¿Qué es Maven?

Maven es un herramienta que se encarga de gestionar y construir proyectos Java. En ocasiones entendemos que es un gestor de dependencias, pero es mucho más que eso, estas son sus funciones principales:

- Es un framework para gestionar proyectos.
- Es un repositorio de librerías.
- Maneja el concepto de dependencias del proyecto.
- Compila el proyecto, verifica con pruebas unitarias y lo empaqueta.
- Tiene una arquitectura basada en plugins.
- Automatiza el despliegue de proyectos web.

Obtener la última versión de Maven

Para descargar Maven sigue estos pasos:

- Accede a la dirección web: http://maven.apache.org/
- Haz clic en el vínculo “Download” de la izquierda.
- Descarga el archivo apache-maven-3.8.6-bin.zip.
- Descomprime el archivo descargado en la ubicación que desees.

Eclipse viene con una versión embebida de Maven, pero en lugar de usar esta versión, podemos hacer que utilice la última versión disponible que hemos descargado anteriormente. Para ello vamos a seguir estos pasos:

- Dentro de eclipse vamos a Windows / preferences.
- En las opciones de la izquierda seleccionamos Maven / Installations.
- Botón Add.

Crear proyecto nuevo basado en un arquetipo

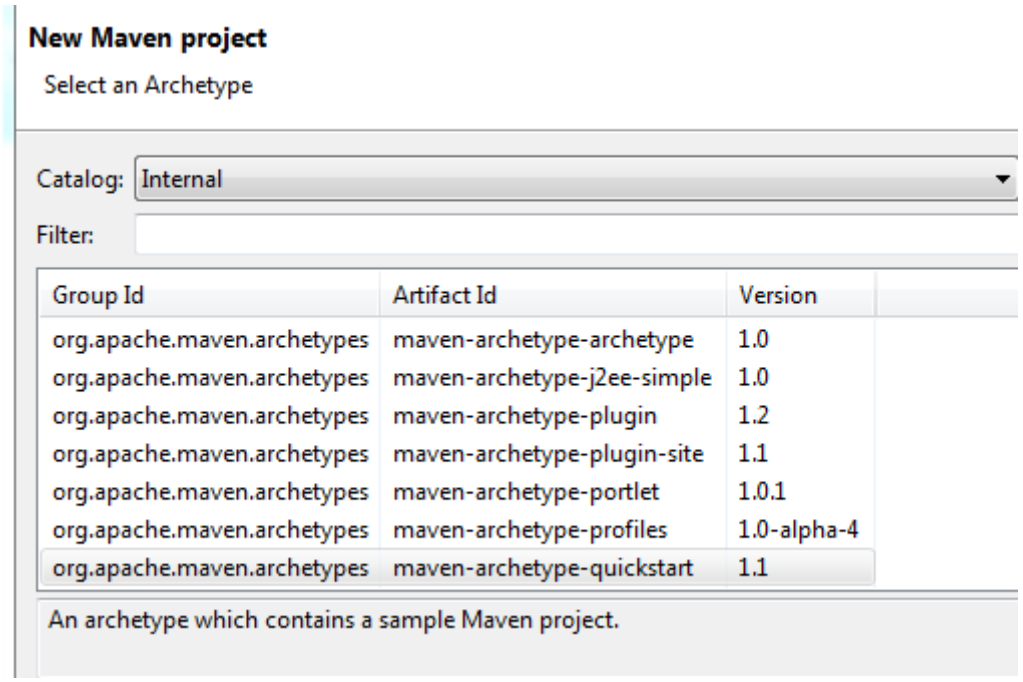
Por lo general, cuando creamos un proyecto nuevo Maven, lo creamos basándonos en un arquetipo.

- Los **arquetipos** son plantillas de proyectos.

- Los proyectos Maven se denominan **artefactos**.

