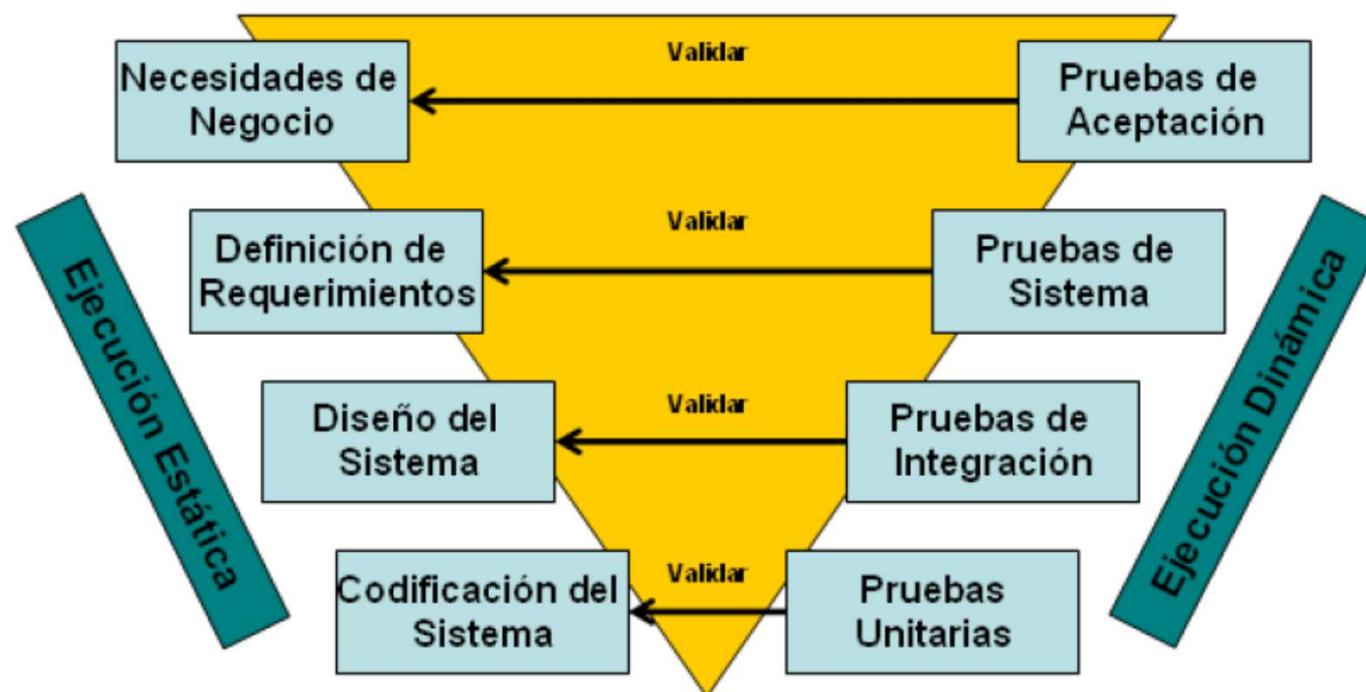


Selenium



Pruebas

- Los test no se realizan para probar que algo funciona, sino para encontrar fallos.
- Se pueden probar distintas fases del desarrollo (modelo en V)



Pruebas de Caja Blanca

- Tipo de pruebas de software que se realiza sobre las funciones internas de un módulo.
- Buscan recorrer todos los caminos posibles del modulo, cerciorándose de que no fallen.
- Dado que probar todo es inviable en la mayor parte de los casos, se define **cobertura** como una medida porcentual de cuánto código se ha cubierto.
- **Las pruebas de caja blanca nos convencen de que un programa hace bien lo que hace, pero no de que haga lo que necesitamos.**

Pruebas de Caja Negra

- Se dice que una prueba es de **caja negra** cuando prescinde de los detalles del código y se limita a lo que se ve desde el exterior. Intenta descubrir casos y circunstancias en los que el módulo no hace lo que se espera de él, ejercitan los requisitos funcionales.
- Las pruebas de caja negra están especialmente indicadas en aquellos módulos que van a ser interfaz con el usuario.

