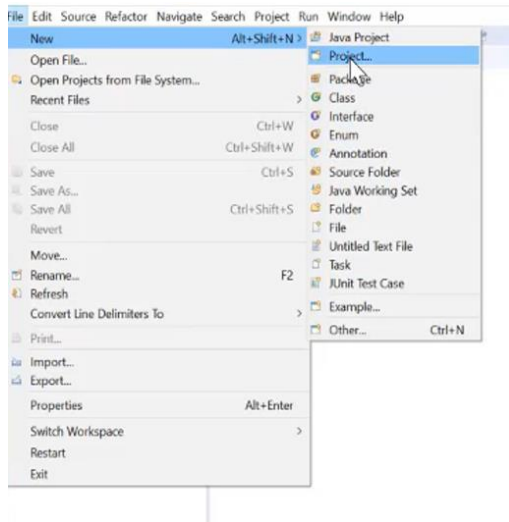


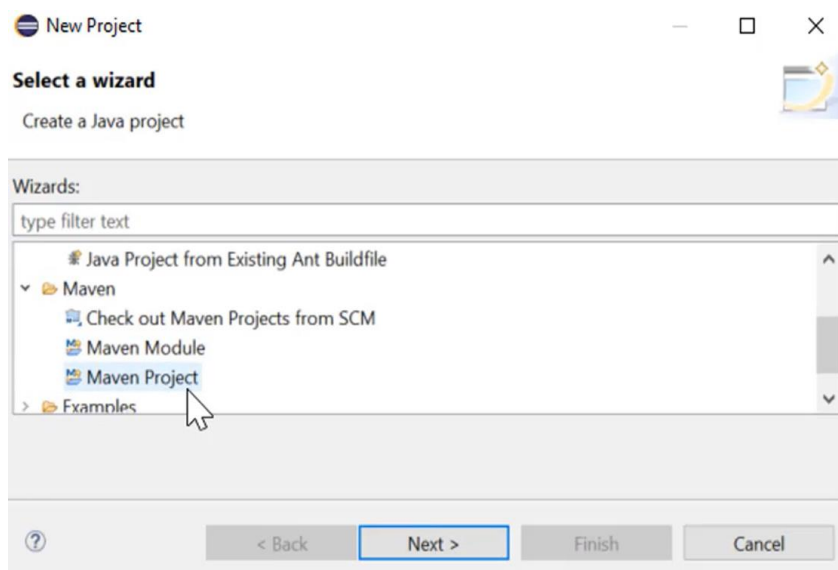
Creación de proyecto con java Maven en Eclipse

Creación de Proyecto Maven

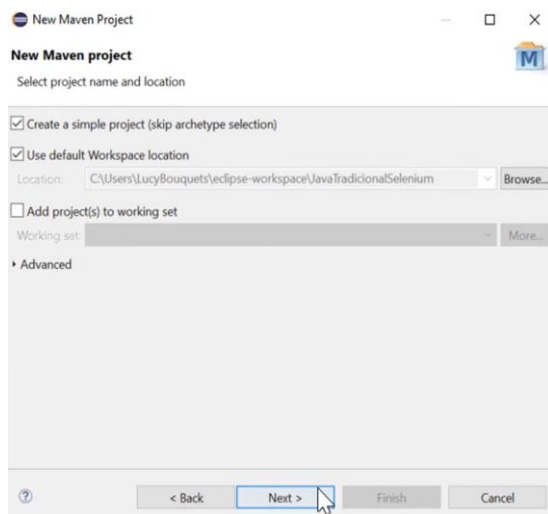
File/new/project



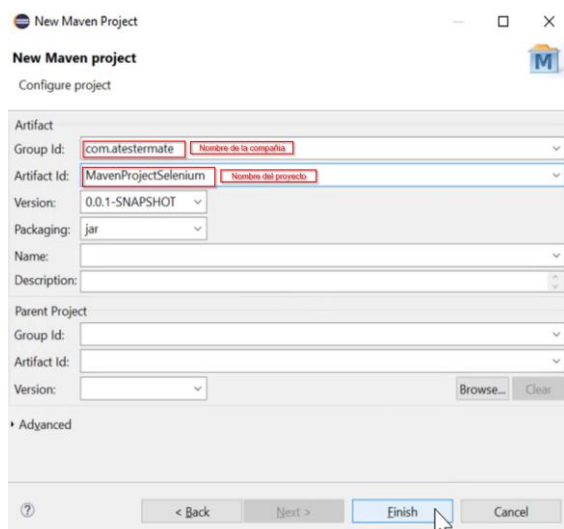
Seleccionamos Maven project



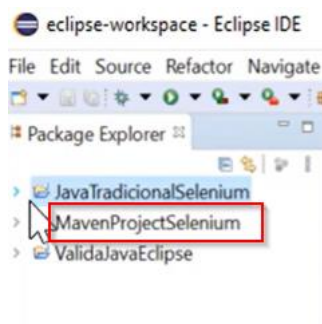
Marcamos los dos primeros check y pulsamos Next



