

Creación de Proyecto Base con IntelliJ

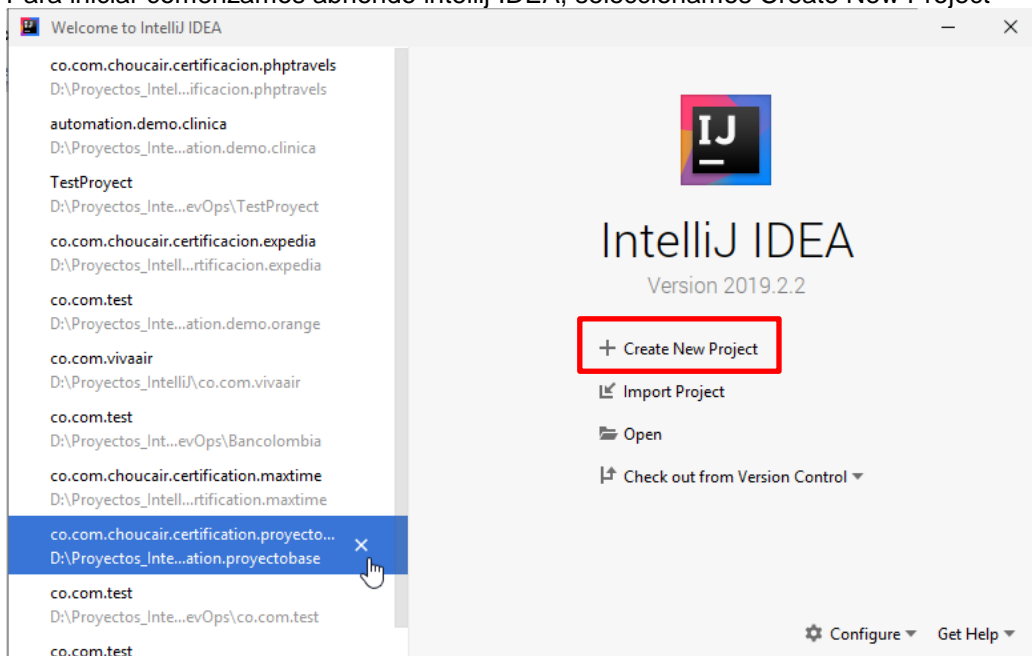
En esta guía veremos cómo crear un proyecto desde cero en el IDE **IntelliJ IDEA** utilizando el Patrón de Diseño **Screenplay**.



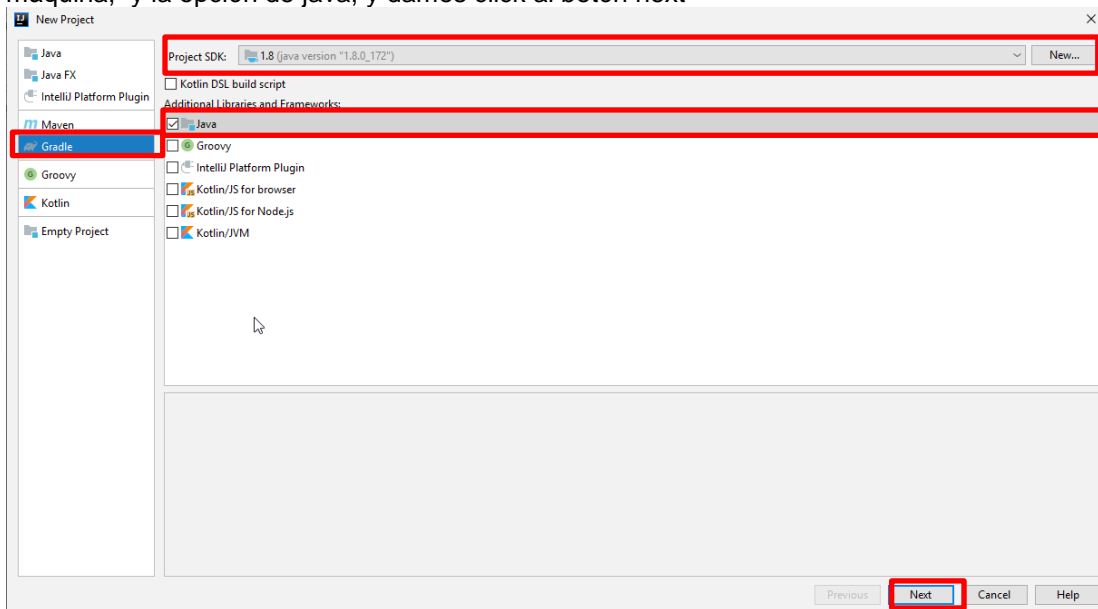
¡EMPECEMOS!



