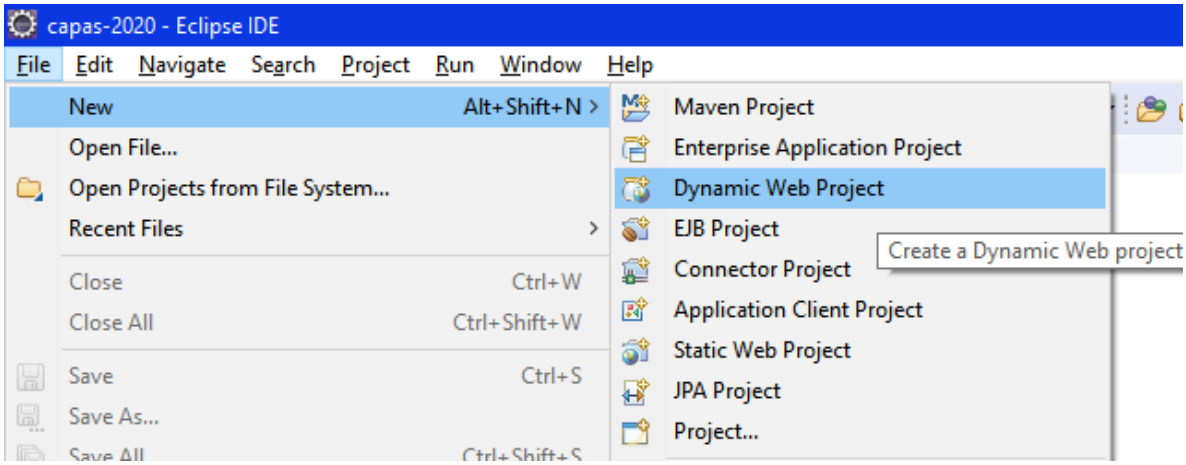
Creación de proyectos Maven con Eclipse

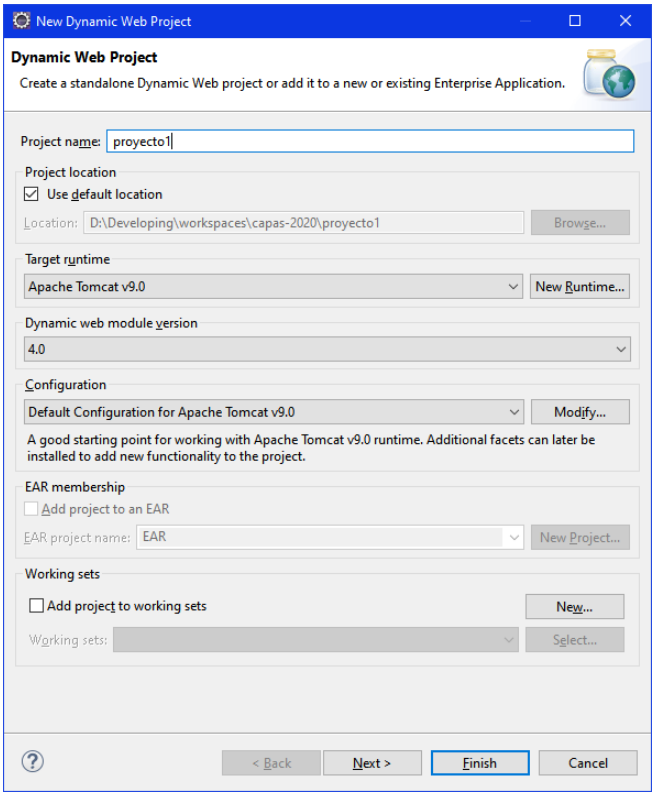
Los proyectos que estaremos creando con Eclipse son del tipo “Dynamic Web Project”.

Pasos para crear un nuevo proyecto web dinámico:

