4.bp.blogspot.com/-tIJcw97QhfQ/WXr8bE4X91I/AAAAAAAAIEU/YmbGKCsYmP8c_EMkSunidDMmqYAR5K0IQCLcBGAs/s1600/8.png)

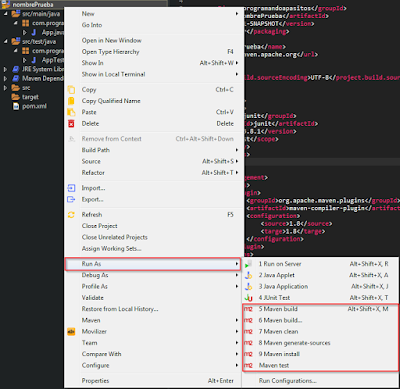
En la ventana que se abrirá marcamos con un check nuestro proyecto dejando la siguiente configuración

[](https://2.bp.blogspot.com/-EFg2t9ktjBY/WXr9fY_fRAI/AAAAAAAAIEc/FbMaZMZF5fgLobt8QNG90KIR8BPtfHjGACLcBGAs/s1600/9.png)

Cuando acabe el proceso de actualización veremos como desaparece el error de nuestro proyecto.

Hecho esto, vamos a lanzar por primera vez nuestro proyecto **Maven** y aprovecharemos para ver los ciclos de vida posibles que tiene.

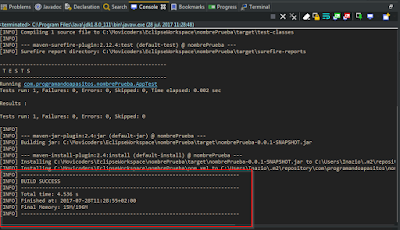
Hacemos clic con el botón derecho sobre nuestro proyecto y vamos a **Run As**. Ahí nos aparecerán diversas opciones

[](https://4.bp.blogspot.com/-qk0B34xZuW0/WXsASl7B7lI/AAAAAAAAIEs/_QhcW8W8-XQNcvbG3O6ihp6J7VsJXlOkACLcBGAs/s1600/11.png)

¿Qué hace cada una de ellas?

* **Maven build** → Compila el código del proyecto
* **Maven clean** → Elimina todos los ficheros 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.

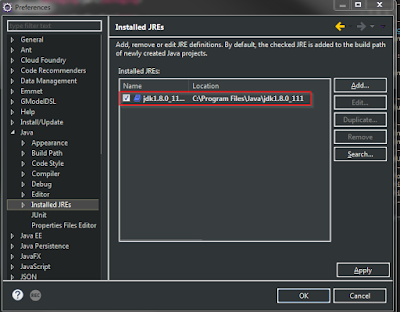
Para nuestra prueba, vamos a elegir la opción de **Maven install**.  
Se nos mostrará una ventana de consola con el proceso actual de dicha acción.

[](https://1.bp.blogspot.com/-I_g3hnMPQF0/WXsEE8pW5YI/AAAAAAAAIE0/aDNYXM-IPW8dsg1kGIBdFBkPyj2plCHbACLcBGAs/s1600/12.png)

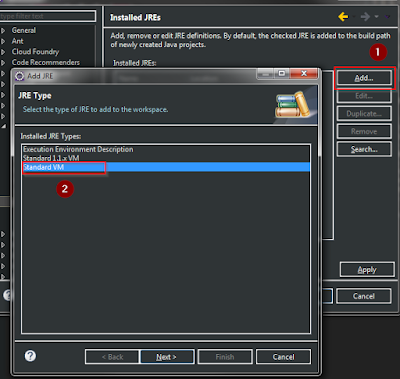
Si vemos un mensaje de **BUILD SUCCESS** como en la imagen de arriba significa que el proceso se ha completado con éxito.  
  