Para iniciar comenzamos abriendo IntelliJ IDEA, seleccionamos Create New Project

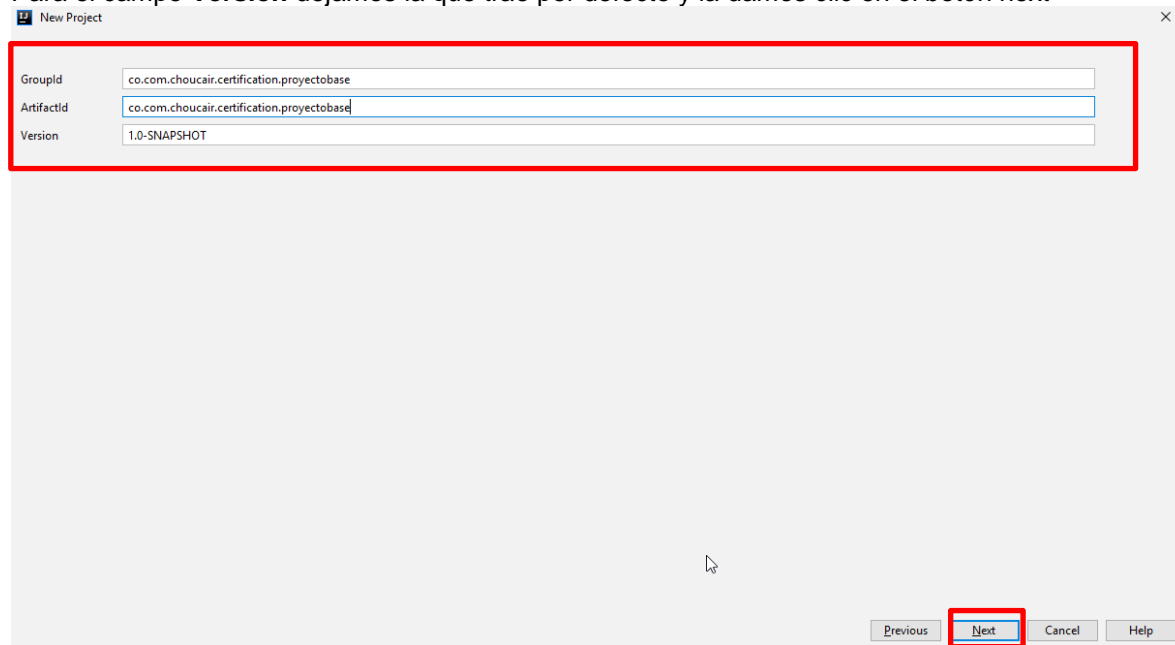


Luego seleccionamos la opción Gradle, escogemos la versión del SDK que tengamos instalado en nuestra máquina, y la opción de java, y damos click al botón next



En la opción GroupId y ArtifactId colocamos:
co.com.choucair.certification.[**nombreproyecto**]

Para el campo **Versión** dejamos la que trae por defecto y la damos clic en el botón next



New Project

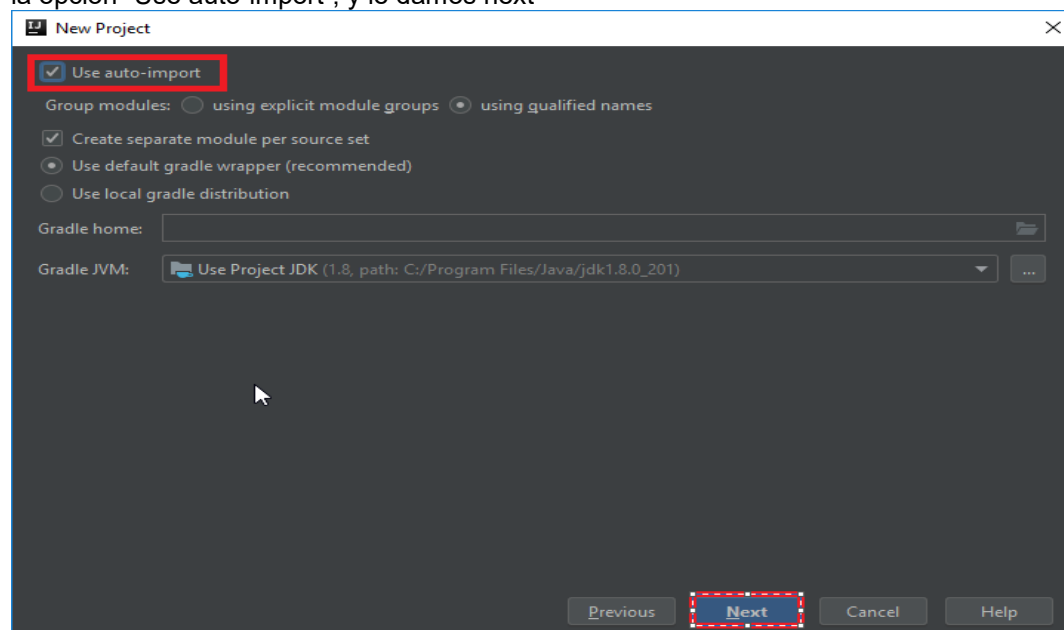
GroupId: co.com.choucair.certification.projectobase

ArtifactId: co.com.choucair.certification.projectobase

Version: 1.0-SNAPSHOT

Previous Next Cancel Help

Importante: En versiones posteriores a la 19 es probable que tengan que hacer este paso. Seleccionamos la opción "Use auto-import", y le damos next



New Project

☒ Use auto-import

Group modules: ☐ using explicit module groups ☒ using qualified names

☒ Create separate module per source set

☒ Use default gradle wrapper (recommended)