Introducimos el nombre de la compañía y del proyecto

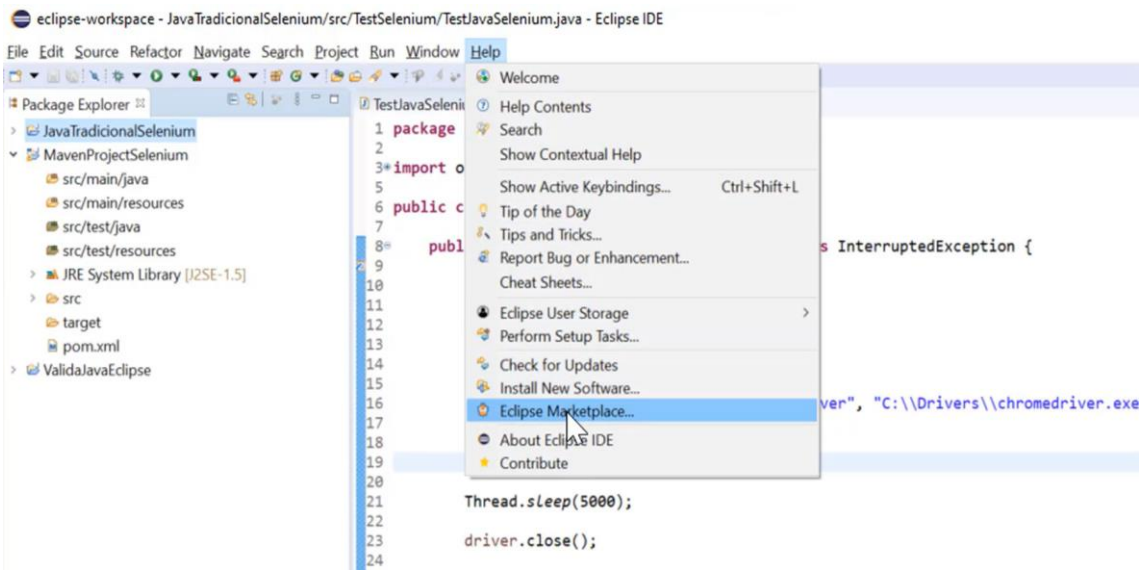


Se crea el proyecto

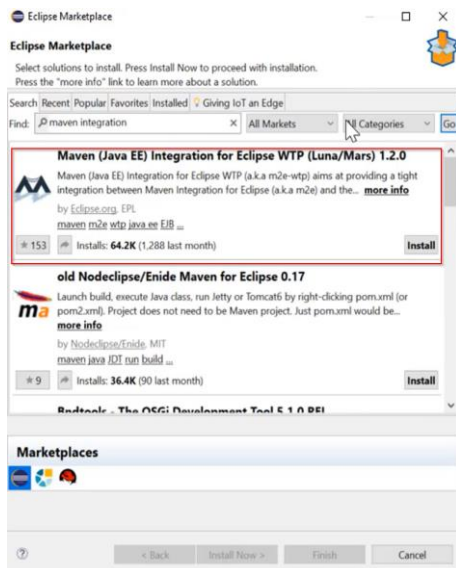


Instalación de plugin

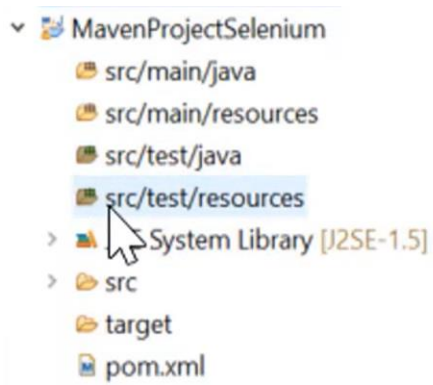
Tenemos que crear un plugin para que Eclipse pueda usar el Maven, para eso vamos a help/Eclipse Marketplace



Buscamos y instalamos Maven integration



Instalación de repositorio en pom



En el proyecto se observa el src/test/java: aquí se guardara todos los script de java


En src/test/resource: se guarda todos los recursos que no son de java

Pom.xml: aquí se guarda las dependencias Maven

Nos vamos a <https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java>


Found 1109 results

Sort: **relevance** | popular | newest




Selenium Java
org.seleniumhq.selenium » selenium-java
1,681 usages
Apache

Selenium provides support for the automation of web browsers. It provides extensions to emulate user interaction with browsers, a distribution server for scaling browser allocation, and the infrastructure for implementations of the W3C WebDriver specification.
Last Release on Nov 1, 2023



Selenium Server
org.seleniumhq.selenium » selenium-server
320 usages
Apache

Selenium automates browsers. That's it! What you do with that power is entirely up to you.
Last Release on Jul 2, 2019




Selenium API
org.seleniumhq.selenium » selenium-api
498 usages
Apache

Selenium automates browsers. That's it! What you do with that power is entirely up to you.