Probando que funcione el software

- Existen en el mercado numerosos **frameworks** que permiten probar el código desarrollado:
 - JUnit, NGUnit, Selenium, dbUnit, httpUnit, Cucumber, EasyMock, Mockito, Concordion, ...
- La base sobre la que se fundamentan es la **Validación Rojo-Verde**.

Probando las cualidades del software

- **Correcto.** Se comporta según los requisitos.
- **Robusto.** Se comporta de forma razonable, aun en situaciones inesperadas.
- **Confiable.** Se comporta según las expectativas del usuario (Correcto + Robusto).
- **Eficiente.** Emplea los recursos justos.
- **Amigable.** Fácil de utilizar.
- **Verificable.** Sus características pueden ser comprobadas.
- **Mantenible.** Se puede modificar fácilmente.

Probando las cualidades del software

- **Reusable.** La misma pieza de software se puede usar sin cambios en otro proyecto.
- **Portable.** Es ejecutable en distintos ambientes (SO, ...).
- **Legible.** El código es fácilmente interpretable.
- **Interoperable.** Puede interaccionar con otros software.

Most Important Test

- **Unitario.** Prueba un modulo lógico.
- **Integración.** Prueba un conjunto de módulos trabajando juntos.
- **Regresión.** Determina si los cambios recientes en un módulo afectan a otros módulos.
- **Humo.** Pruebas de integración completa, ejecutadas de forma periódica, que buscan encontrar errores en releases de forma temprana.

Most Important Test

- **Sistema.** Verificación de que el ingreso, procesado y recuperación de datos se hace de forma correcta.
- **Aceptación.** Determinación por parte del cliente de si acepta el modulo.
- **Stress.** Comprobación del funcionamiento del sistema ates condiciones adversas, como memoria baja, gran cantidad de accesos y concurrencia en transacciones.
- **Carga.** Tiempo de respuesta para las transacciones del sistema, para diferentes supuestos de carga.

Most Important Metrics

- **Complejidad ciclomática.** Numero de caminos independientes en el código.
- **Cobertura.** Cantidad de código fuente cubierto por test.
- **Código duplicado.** Mide las veces que aparecen un numero de líneas repetidas (> 10 líneas).
- **Comentarios.** Cantidad de código que tiene documentación.

Most Important Metrics

- **Diseño del software.** Indica el grado de acoplamiento de los módulos entre si.
- **Líneas.** Cantidad de líneas del código.
- **Malas prácticas de codificación.** Aparición de numero mágicos, bloques de try sin procesar, ...

Selenium

- Paquete de herramientas para automatizar pruebas de aplicaciones Web en dispositivos.

<http://seleniumhq.org/>

- La documentación la podremos obtener en

<http://seleniumhq.org/docs/index.html>

- Las herramientas que componen el paquete son
 - Selenium IDE.
 - Selenium Remote Control (RC) o selenium 1.
 - Selenium WebDriver o selenium 2.

Alternativas

- Existen numerosas alternativas en el mercado para automatizar la realización de pruebas sobre la UI Web, algunas de ellas son:
- TestComplete
- Ghost Inspector
- Imacros
- Telerik
- Testing Whiz
- HP Unified Functional Testing (UFT)
- Katalon Studio
- TestCraft

