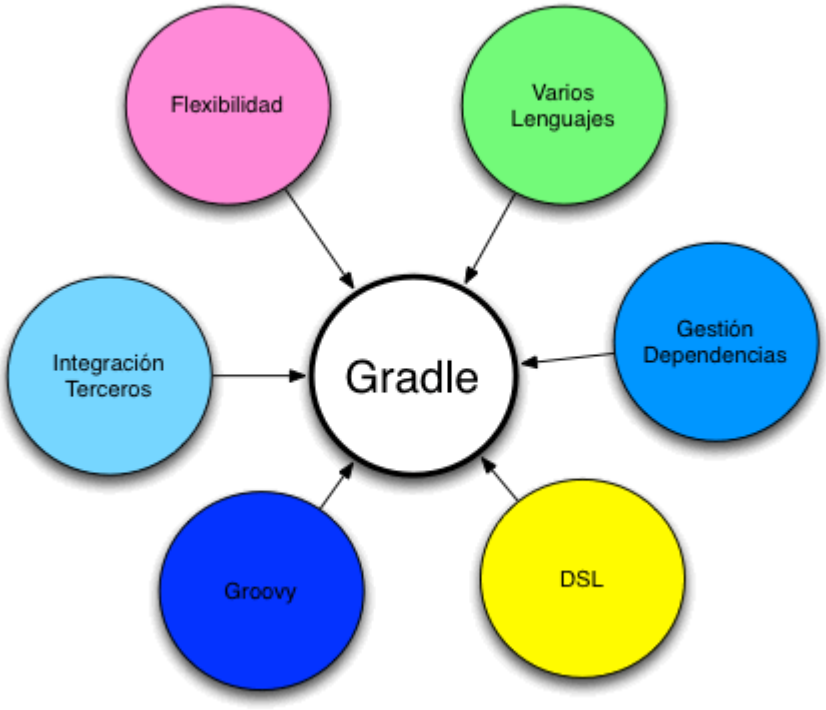
Poco a poco todo avanza en el mundo de la programación y uno de los campos que siempre necesita de mejoras continuas es el mundo de la automatización de builds (construcciones). Todos hemos manejado estos años ant y maven para automatizar la construcción de proyectos lo máximo posible. Poco a poco Gradle se esta haciendo hueco como nueva herramienta para gestionar la automatización de los builds.

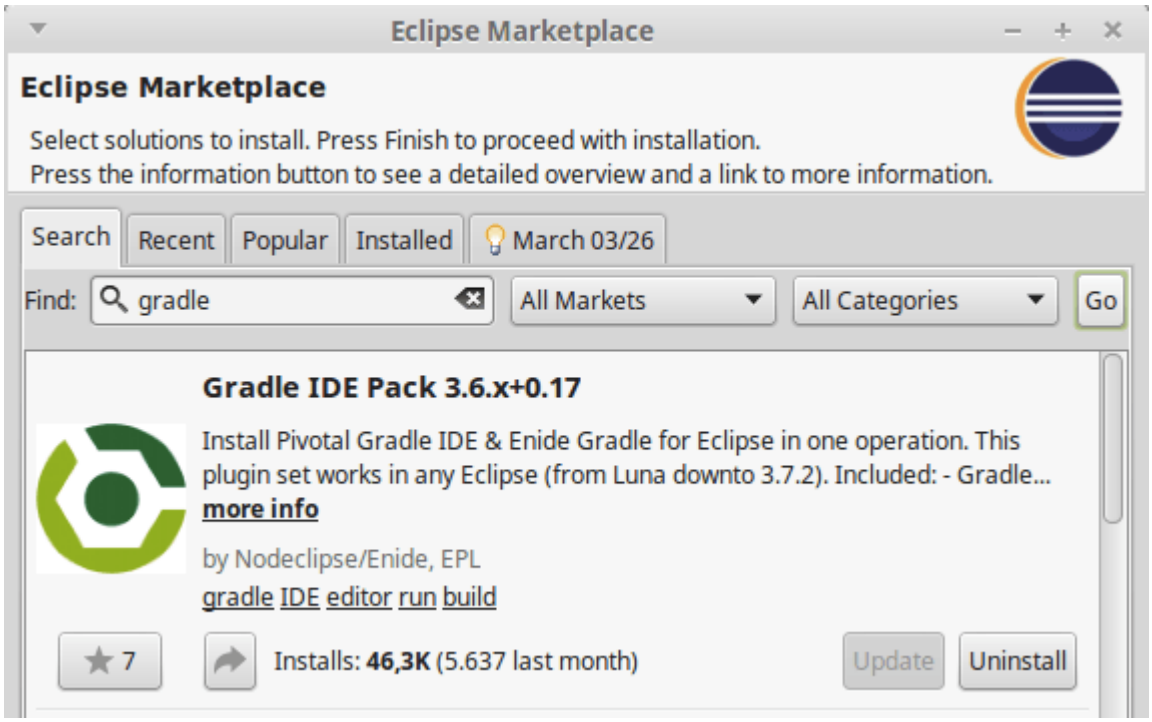
¿Qué es Gradle?