Cogemos el más reciente

Home » org.seleniumhq.selenium » selenium-java



Selenium Java
Selenium provides support for the automation of web browsers. It provides extensions to emulate user interaction with browsers, a distribution server for scaling browser allocation, and the infrastructure for implementations of the W3C WebDriver specification.

License	Apache 2.0
Categories	Web Testing
Tags	quality selenium testing web
Ranking	#284 in MvnRepository (See Top Artifacts) #1 in Web Testing
Used By	1,681 artifacts

Central (146) Atlassian 3rdParty (2) EmeryyaPub (3) Alfresco (1) ICM (3)					
Version	Vulnerabilities	Repository	Usages	Date	
4.15.x	4.15.0	Central	37	Nov 01, 2023	
4.14.x	4.14.1	Central	27	Oct 12, 2023	
	4.14.0	Central	10	Oct 10, 2023	
4.13.x	4.13.0	Central	40	Sep 25, 2023	

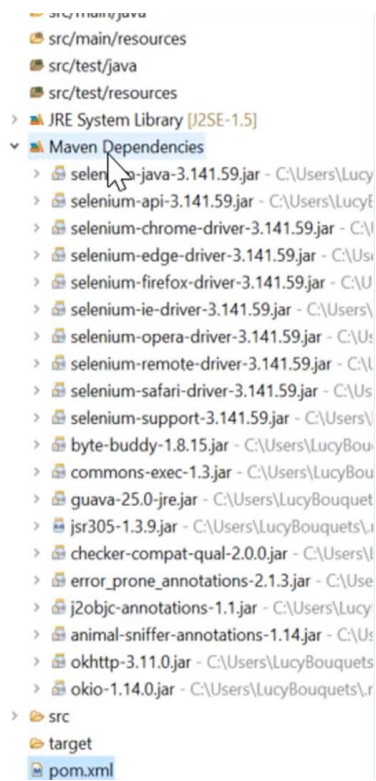
Se introduce la dependencia de selenium y se da a guardar

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2003
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.atestermate</groupId>
  <artifactId>MavenProjectSelenium</artifactId>
  <version>0.0.1-SNAPSHOT</version>

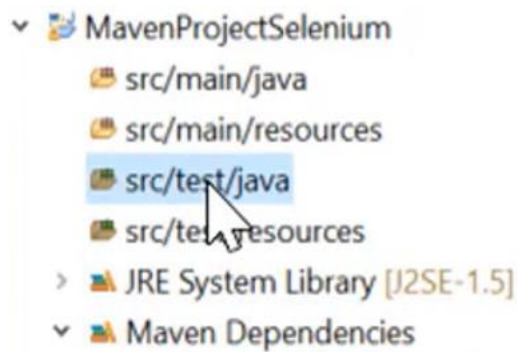
  <dependencies>
    <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
    <dependency>
      <groupId>org.seleniumhq.selenium</groupId>
      <artifactId>selenium-java</artifactId>
      <version>3.141.59</version>
    </dependency>