1. Ir a File -> New -> Dynamic Web Project:



2. Escribir un nombre al proyecto y dar clic a “Finish”.

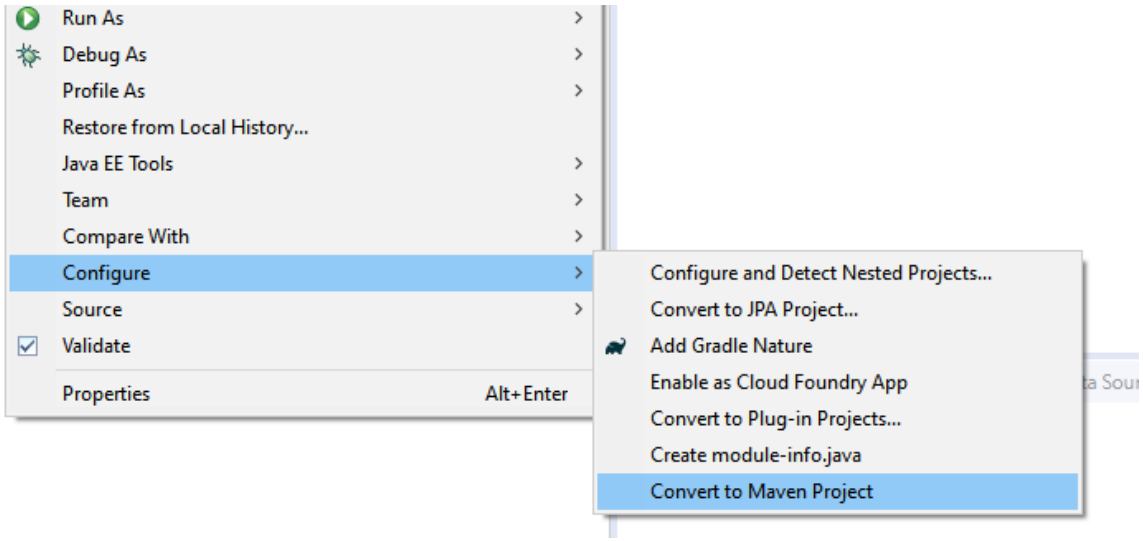


El proyecto ahora aparecerá en la pestaña Project Explorer.

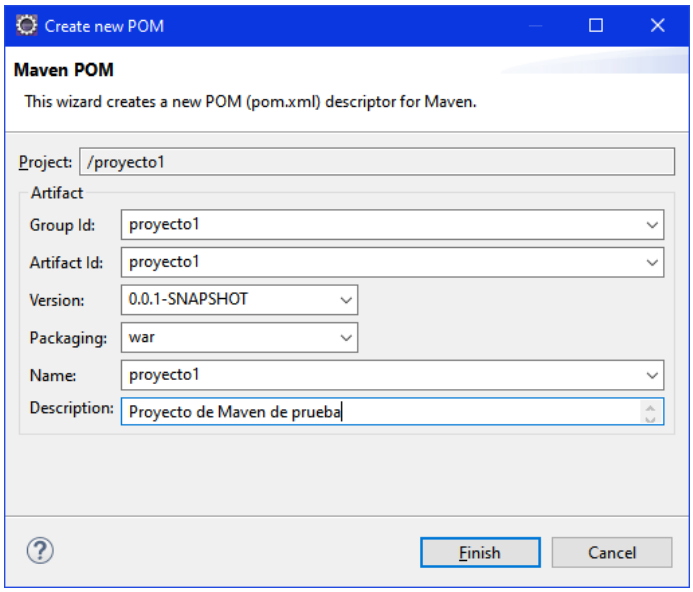
Ahora que ya tenemos creado el proyecto, tenemos que configurarlo para que tenga soporte de Apache Maven. Eclipse trae integrado Maven, por lo que la instalación mencionada en la diapositiva 11 es opcional.

Pasos para configurar el proyecto dinámico web con Maven.

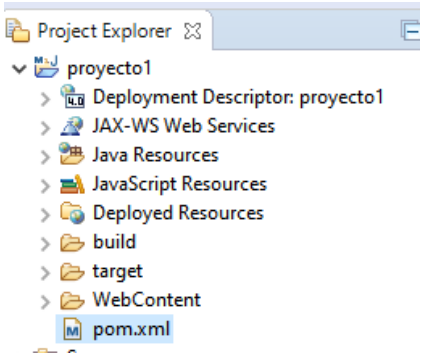
1. Hacer clic derecho al proyecto y seleccionar Configure -> Convert to Maven Project.



2. En la ventana siguiente, asegurarse que Packaging tenga “war”, los campos **Name** y **Description** son opcionales. Un archivo WAR es igual al archivo JAR que se genera cuando se compila un programa Java de escritorio, con la diferencia que el WAR es para utilizarlo en servidores como el Tomcat.



3. Dar clic a Finish, y veremos que se abrirá automáticamente el archivo pom.xml, el cual se encuentra en la raíz de nuestro proyecto.



4. Recordemos que utilizaremos Maven como gestor de dependencias, esto es, para gestionar automáticamente los archivos JAR de las librerías que estaremos utilizando de cada framework. Dichas dependencias van en el nodo **dependencies,** por lo que lo agregaremos a nuestro archivo pom.xml, este nodo irá justo después del cierre del nodo **build:**

```
      </plugin>
    </plugins>
  </build>

  <dependencies>

  </dependencies>

</project>
```