Ya tenemos unos cuantos arquetipos, pero podemos agregar más. Vamos a crear un nuevo artefacto (proyecto) y añadiremos más arquetipos. Sigue estos pasos:

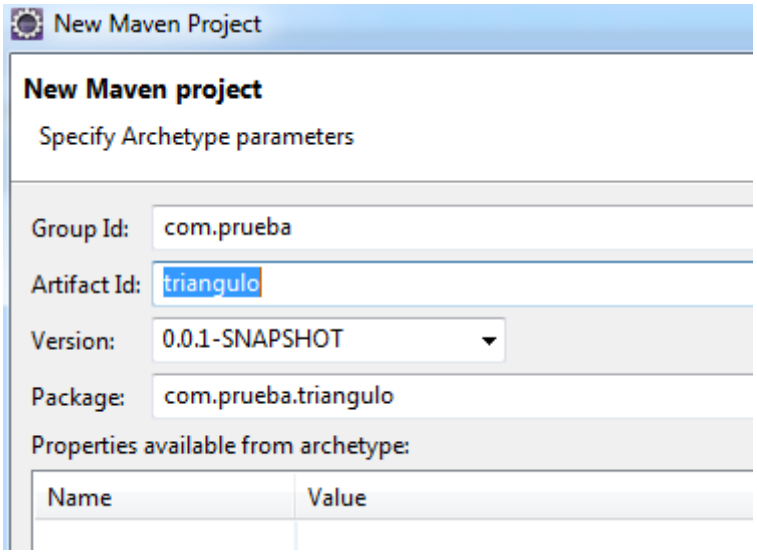
- File / New / Maven Project.
- Dejamos desmarcada la opción “Create a simple Project (skip archetype selection)” y hacemos clic en el botón “Next”.
- Ahora mismo estás viendo la lista desplegable de catálogos de arquetipos (Catalog) y debajo la lista de arquetipos. Selecciona el arquetipo maven-archetype-quickstart del catálogo Internal.



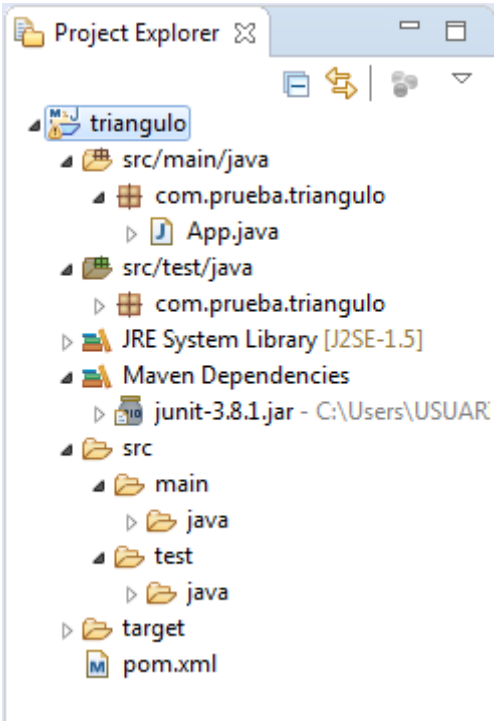
  También es posible crear nuevos catálogos locales o remotos. Puedes probar a agregar el siguiente catálogo remoto haciendo clic en el botón “Configure”:

  https://repo1.maven.org/maven2/archetype-catalog.xml

- Rellenamos los datos para el nuevo proyecto Maven y pulsamos Finish.

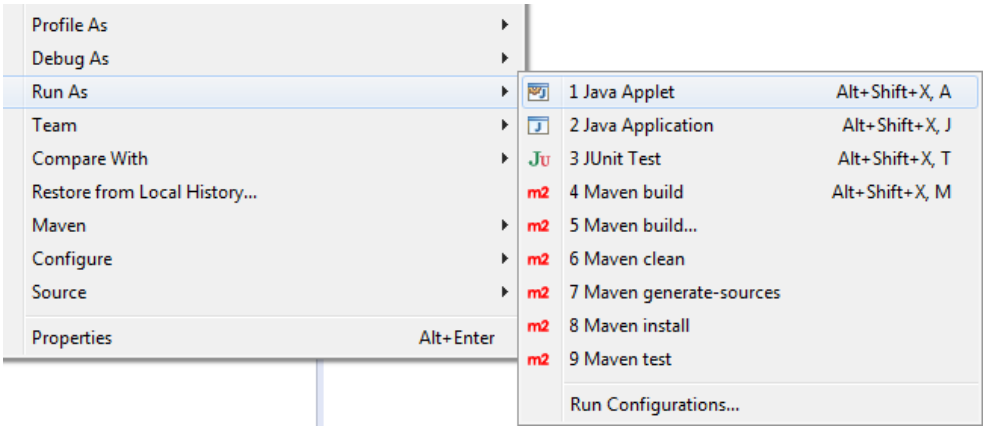


- Rellenamos los datos para el nuevo proyecto Maven y pulsamos Finish.
- Ahora tenemos un nuevo proyecto Java con la siguiente estructura:



Los comandos de Maven

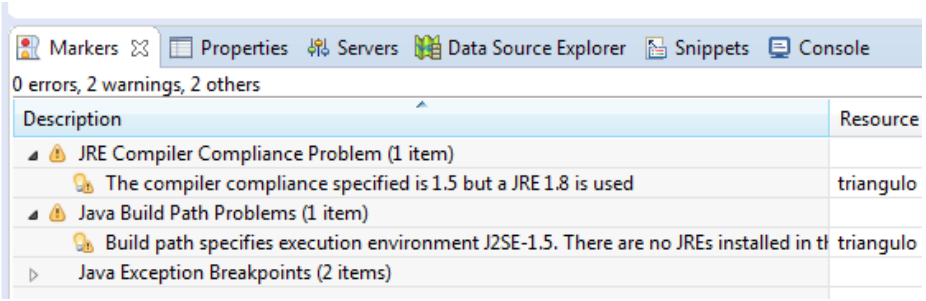
Haciendo clic derecho sobre el proyecto y seleccionando la opción “Run As”, comprobamos que aparecen varios comandos pre-configurados por el plugin de Maven en Eclipse.



Como una simple prueba, podemos ejecutar el comando Maven test.

Conflictos con la versión de Java

Si activamos la ficha “Markers” nos damos cuenta de que existen unos warnings relacionados con un conflicto con la versión de Java.



Podemos solucionar este conflicto dejando que Maven se encargue de la gestión del JRE introduciendo la siguiente entrada en el archivo POM.

```
<build>
	<pluginManagement>
		<plugins>
			<plugin>
				<groupId>org.apache.maven.plugins</groupId>
				<artifactId>maven-compiler-plugin</artifactId>
				<configuration>
					<source>16</source>
					<target>16</target>
				</configuration>
			</plugin>
		</plugins>
	</pluginManagement>
</build>
```

Este cambio nos ha provocado un error que se soluciona actualizando el proyecto de la siguiente manera:

- Hacemos clic derecho sobre el nombre del proyecto.
- Seleccionamos Maven / Update Project.

Comenzar a programar

Imaginar que queremos convertir un objeto POJO en un archivo JSON. Para lograr el objetivo, buscaríamos en la última versión de la herramienta Gson en el repositorio de maven (http://mvnrepository.com/) hasta localizar la dependencia necesaria que sería esta:

```
<!-- https://mvnrepository.com/artifact/com.google.code.gson/gson -->
<dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.8.9</version>
</dependency>
```

Copiamos esta dependencia en el archivo POM dentro de la etiqueta <dependencies>

Ahora vamos a hacer uso de la librería GSON.

Para comenzar creamos la siguiente clase:

```
package com.prueba.triangulo;

public class Triangulo {
	private int lado1;
	private int lado2;
	private int lado3;
	private String tipo;

	public Triangulo(int lado1, int lado2, int lado3) {
		super();
		this.lado1 = lado1;
		this.lado2 = lado2;
		this.lado3 = lado3;
		this.tipo = this.getTipo();
	}

	public int getLado1() {
		return lado1;
	}
	public void setLado1(int lado1) {
		this.lado1 = lado1;
		this.tipo = this.getTipo();
	}
	public int getLado2() {
		return lado2;
	}
```

```
	public void setLado2(int lado2) {
		this.lado2 = lado2;
		this.tipo = this.getTipo();
	}
	public int getLado3() {
		return lado3;
	}
	public void setLado3(int lado3) {
		this.lado3 = lado3;
		this.tipo = this.getTipo();
	}

	public String getTipo() {
		if (this.lado1==this.lado2 && this.lado2==this.lado3) {
			return"Equilátero";
		}
		else if (this.lado1 == this.lado2 || this.lado2==this.lado3 ||
this.lado1==this.lado3) {
			return"Isósceles";
		}
		else {
			return"Escaleno";
		}
	}

	@Override
	public String toString() {
		return"Triangulo [lado1=" + lado1 + ", lado2=" + lado2 + ",
lado3=" + lado3 + ", tipo=" + tipo + "]";
	}
}
```