☐ Use local gradle distribution

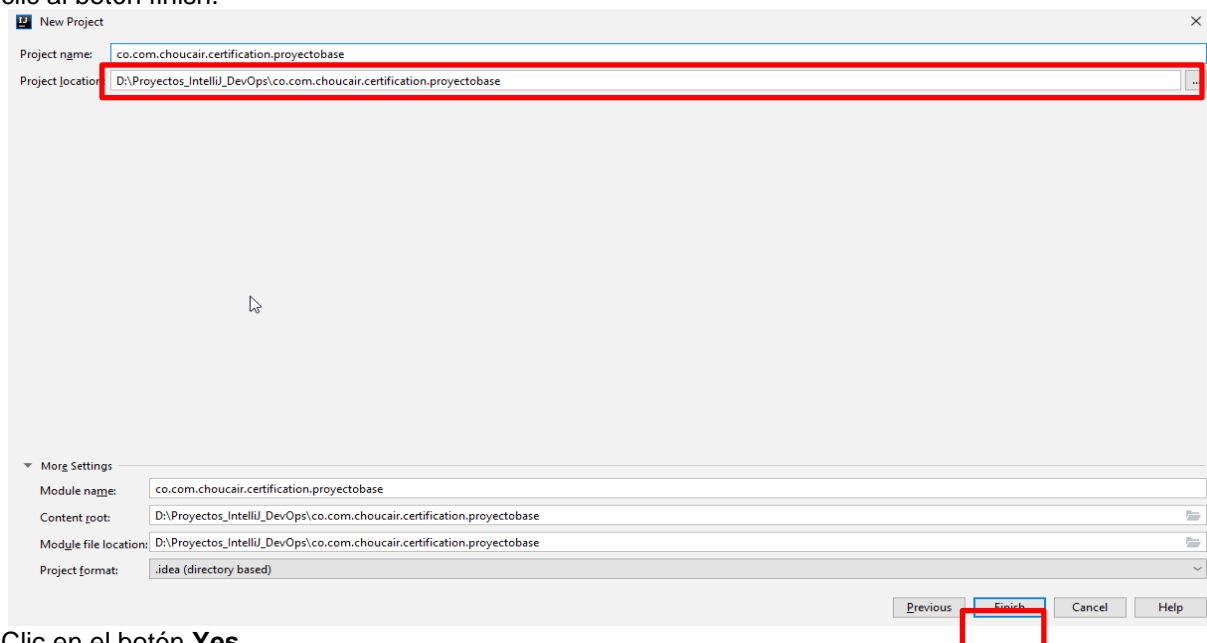
Gradle home: [Browse]

Gradle JVM: Use Project JDK (1.8, path: C:/Program Files/Java/jdk1.8.0_201) [More]

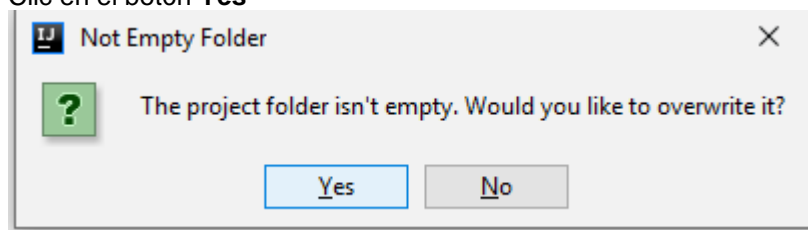
Previous Next Cancel Help



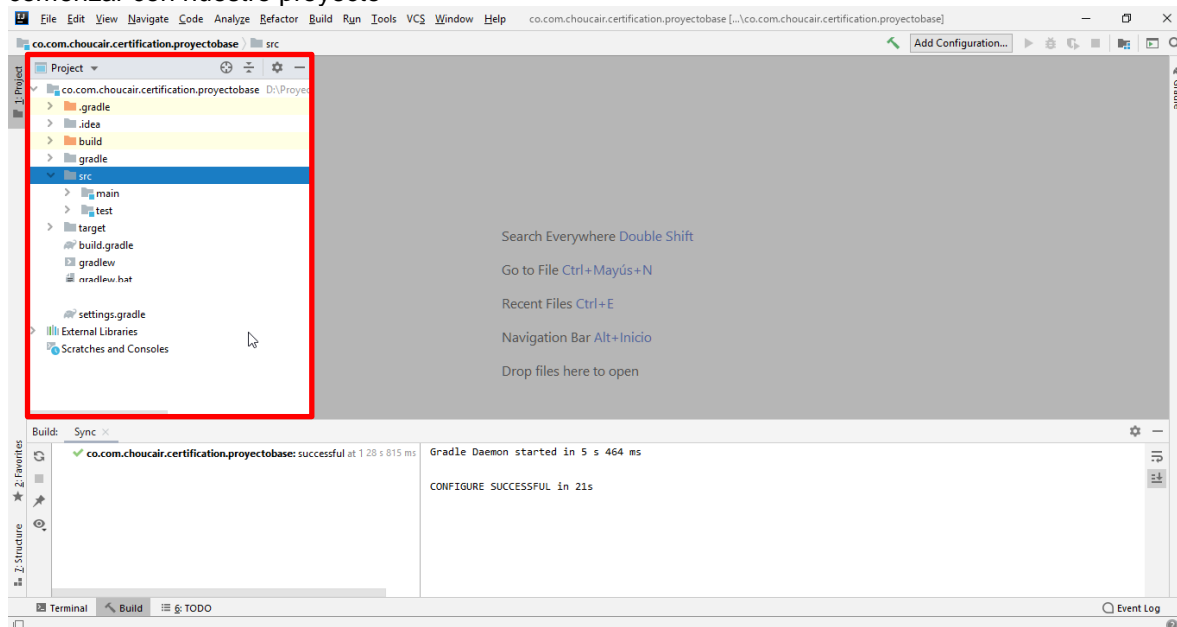
Seleccionamos **Project location:** y buscamos la ruta donde queremos crear nuestro proyecto y le damos clic al botón finish.