  </dependencies>
</project>
```

Comprobamos las dependencias:



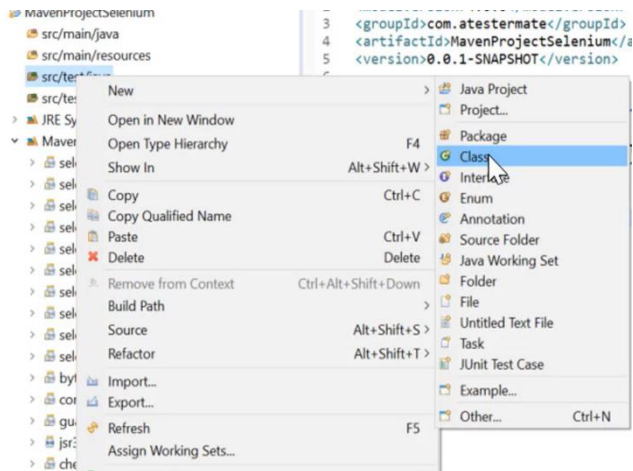
Creación de clase



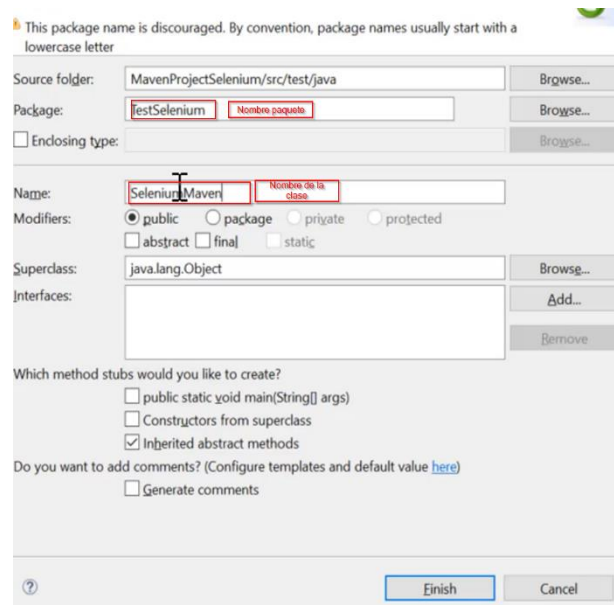
Creamos una clase y en
src/test/java pulsamos clic
derecho

Existen dos maneras de ejecutar el código puede ser desde el propio ide o desde Maven se hará de las dos maneras:

Lanzar desde IDE:



Primero creamos una clase



Le damos un nombre al paquete y a la clase.

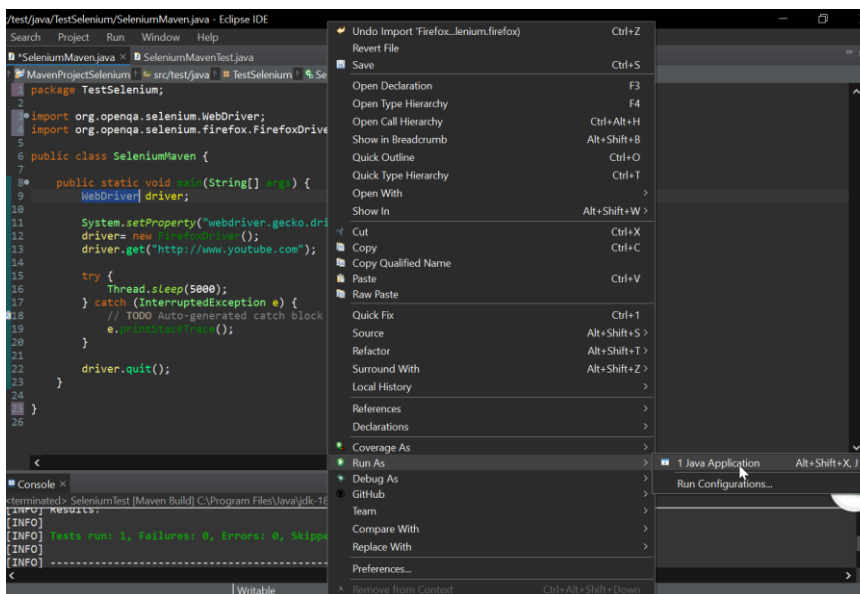
Marcamos la opción public static void main, para crear el main

```

1 SeleniumMaven.java x SeleniumMavenTest.java
2 MavenProjectSelenium src/test/java TestSelenium SeleniumMaven main(String[]) : void
3 package TestSelenium;
4
5 import org.openqa.selenium.WebDriver;
6 import org.openqa.selenium.firefox.FirefoxDriver;
7
8 public class SeleniumMaven {
9
10     public static void main(String[] args) {
11         WebDriver driver;
12
13         System.setProperty("webdriver.gecko.driver", "src\\test\\resources\\geckodriver.exe");
14         driver= new FirefoxDriver();
15         driver.get("http://www.youtube.com");
16
17         try {
18             Thread.sleep(5000);
19         } catch (InterruptedException e) {
20             // TODO Auto-generated catch block
21             e.printStackTrace();
22         }
23
24         driver.quit();
25     }
26 }