Es posible que, en vez de un mensaje de éxito, veamos lo siguiente:

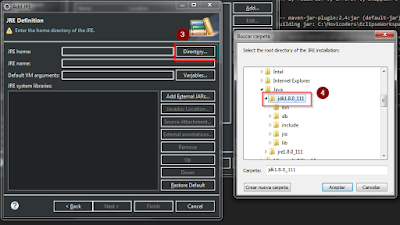
No compiler is provided in this environment. Perhaps you are running on a JRE rather than a JDK

Y nos esté dando un error. Esto quiere decir que tenemos mal configurada la ruta al **JDK** en el **workspace de Eclipse**.  
  
Para solucionarlo hay que ir a **Window → Preferences → Java → Installed JRE**. Lo que debe aparecer es la versión de **JDK**, como en esta imagen

[](https://3.bp.blogspot.com/-VuVLm2WgZg4/WXsFeg3M1TI/AAAAAAAAIFA/NV-_Z7tGNZ0z5tVQFPNCqrzkcXNA6TCUACLcBGAs/s1600/13.png)

Si en vez de eso tenemos el **JRE** es el motivo de que veamos ese error. Para subsanarlo pulsamos sobre **Add → Standard VM → Directory** y elegimos la carpeta de instalación de nuestro **JDK**

[](https://1.bp.blogspot.com/-q-VxxIus2G8/WXsGU3afXgI/AAAAAAAAIFI/T60ex5sTyqUXfLmSRDr6JELjLkgUtS8XQCLcBGAs/s1600/14.png)

[](https://2.bp.blogspot.com/-dRObXQ8_cwQ/WXsGU_voXII/AAAAAAAAIFE/ti-pQOuk1vEBCdf0RdgvvwasZhGD93DeQCLcBGAs/s1600/15.png)

Aplicamos los cambios, aceptamos y ya podemos repetir el proceso del **Maven install** habiendo corregido ese error.

**Dependencias**

Para gestionar las dependencias, dentro del **POM**deberemos fijarnos en el tag **dependencies**.

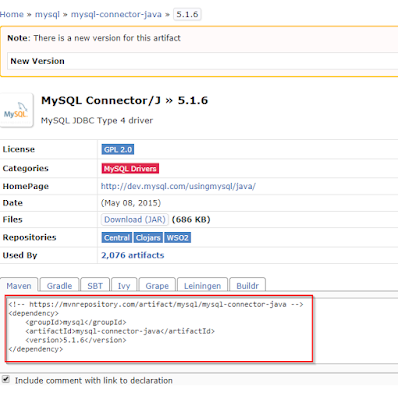
Por defecto tenemos cargada la librería de junit, que nos servirá para ver un ejemplo de cómo añadir más librerías.

Tenemos, al igual que con los plugin, los tag de groupId y artifactId. Aparte de eso, lo que nos interesa también es el tercer tag, version, que nos permitirá indicar la versión de la librería y, en caso de tener que modificarla, bastará con cambiar ese valor y volver a hacer un install.

Así pues... ¿de dónde podemos conseguir las dependencias?

Bien, hay varios sitios y Google lo sabe todo, pero así más concretamente hay una página web llamada [MVN Repository](http://mvnrepository.com/) que cuenta con un amplio abanico de dependencias organizadas por temas que es de lo mejorcito que podemos encontrar.

Simplemente entraremos en la dependencia que nos interese, por ejemplo el conector JDBC de MySQL, elegiremos la versión que queramos cargar en nuestro proyecto y en la parte inferior de la página nos aparecerá el código XML de Maven que deberemos copiar en el POM.

[](https://1.bp.blogspot.com/-6g7MVJZ3FNU/WXr_OGr9rMI/AAAAAAAAIEk/My18_JsHL0AxnXETNNxbtxNvcE2jb9MuwCLcBGAs/s1600/10.png)

De ese modo nuestro POM quedaría tal que así

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.programandoapasitos</groupId>

<artifactId>nombrePrueba</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>nombrePrueba</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

*<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->*

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.6</version>

</dependency>

</dependencies>

<build>

<pluginManagement>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</pluginManagement>

</build>

</project>

Y después de hacer un Maven install ya estaremos en disposición de usar nuestra nueva dependencia.