Clic en el botón **Yes**



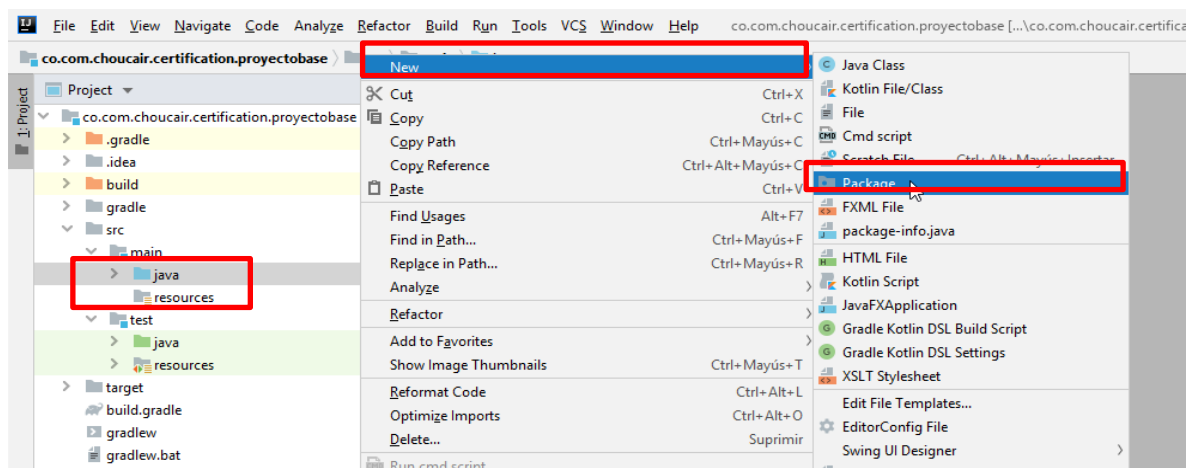
Como podemos ver, se crea la estructura inicial del proyecto y nos descarga unas primeras librerías para comenzar con nuestro proyecto



Procedemos a crear el arquetipo para trabajar con el Patrón Screenplay. Para esto desplegamos la carpeta `src/main` y en la carpeta `java`, damos clic derecho sobre la misma y seleccionamos `New>Package`, esto nos permitirá crear los paquetes necesarios para nuestro proyecto `co.com.choucair.certification.[nombreproyecto]`. Los paquetes son:

- exceptions**
- interactions**
- model**
- questions**
- tasks**
- userinterface**
- util**



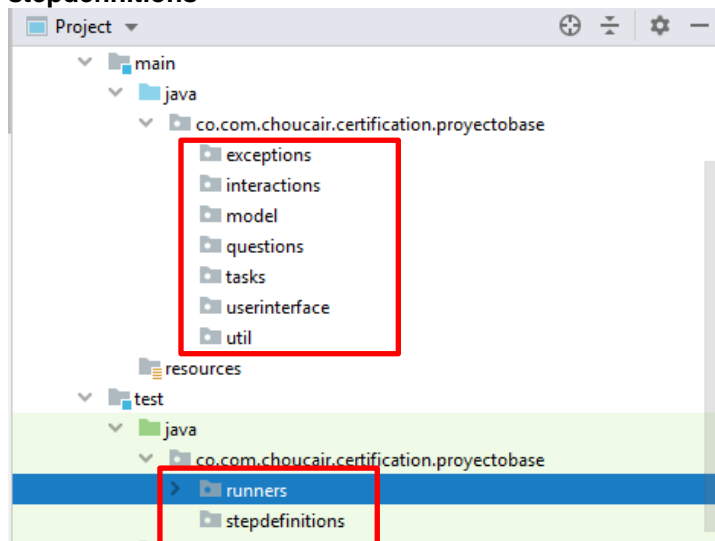


Nuestro proyecto debe quedar así una vez creamos nuestras sub-paquetes en la ruta `src/main/java/co.com.choucair.certificacion.proyectobase`.

