**Introducción**

**Selenium** es un entorno de pruebas de software para aplicaciones basadas en la web.

Lenguajes de programación utilizados para selenium:

* Java
* C#
* Ruby
* Python
* Javascript (Node)

**JUnit** es un conjunto de clases **(*framework*)** que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente.

**JUnit** es también un medio de controlar las pruebas de regresión (búsqueda de bugs o errores), necesarias cuando una parte del código ha sido modificado y se desea ver que el nuevo código cumple con los requerimientos anteriores y que no se ha alterado su funcionalidad después de la nueva modificación.

**Maven** es una herramienta de software para la gestión y construcción de proyectos Java basado en un formato **XML**. Una característica clave de Maven es que está listo para usar en red. El motor incluido en su núcleo puede dinámicamente descargar plugins de un repositorio, el mismo repositorio que provee acceso a muchas versiones de diferentes proyectos Open Source en Java, de Apache y otras organizaciones y desarrolladores.

**Desarrollo**

Como crear un proyecto con selenium en eclipse:

1. Abrimos el IDE eclipse
2. File > New > Other… > Maven > Maven Project
3. Tildar la casilla que dice: Create a simple project (skip archetype selection) y dar a siguiente
4. En group id insertar un nombre para el proyecto, luego en artifact id hacemos lo mismo y dar a finish
5. Dentro del proyecto buscar pom.xml y hacer los siguientes pasos: Click derecho > Open with > Text editor
6. Dentro del pom.xml que abrimos como editor de texto deberán introducir las dependencias de jUnit y Maven Selenium.

jUnit dependency: <https://junit.org/junit4/dependency-info.html>

Maven Selenium dependency: <https://www.seleniumhq.org/download/maven.jsp>

Les tiene que quedar algo asi:

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.project</groupId>

<artifactId>project</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>3.141.59</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13-beta-1</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

1. Actualizar el navegador google chrome: entrar en los tres puntos arriba a la derecha y ir a ayuda luego información de Google Chrome, esperar que se actualice solo y fijarse que versión es.
2. Entrar en la página de driver chrome (<http://chromedriver.chromium.org/>) y descargar el driver con la versión de su google chrome.
3. Descomprimir la descarga en su escritorio, debe quedar una carpeta llamada chromedriver\_win32 y adentro un chromedriver\_win32.exe
4. En el eclipse, crear una carpeta en src/test/resources llamada chromedriver y luego importar dentro de esa carpeta el chromedriver\_win32.exe que está ubicada en el escritorio.
5. Para finalizar crear una clase con el nombre que quieran en src/test/java y para tener un ejemplo de cómo se utiliza el código vean el ejemplo en la carpeta del drive o les dejo un video tutorial de ayuda: <https://www.youtube.com/watch?v=O5kvqVriJdk&t=661s>

Básicamente selenium tiene tres secciones para ejecutarse: BEFORE, TEST y AFTER

***Before*** es lo que debería hacer antes de realizar la ejecución del código de prueba

private WebDriver driver;

@Before

public void setUp(){

System.setProperty("webdriver.chrome.driver","./src/test/resources/chromedriver/chromedriver.exe");

**Seteamos el chrome como driver**

driver = new ChromeDriver();

driver.manage().window().maximize();

**Maximizamos la ventana del chrome**

driver.get("<http://www.dolarhoy.com/>");

//driver.get("<https://www.google.com.ar/>");

**Le decimos a qué página queremos que redireccione**

}

***Test*** es lo que debería ejecutarse para realizar la prueba

@Test

public void testGooglePage(){

WebElement compra = driver.findElement(By.cssSelector("div[class='col-md-6 compra']"));

**Seleccionamos el HTML de la página y lo traemos como un objeto, para buscar el HTML necesitan ir a la página que desean utilizar y apretar CTRL+F, luego seleccionar que tipo de tag (div,span,etc) y entre corchetes la clase que lleva si localizo algún tag con esa clase se ve remarcado en color**

WebElement venta = driver.findElement(By.cssSelector("div[class='col-md-6 venta']"));

List<String> informacion = new ArrayList<String>();

informacion.add(compra.getText());

informacion.add(venta.getText());

**Con el método getText() agarramos todo el contenido de texto que tenga ese tag**

for(int i=0; i<informacion.size(); i++){

System.out.println(informacion.get(i));

}

//WebElement searchbox = driver.findElement(By.name("q"));

//searchbox.clear();

**El método clear() se utiliza para vaciar el contenido de algún input en la mayoría del tiempo**

//searchbox.sendKeys("dolay hoy");

**El método sendKeys(); se utiliza para enviar texto y que el software lo escriba en un input**

//searchbox.submit();

**El método submit() es la acción de dar un “enter”**

}

Por último, ***after*** que es lo que debería hacer cuando se termina de ejecutar el test

@After

public void tearDown(){

driver.quit();

}

**En este caso solamente cerramos la página web y finalizamos la ejecución.**