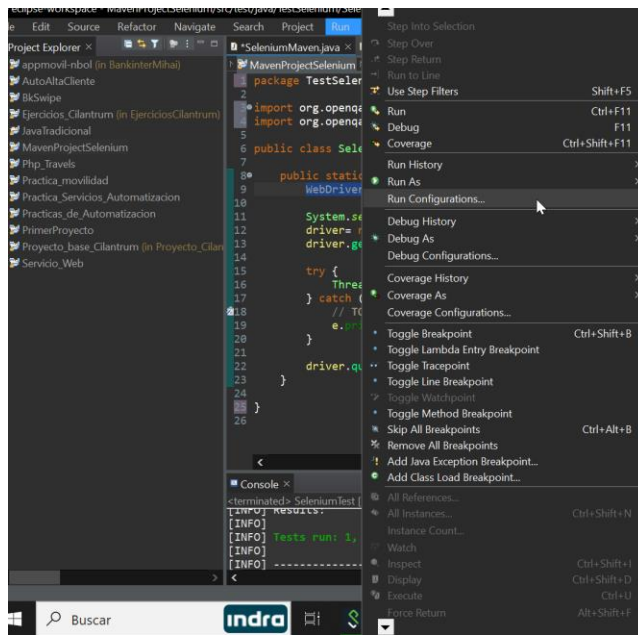
```

Dentro del main insertamos el código, no olvidar el driver del navegador en este caso el driver geckodriver.exe de mozilla

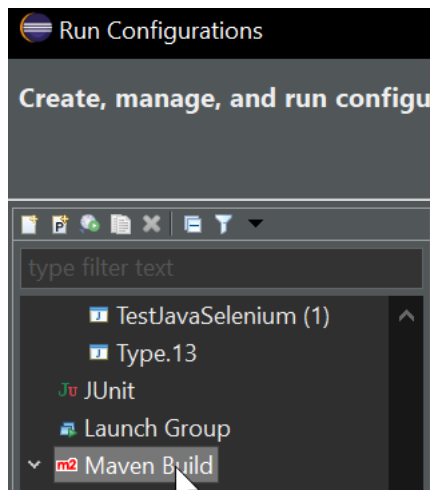


Corremos con Java Application

Ejecutar proyecto como Maven:

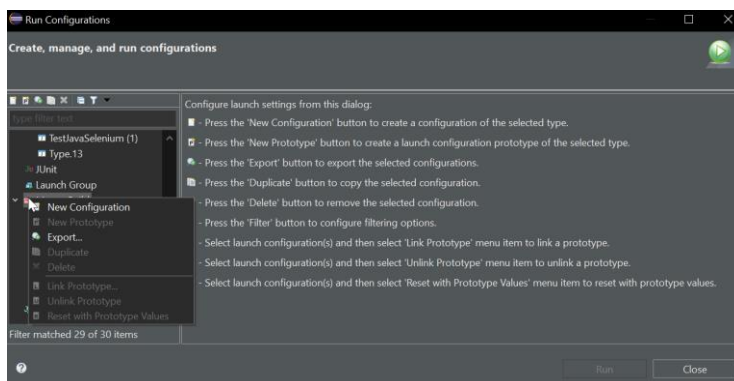


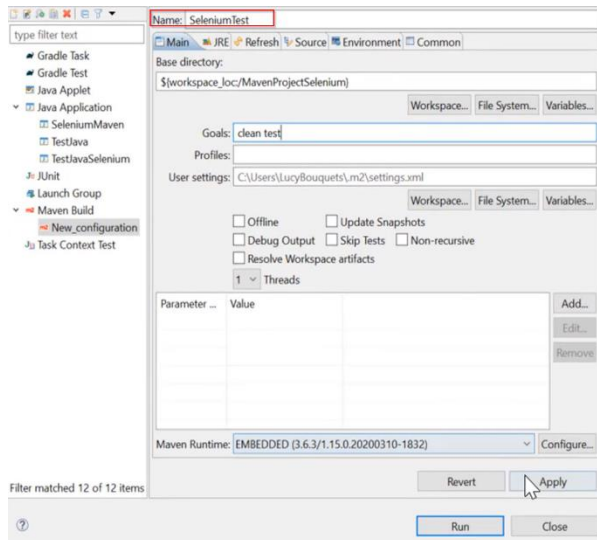
Nos vamos a Run Configurations



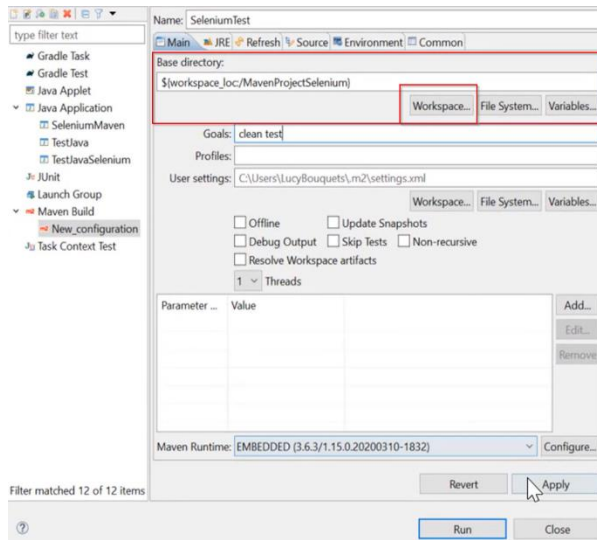
Seleccionamos Maven Build

Clic derecho y pulsamos new Configuration