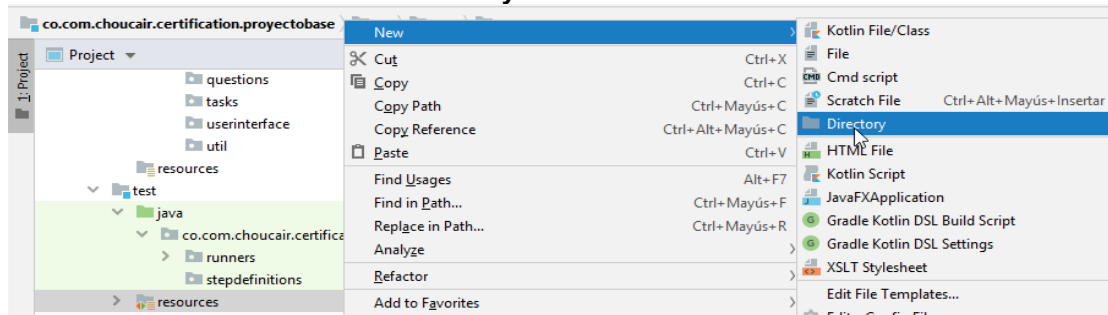
El anterior paso, lo debemos repetir para la ruta `src/test/java`, donde inicialmente se crea el paquete con el nombre `co.com.choucair.certificacion.[nombreproyecto]` y finalmente dentro de este último vamos a crear los paquetes:

runners

stepdefinitions



Ahora procedemos a crear las carpetas **features** y **driver** en la ruta `src/test/resources`, esto dando clic derecho sobre **resources**>**New**>**Directory**.



Ahora vamos a modificar el archivo **"build.gradle"** y cambiamos su contenido completo por el que se presente a continuación:

```
repositories {
    mavenCentral()
}

buildscript {
    repositories {
        mavenCentral()
    }
    dependencies {
        classpath("net.serenity-bdd:serenity-gradle-plugin:1.9.9")
    }
}

apply plugin: 'java'
apply plugin: 'eclipse'
apply plugin: 'idea'
apply plugin: 'net.serenity-bdd.aggregator'

compileJava.options.encoding = "UTF-8"
compileTestJava.options.encoding = "UTF-8"

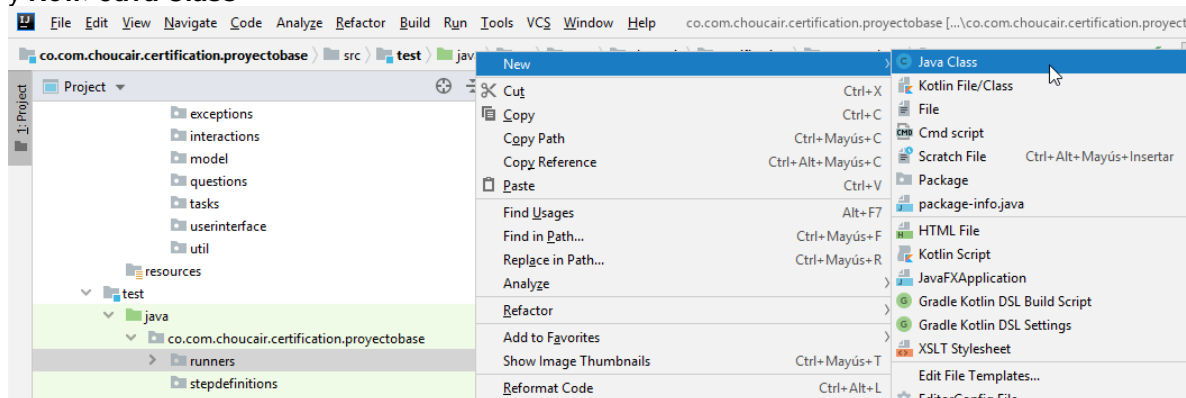
dependencies {
    testCompile 'net.serenity-bdd:serenity-core:1.1.1'
    testCompile 'net.serenity-bdd:serenity-junit:1.1.1'
    testCompile('junit:junit:4.12')
    testCompile('org.assertj:assertj-core:1.7.0')
    testCompile('org.slf4j:slf4j-simple:1.7.7')

    compile 'net.serenity-bdd:serenity-core:1.9.9'
    compile 'net.serenity-bdd:serenity-junit:1.9.9'
    compile 'net.serenity-bdd:serenity-cucumber:1.9.5'
    compile 'net.serenity-bdd:serenity-screenplay:1.9.9'
    compile 'net.serenity-bdd:serenity-screenplay-webdriver:1.9.9'
}

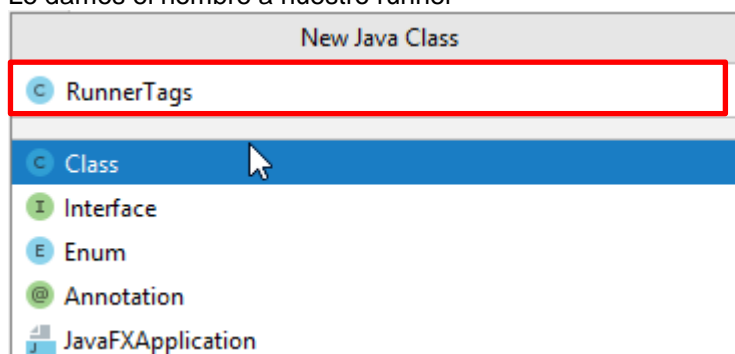
gradle.startParameter.continueOnFailure = true
```



Ahora vamos a crear nuestra clase Runner, para este proceso nos posicionamos en la ruta `src/test/java/co.com.choucair.certificacion.proyectobase/runners` y damos clic derecho en la carpeta runner y **New>Java Class**