Gradle es una herramienta de automatización de la construcción de nuestro código que bebe de las aportaciones que han realizado herramientas como ant y maven pero intenta llevarlo todo un paso mas allá. Para empezar se apoya en Groovy y en un DSL (Domain Specific Language) para trabajar con un lenguaje sencillo y claro a la hora de construir el build comparado con Maven. Por otro lado dispone de una gran flexibilidad que permite trabajar con ella utilizando otros lenguajes y no solo Java. Dispone por otro lado de un sistema de gestión de dependencias sólido.

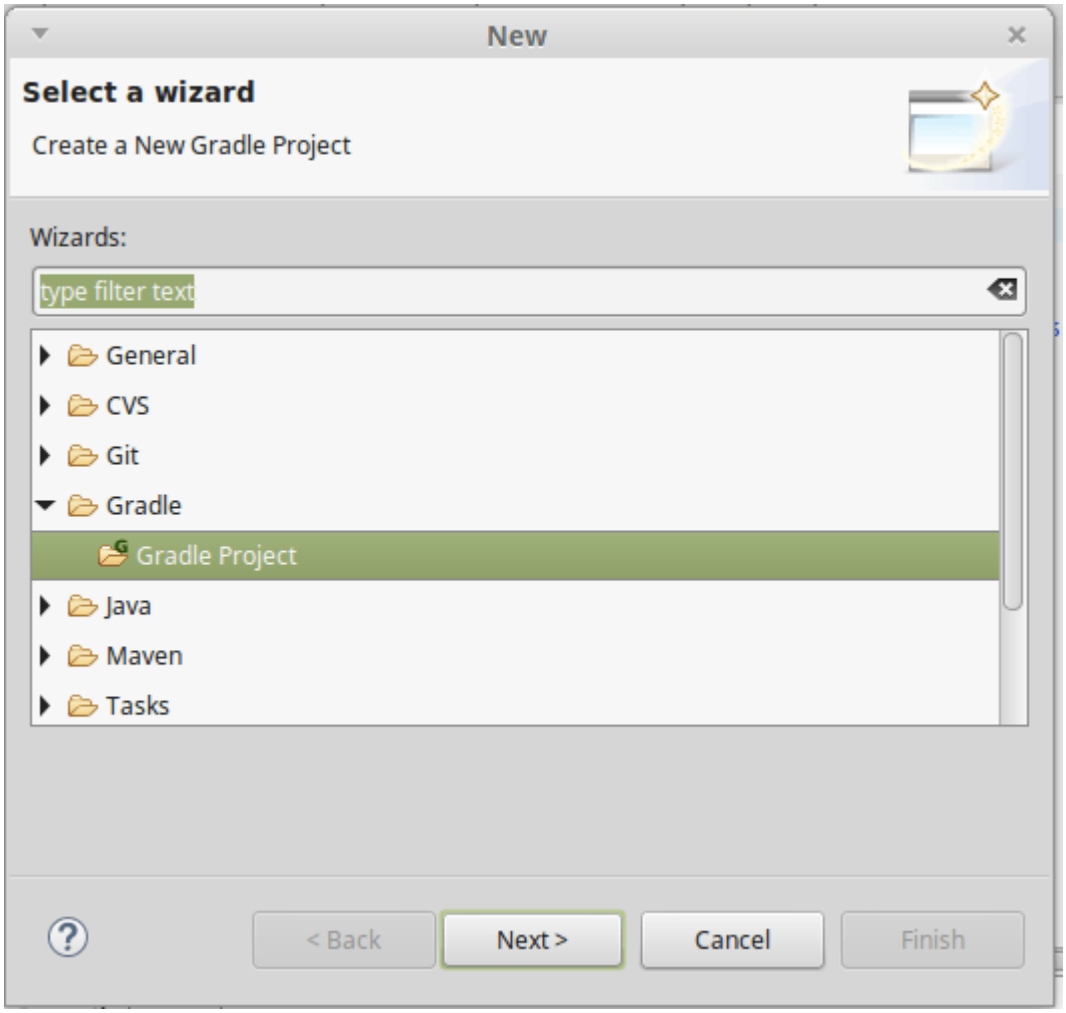


Vamos a construir un ejemplo sencillo de Gradle. Para ello el primer paso será instalarnos

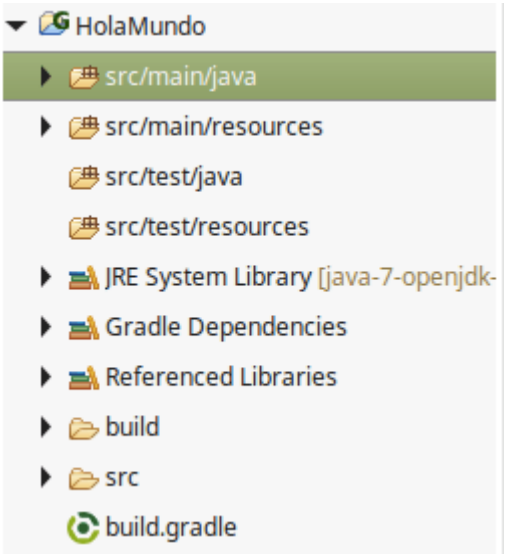
las herramientas de Gradle para eclipse a través del marketplace.