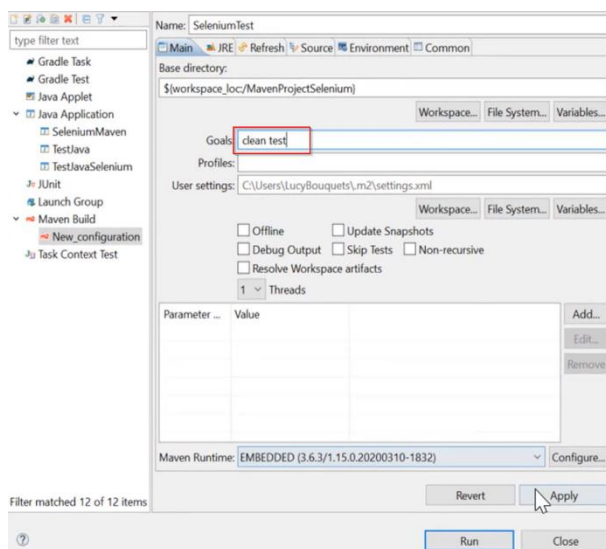




Insertamos el nombre



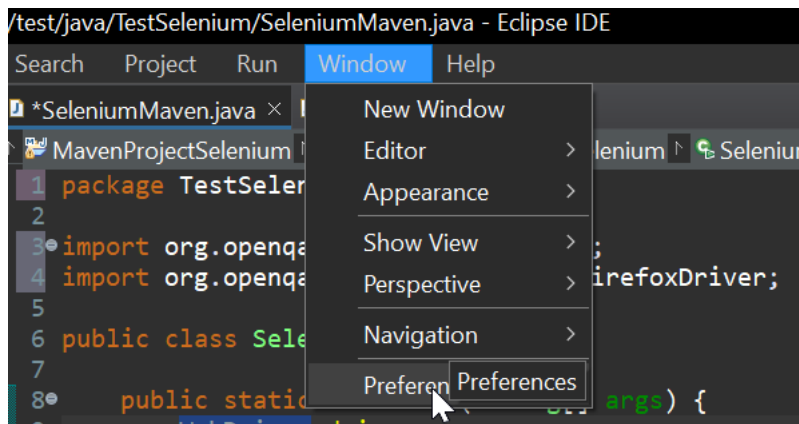
Seleccionamos el proyecto



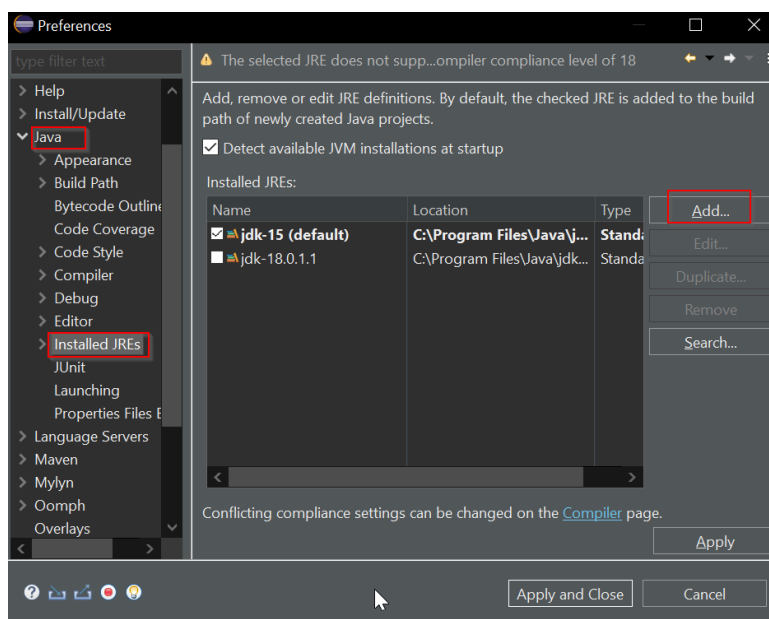
Le decimos que primero haga un clean y luego un test

Pulsamos Apply

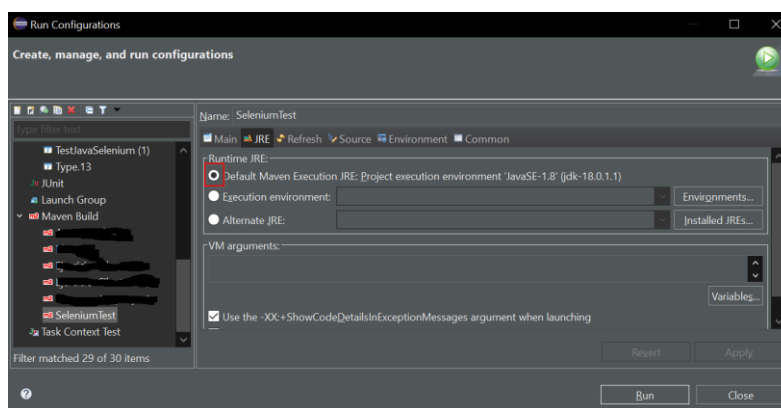
Luego se necesita Instalar el JDK



Vamos a
Windows/preferences

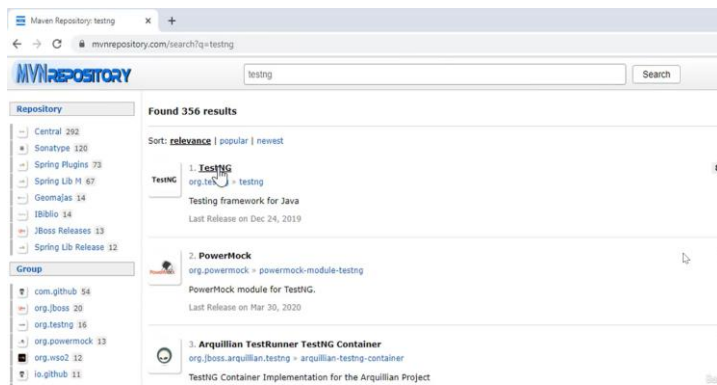


java/installed JREs/Add y
agregamos la carpeta
donde se encuentra el JDK