Le damos el nombre a nuestro runner



Cuando creamos la clase, debemos colcoar el siguiente código y eliminar lo que se carga en la misma:

```
import cucumber.api.CucumberOptions;  
import cucumber.api.SnippetType;  
import net.serenitybdd.cucumber.CucumberWithSerenity;  
import org.junit.runner.RunWith;
```

```
@RunWith(CucumberWithSerenity.class)  
@CucumberOptions (features = "src/test/resources/features/demo.feature",  
                  tags = "@tag1",  
                  glue = "co.com.choucair.certificacion.proyectobase.stepdefinitions",  
                  snippets = SnippetType.CAMELCASE )
```

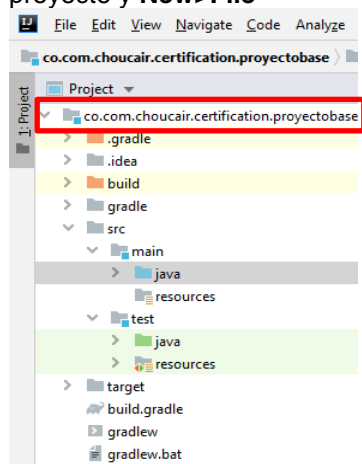
```
public class RunnerTags {  
  
}
```

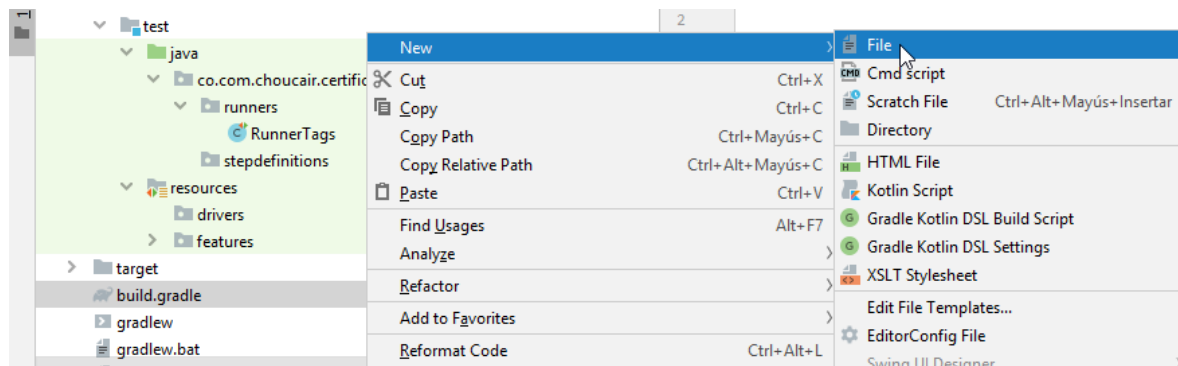


Nuestra clase RunnerTags debería quedar así:

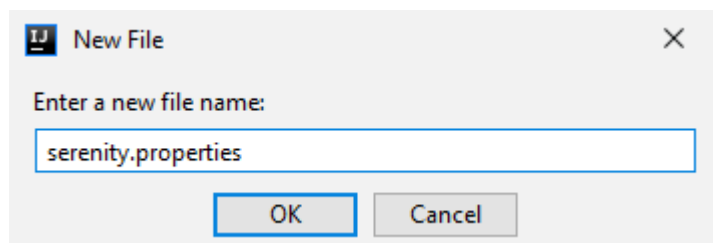
```
RunnerTags.java
1 package co.com.choucair.certification.proyectobase.runners;
2
3 import cucumber.api.CucumberOptions;
4 import cucumber.api.SnippetType;
5 import net.serenitybdd.cucumber.CucumberWithSerenity;
6 import org.junit.runner.RunWith;
7
8
9 @RunWith(CucumberWithSerenity.class)
10 @CucumberOptions(
11     features="src/test/resources/features",
12     tags = "@Regresion",
13     glue = "co.com.choucair.certification.proyectobase.stepdefinitions",
14     snippets = SnippetType.CAMEL_CASE)
15
16 public class RunnerTags {
17 }
18
```

Ahora procedemos a crear el archivo **serenity.properties**, para eso damos clic derecho sobre la raíz del proyecto y **New>File**





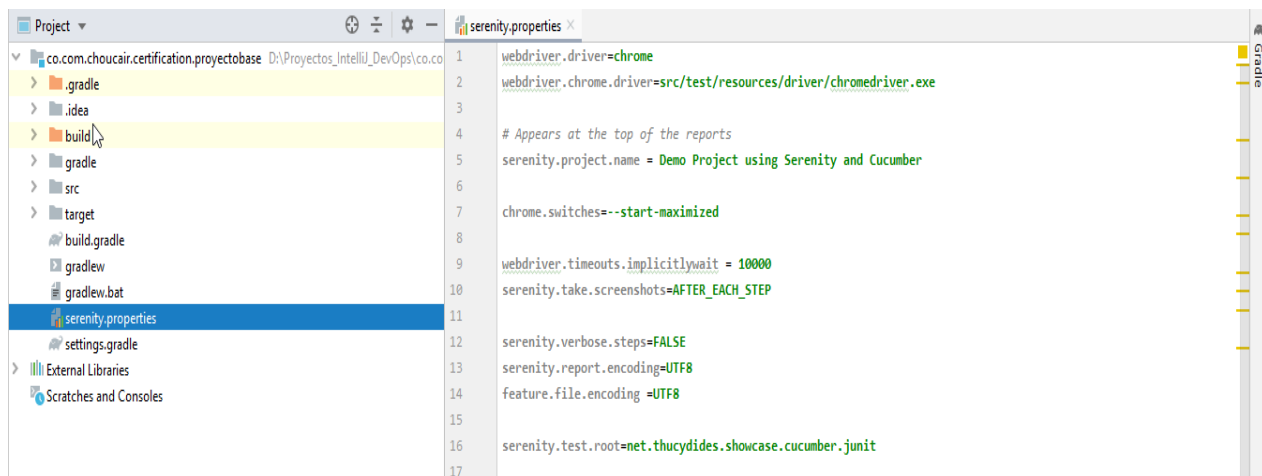
Escribimos **serenity.properties** y damos clic en **OK**