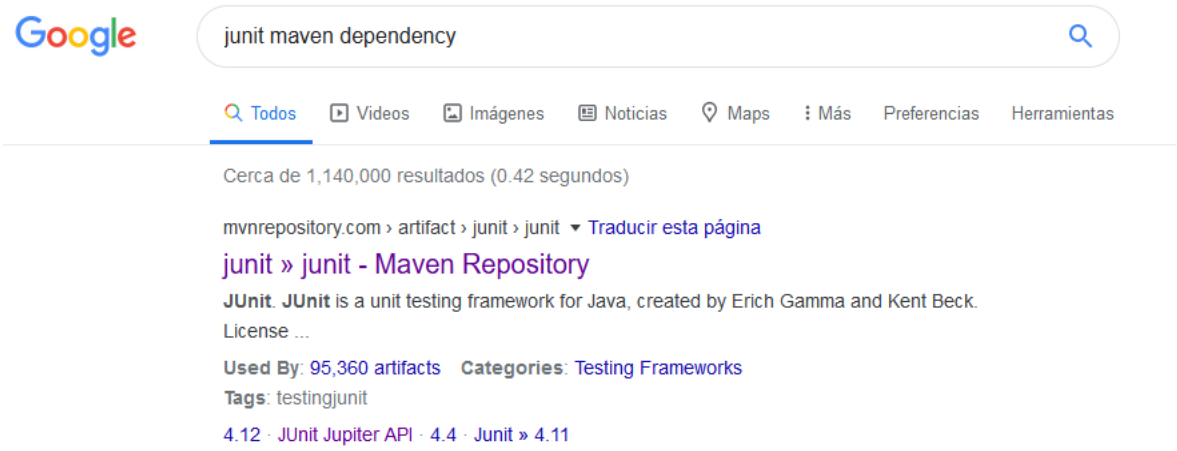
5. Ahora agregaremos nuestra primera dependencia, para esto agregaremos un nodo **dependency** y dentro de él, los nodos **groupId, artifactId y version.** En este ejemplo agregamos la librería de Junit (para pruebas unitarias)

```
<dependencies>
    <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api -->
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-api</artifactId>
        <version>5.6.0</version>
    </dependency>

</dependencies>
```

Ahora bien, ¿Cómo sabemos los datos que debemos poner en cada uno de los nodos **groupId, artifactId y version**? El sitio **mvnrepository.com** maneja toda esta información, esto quiere decir que aquí podemos buscar la dependencia, o desde Google, escribiendo “<nombre de la librería> Maven dependency”, por ejemplo “junit Maven dependency” o “hibernate Maven dependency” y los primeros resultados nos arrojarán al sitio mencionado directamente.

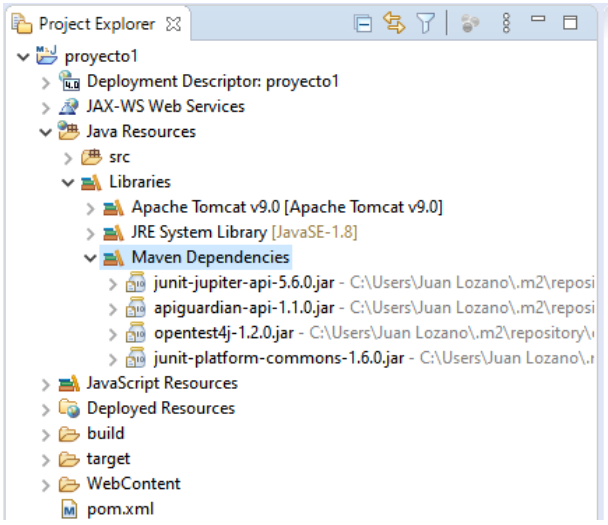
Por ejemplo, si buscamos junit Maven dependency en Google, nos arrojará el siguiente resultado:



Al entrar, vemos que lo han movido, por lo que procedemos a acceder al nuevo lugar de la librería.

Se va mostrar ahora un listado con las versiones que se manejan, para que elijamos la versión que mas nos convenga. Seleccionaremos la versión 5.6.0, y se abrirá una página que ya contiene el nodo **dependency** con toda la información necesaria listo para ser agregado a nuestro pom.xml.

Ahora, cuando agreguemos el nodo a nuestro archivo pom.xml y lo guardemos, Eclipse descargará los archivos JAR necesarios para la librería. Estos archivos los encontraremos en la siguiente sección de nuestro proyecto:



De esta manera hemos agregado las dependencias a nuestro proyecto de manera centralizada, por lo que cuando este proyecto sea abierto en otra computadora, Eclipse se encargará de descargar automáticamente los archivos necesarios.

Dichos archivos, media vez estén descargados, se guardan en un repositorio local. Este repositorio se encuentra en nuestra carpeta de usuario y se llama **.m2**, es escondido, por lo que hay que activar la opción de ver archivos y carpetas escondidas para poder verlo.

Lo que hace Maven es verificar que el archivo se encuentre primero en nuestro repositorio local, si lo encuentra entonces utiliza ese, de lo contrario lo descarga y lo guarda en una estructura de carpeta basándose en los datos del groupId, artifactId y versión. Por ejemplo, para la dependencia Junit que agregamos, el archivo se encontrará en:

C:\Users\<Usuario>\.m2\repository\org\junit\jupiter\junit-jupiter-api\5.6.0

Ya que el groupId es **org.junit.jupiter**, realiza un Split basándose en los puntos, y crea una carpeta con cada parte (org, junit, jupiter). Luego utiliza el artifactId (en este caso **junit-jupiter-api**) para crear otra carpeta y por último la versión para crear la carpeta final (ya que puede haber varias versiones de la misma librería).