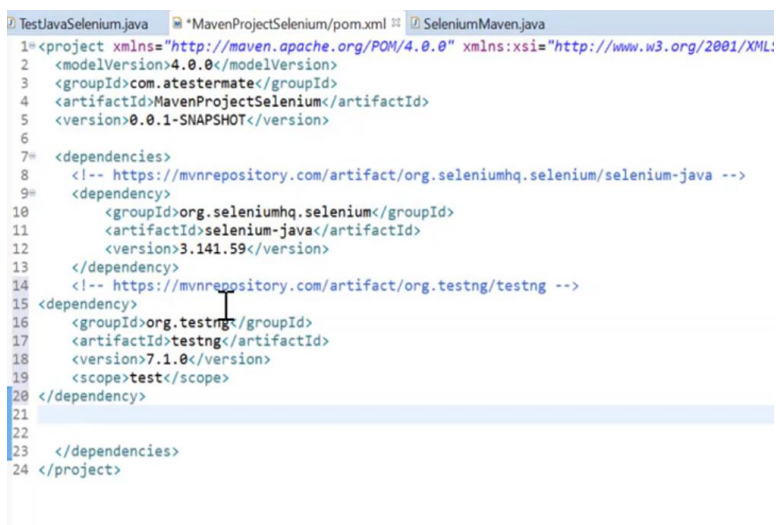


Volvemos Abrir el
mavenBuild de
SeleniumTest y en la
pestaña de JRE
seleccionamos el default
Maven JRE

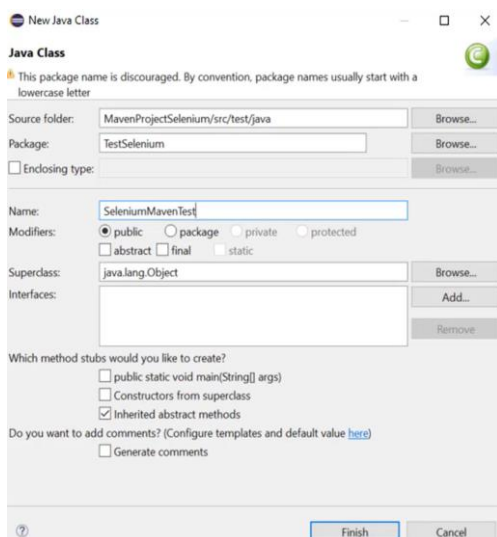
Luego necesitamos descargar la dependencia de TestNG



Pegamos la dependencia y le damos a guardar



Después creamos una nueva clase sin main

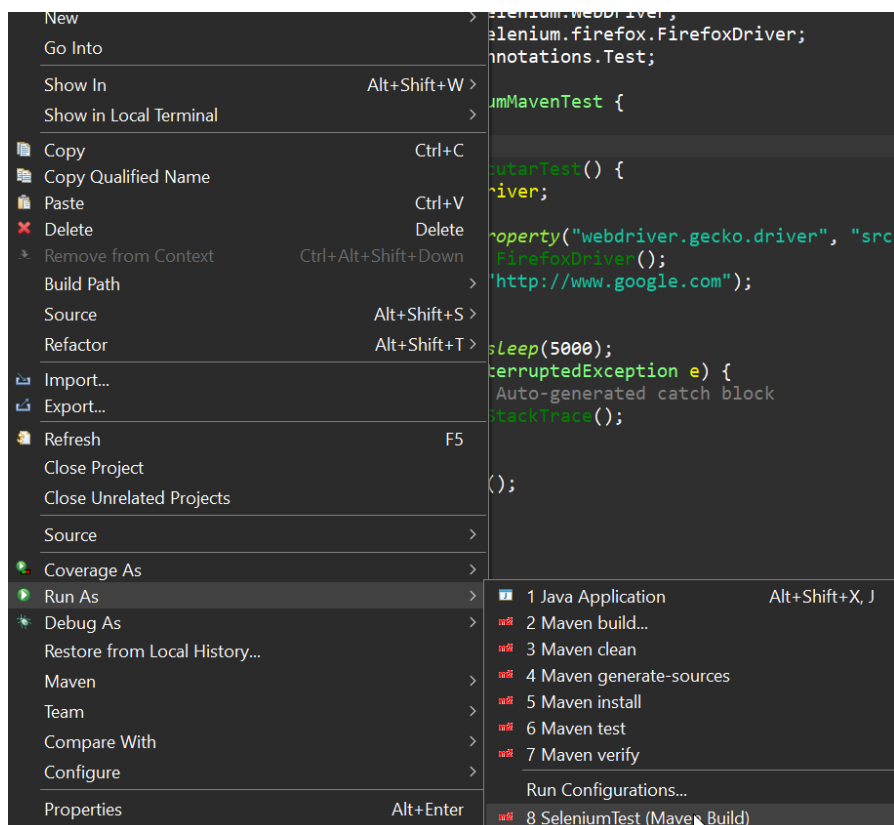


Dentro de la clase crear un método public con una etiqueta @Test y dentro del método el código

Ejemplo:

```
SeleniumMavenTest.java ×
MavenProjectSelenium ▾ src/test/java ▾ TestSelenium ▾ SeleniumMavenTest ▾ • ejecutarTest() : void
1 package TestSelenium;
2
3 import org.openqa.selenium.WebDriver;
4 import org.openqa.selenium.firefox.FirefoxDriver;
5 import org.testng.annotations.Test;
6
7 public class SeleniumMavenTest {
8
9     @Test
10    public void ejecutarTest() {
11        WebDriver driver;
12
13        System.setProperty("webdriver.gecko.driver", "src\\test\\resources\\geckodriver.exe");
14        driver= new FirefoxDriver();
15        driver.get("http://www.google.com");
16
17        try {
18            Thread.sleep(5000);
19        } catch (InterruptedException e) {
20            // TODO Auto-generated catch block
21            e.printStackTrace();
22        }
23
24        driver.quit();
25    }
26
27 }
28
```

Por último ejecutamos con el Maven que hemos creado anterior mente



Hay que recordar el driver del navegador en este caso al ser ejecutado en Firefox se llama geckodriver.exe

Explicación del código:

```
public class SeleniumMavenTest {  
  
    @Test  
    public void ejecutarTest() {  
        WebDriver driver;  
  
        System.setProperty("webdriver.gecko.driver", "src\\test\\resources\\geckodriver.exe");  
        driver= new FirefoxDriver();  
        driver.get("http://www.google.com");  
  
        try {  
            Thread.sleep(5000);  
        } catch (InterruptedException e) {  
            // TODO Auto-generated catch block  
            e.printStackTrace();  
        }  
  
        driver.quit();  
    }  
}
```

1. Declaración de la variable driver:

javaCopy code

Se declara una variable llamada `driver` del tipo `WebDriver`, que es una interfaz en Selenium utilizada para la interacción con los navegadores web.

2. Configuración del controlador de Gecko (para Firefox):

javaCopy code

```
System.setProperty("webdriver.gecko.driver",  
"src\\test\\resources\\geckodriver.exe")
```

Se establece la propiedad del sistema "webdriver.gecko.driver" con la ubicación del controlador de Gecko (geckodriver.exe). Este controlador es necesario para que Selenium pueda interactuar con el navegador Firefox.

3. Inicialización del objeto driver como un nuevo FirefoxDriver:

javaCopy code

```
driver = new FirefoxDriver();
```

Se instancia un objeto de la clase `FirefoxDriver` y se asigna a la variable `driver`. Esto crea una nueva instancia del navegador Firefox.

4. Abrir la página web de YouTube:

javaCopy code

```
driver.get("http://www.youtube.com")
```

Se utiliza el método `get` de `driver` para abrir la URL especificada en el navegador. En este caso, la URL es "<http://www.youtube.com>".

5. Esperar 5000 milisegundos (5 segundos):

javaCopy code

```
try {  
    Thread.sleep(5000);  
} catch (InterruptedException e) {  
    e.printStackTrace();  
}
```

Se utiliza un bloque `try-catch` para manejar la excepción `InterruptedException`. Durante la ejecución, el hilo actual (en este caso, el programa) se duerme (o pausa) durante 5000 milisegundos (5 segundos).

Esto se hace para esperar que la página cargue completamente antes de continuar con el código.

6. **Cerrar el navegador al finalizar:**

javaCopy code

Se utiliza el método `quit` de `driver` para cerrar el navegador y liberar los recursos asociados.

En resumen, este código automatiza la apertura del navegador Firefox, navega a la página de YouTube, espera 5 segundos y luego cierra el navegador. Este es un ejemplo simple de automatización de pruebas o de interacción automatizada con un sitio web.