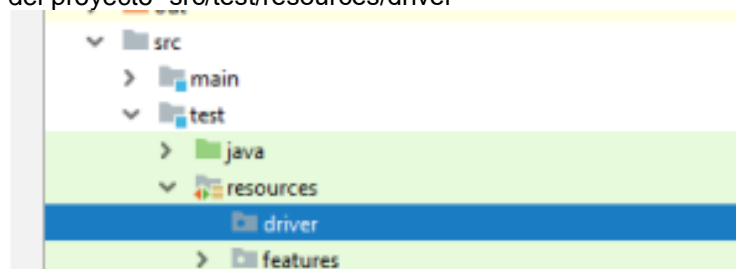
El contenido del archivo lo vamos a reemplazar por las siguientes linea:

```
webdriver.driver=chrome
webdriver.chrome.driver=src/test/resources/driver/chromedriver.exe
serenity.project.name = Demo Project using Serenity and Cucumber
chrome.switches=--start-maximized
webdriver.timeouts.implicitlywait = 10000
serenity.take.screenshots=AFTER_EACH_STEP
serenity.verbose.steps=FALSE
serenity.report.encoding=UTF8
feature.file.encoding =UTF8
serenity.test.root=net.thucydides.showcase.cucumber.junit
```

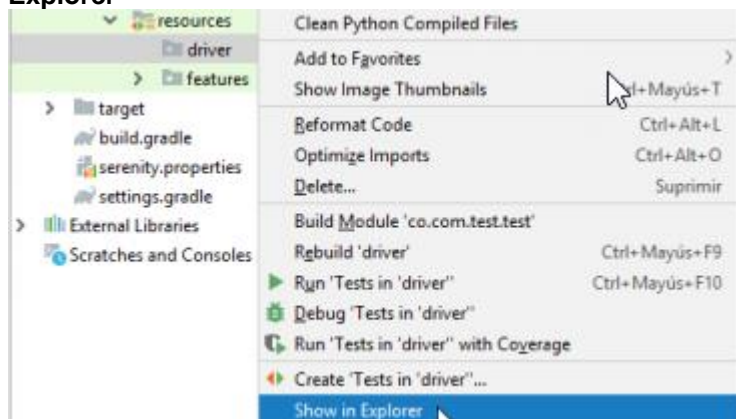




Seguimos con la descarga del driver para nuestro navegador. Este archivo lo debemos colocar en la ruta del proyecto “src/test/resources/driver”

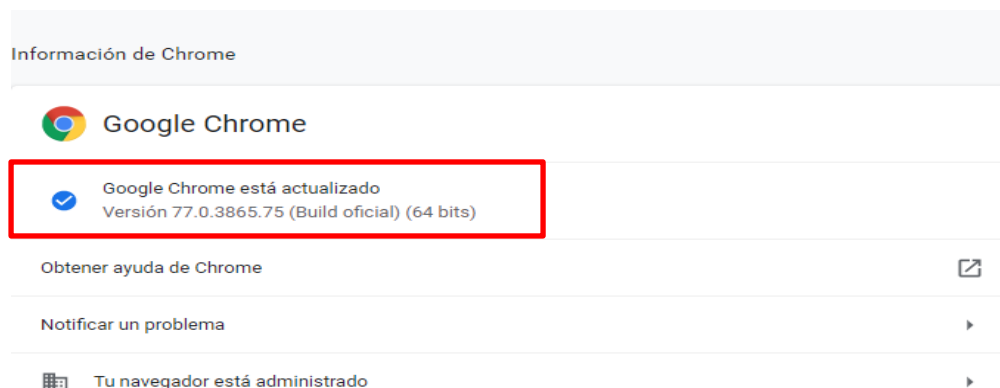
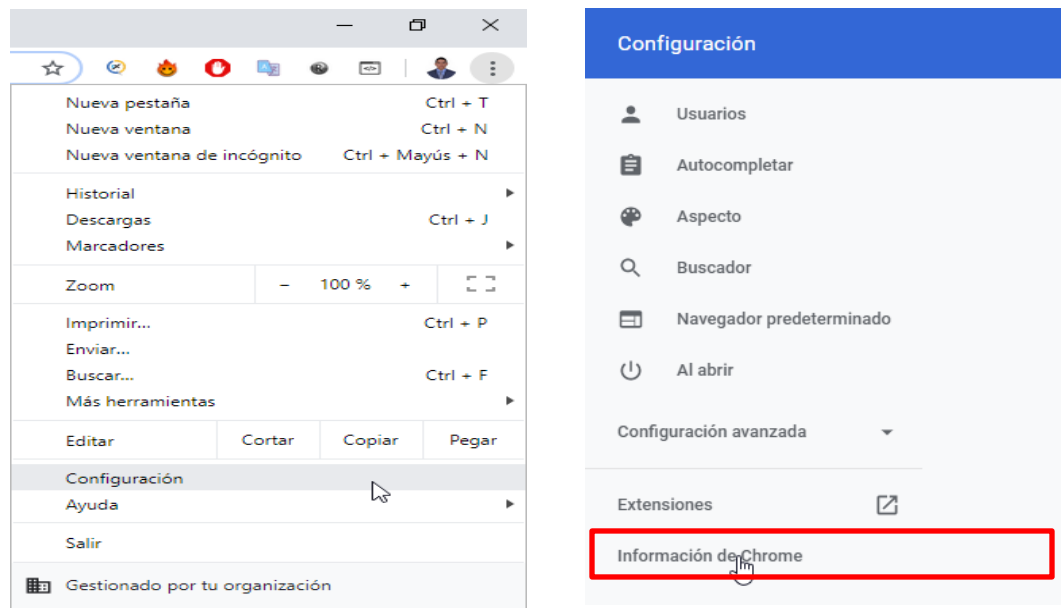