Ahora podemos modificar la clase App que ya nos había suministrado Maven por el siguiente código:

```
package com.prueba.triangulo;

import java.util.ArrayList;

import com.google.gson.Gson;

public class App
{
public static void main( String[] args )
   {
       ArrayList<Triangulo> triangulos = new ArrayList<Triangulo>();
       triangulos.add(new Triangulo(1, 2, 3));
       triangulos.add(new Triangulo(2, 2, 3));
       triangulos.add(new Triangulo(2, 2, 2));
       triangulos.add(new Triangulo(1, 5, 3));
       triangulos.add(new Triangulo(1, 5, 5));
       triangulos.add(new Triangulo(2, 2, 2));

       Gson gson = new Gson();
       String formatoJSON = gson.toJson(triangulos);
       System.out.println(formatoJSON);

    }
```

```
}
```

¿Qué es el archivo POM?

POM son las iniciales de Proyect Object Model. Se trata de un fichero XML fundamental para los proyectos Maven. Contiene valores de configuración relevantes que determinan la forma en que se construye el proyecto de manera automática. Algunas de las configuraciones son las dependencias del proyecto, englobadas dentro de la etiqueta <dependencies>.

¿Qué es el SNAPSHOT?

El SNAPSHOT se refiere a la versión del proyecto que está desarrollándose.

0.0.1-SNAPSHOT: primera versión (SNAPSHOT en desarrollo)

0.0.2-SNAPSHOT: segunda versión.

0.0.3-SNAPSHOT: tercera versión.

Las dependencias

Dentro del archivo POM se especifican las dependencias necesarias por medio de entradas de tipo <dependency>. Vamos a ver algunos términos relacionados con las dependencias:

- **Dependencia transitoria:** se da cuando al incorporar una dependencia, se añaden automáticamente otras dependencias que dependen de la primera**.**
- **Alcance de las dependencias (Dependency Scope):** determina en qué ámbitos serán necesarias las dependencias. Estos son los valores más comunes:
    - Compile: dependencia necesaria para la compilación. Es el valor predeterminado.
    - Provided: se utiliza en aplicaciones que van a ser desplegadas en un servidor para indicar que dicha librería estará presente

en el servidor y no será necesario colocarla dentro de empaquetado jar, war o ear.
    - Runtime: dependencia necesaria en tiempo de ejecución que no es necesaria para compilar.
    - Test: dependencia sólo necesaria para las pruebas unitarias.

- **Tipos de dependencias: hace referencia al tipo de empaquetado** (packaging) de la dependencia. Por defecto se asume tipo JAR.
    - POM
    - EAR
    - WAR
    - EJB
    - JAR

El repositorio Maven

Un repositorio es un lugar o almacén donde se guardan cosas, en este caso, el repositorio sirve para guardar librerías, proyectos o arquetipos. Existen varios tipos de repositorios:

- **Repositorio local:** situado en nuestro equipo local. Por defecto está situado en .m2/repository. Para ver la ubicación del repositorio local dentro de eclipse, podemos seleccionar por medio del menú Windows / Preferences / Maven / User settings
- **Repositorio central:** es el repositorio central de Maven, ubicado en https://repo1.maven.org/maven2/.
- **Repositorio remoto:** es un directorio situado en un equipo remoto donde se almacenan librerías, proyectos, arquetipos, etc. A diferencia del repositorio central, los repositorios remotos no forman parte del proyecto maven, sino que son de terceros.
- **Plugin repository:** son similares a los repositorios de dependencias, pero administran plugins.

Diferencia entre SNAPSHOT y RELEASE

El SNAPSHOT hace referencia a la versión durante el tiempo de desarrollo. El número de SNAPSHOT es mutable.

El RELEASE hace referencia a la versión de un producto ya construido. El número RELEASE es inmutable, ya que una vez construida una versión, no es correcto cambiarla, en todo caso se crea otro nuevo proyecto.

Maven install

El comando Maven Install empaqueta la aplicación en el tipo de archivo escogido (jar, war, ear, etc).