Realizada este primera operación podemos usar el entorno de eclipse para generarnos un proyecto de Gradle.



Seleccionamos el proyecto de QuickStart y se construye el típico proyecto de eclipse que contiene una estructura de carpetas tipo Maven.



Como podemos ver disponemos de un fichero build.gradle que es el encargado de gestionar el build y que mostramos su código a continuación.

Principal.java | build.gradle | log4j2.xml

```
apply plugin: 'java'
apply plugin: 'eclipse'
apply plugin: 'application'

sourceCompatibility = 1.5
mainClassName = "com.arquitecturajava.Principal"
version = '1.0'
jar {
    manifest {
        attributes 'Implementation-Title': 'HolaMundo',
                   'Implementation-Version': version
    }
}

repositories {
    mavenCentral()
}

dependencies {
    compile group: 'org.apache.logging.log4j', name: 'log4j-core', version: '2.2'

 }
 eclipseClasspath {

 }
```

Como podemos ver se trata de un DSL (Domain Specified Language) que es bastante compacto. En las primeras lineas registramos tres plugins que nos permitirán abordar diversas tareas.

Java Plugin: Se encarga de poner a nuestra disposición tareas del mundo java como por ejemplo la tarea “jar” de empaquetado que usamos en la linea 13.

Eclipse Plugin :Se encarga de integrar gradle dentro de eclipse y configurar un classpath basado en las dependencias que asignamos en la linea 24

Application Plugin: Se encarga de añadir las tareas de ejecución de tal forma que desde el propio gradle podamos ejecutar nuestro proyecto definiendo en la linea 11 la clase main.

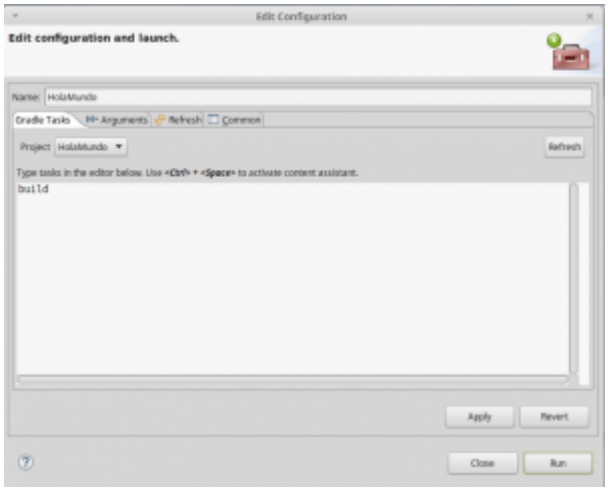
Una vez tenemos definido el fichero podemos ver el contenido de la clase Principal que deseamos ejecutar.

```
package com.arquitecturajava;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class Principal {
 private static final Logger logger = LogManager.getLogger();

 public static void main(String[] args) {
 logger.error("hola, mundo");
 }
}
```

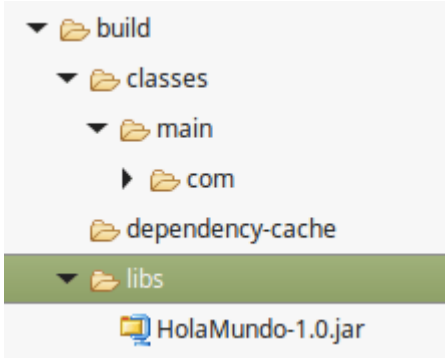
En este caso se trata de una clase que usa el API de log4j (versión2) para imprimir un mensaje de error. Para ello deberemos definir en la carpeta de resources el fichero log4j2.xml que nos configura el appender de consola.

```
&amp;amp;lt;?xml version="1.0" encoding="UTF-8"?&amp;amp;gt;
&amp;amp;lt;Configuration status="WARN"&amp;amp;gt;
&amp;amp;lt;Appenders&amp;amp;gt;
&amp;amp;lt;Console name="Console" target="SYSTEM_OUT"&amp;amp;gt;
&amp;amp;lt;PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] %-5level
%logger{36} - %msg%n" /&amp;amp;gt;
&amp;amp;lt;/Console&amp;amp;gt;
&amp;amp;lt;/Appenders&amp;amp;gt;
&amp;amp;lt;Loggers&amp;amp;gt;
&amp;amp;lt;Root level="error"&amp;amp;gt;
&amp;amp;lt;AppenderRef ref="Console" /&amp;amp;gt;
&amp;amp;lt;/Root&amp;amp;gt;
&amp;amp;lt;/Loggers&amp;amp;gt;
&amp;amp;lt;/Configuration&amp;amp;gt;
```

Realizados estos pasos ya tenemos un mini proyecto y podemos usar gradle y su tarea build para construir nuestro código y generar un jar.



Realizada esta operación gradle nos generará un jar con el proyecto preparado.



De igual forma podríamos haber ejecutado la tarea :run y que gradle nos ejecutara el proyecto.

Otros artículos relacionados :

Introducción a Maven

Maven y Dependencias

Eclipse y Maven