Para acceder a esta ruta vamos a dar clic derecho sobre la carpeta driver y clic en la opción “**Show in Explorer**”



Esto abrirá en una ventana la ruta física de esta carpeta. Posteriormente vamos a verificar la versión de nuestro navegador Chrome a través de la ruta **Configuración**, luego clic en **“Información de Chrome”**.

Para este caso tenemos la **Versión 77.0.3865.75 (Build oficial) (64 bits)**



Una vez hemos verificado la versión de Chrome, continuamos accediendo a la URL: <https://www.seleniumhq.org/download/>. Nos ubicamos en la siguiente sección y damos clic en link **latest** para el Browser driver **Google Chrome Driver**



Third Party Drivers, Bindings, and Plugins

Selenium can be extended through the use of plugins. Here are a number of plugins created and maintained by third parties. For more information on how to create your own plugin or have it listed, consult the docs.

Please note that these plugins are not supported, maintained, hosted, or endorsed by the Selenium project. In addition, be advised that the plugins listed below are not necessarily licensed under the Apache License v.2.0. Some of the plugins are available under another free and open source software license; others are only available under a proprietary license. Any questions about plugins and their license of distribution need to be raised with their respective developer(s).

Third Party Browser Drivers **NOT DEVELOPED** by seleniumhq

Browser

| | | | | |
|---------------------------------------|------------------------|----------------------------|-------------------------------|---------------------------------------|
| Mozilla GeckoDriver | latest | change log | issue tracker | Implementation Status |
| Google Chrome Driver | latest | change log | issue tracker | selenium wiki page |
| Opera | latest | | issue tracker | selenium wiki page |
| Microsoft Edge Driver | | | issue tracker | Implementation Status |

Damos clic a la versión más cercana a la de nuestro navegador.






Downloads

Current Releases

- If you are using Chrome version 78, please download [ChromeDriver 78.0.3904](#).
- If you are using Chrome version 77, please download [ChromeDriver 77.0.3865](#).
- If you are using Chrome version 76, please download [ChromeDriver 76.0.3809](#).

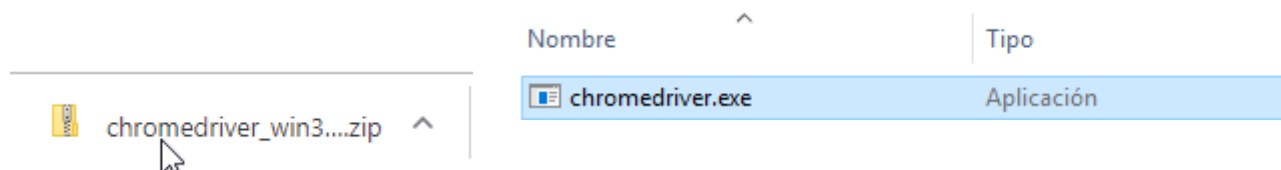
Y damos clic en **chromedriver_win32.zip** para descargar.

Index of /77.0.3865.40/

| | Name | Last modified | Size | ETag |
|---|--|---------------------|--------|----------------------------------|
|  | Parent Directory | | - | |
|  | chromedriver_linux64.zip | 2019-08-20 18:02:46 | 5.17MB | b4431816072192a2d36a10fa8cfd344 |
|  | chromedriver_mac64.zip | 2019-08-20 18:02:48 | 7.05MB | 812570697aadcd7a9038041b27437054 |
|  | chromedriver_win32.zip | 2019-08-20 18:02:49 | 4.54MB | 7e94b11b8157e856b918f64d1b4af424 |
|  | notes.txt | 2019-08-20 18:02:53 | 0.00MB | 0609e0eff91a2087279a1600bb37198e |



Una vez descargado recuerda descomprimir el archivo copiarlo y llevar el ejecutable **chromedriver.exe** a la ruta "src/test/resources/driver".



Hasta aquí ya tendríamos las bases para comenzar con el desarrollo de nuestra automatización.

Ahora ya puedes crear tu proyecto desde cero.