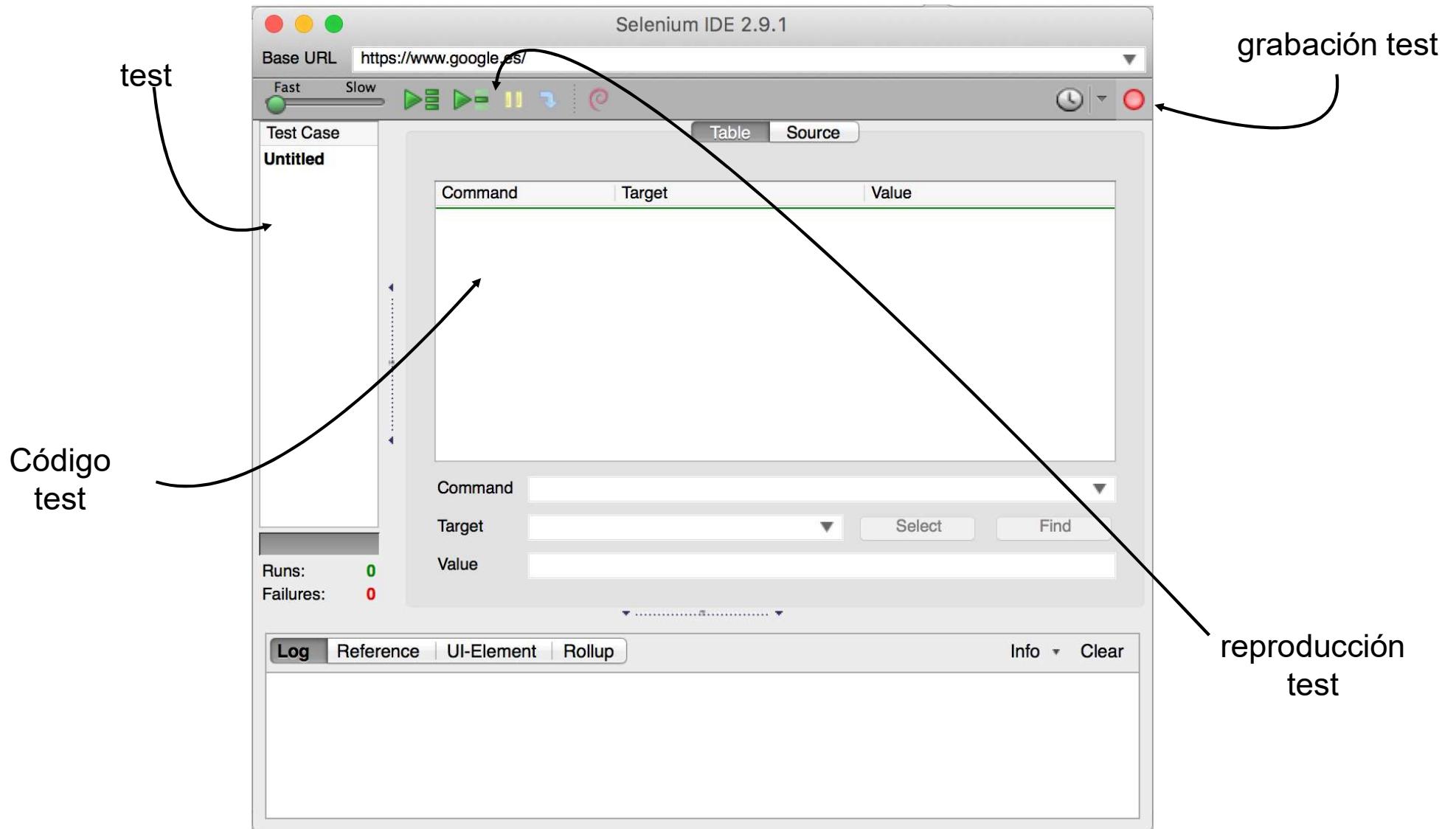
Selenium IDE

- Se trata de un plugin de Firefox, que nos permitirá grabar y reproducir una macro con una prueba funcional, la cual podremos repetir las veces que deseemos.

<https://addons.mozilla.org/es-es/firefox/addon/favorites-selenium-ide/>

- Las acciones que se realizan en la navegación mientras se graba la macro, se traducen en comandos.
- La macro por defecto se guarda en HTML, aunque también se puede obtener como java, c#, Python, ...
- También se podrán insertar validaciones y no solo acciones sobre la pagina.

Selenium IDE



Selenium IDE

- El HTML que se genera tiene una tabla con 3 columnas:

- Comando de Selenium.
- Primer parámetro requerido
- Segundo parámetro opcional

The screenshot shows the Selenium IDE interface with a table view of recorded commands. The table has three columns: Command, Target, and Value. The 'Table' tab is selected at the top. The data in the table is as follows:

Command	Target	Value
open	/search?q=wikipedia&ie=utf-8...	
clickAndWait	link=Wikipedia, la enciclopedia...	
type	id=searchInput	testing
clickAndWait	id=searchButton	

Below the table, there are input fields for Command, Target, and Value, each with a dropdown arrow and a 'Select' button. There is also a 'Find' button.

Selenium IDE

- Los comandos de selenium se dividen en tres tipos:
 - **Acciones**– Acciones sobre el navegador.
 - **Almacenamiento**– Almacenamiento en variables de valores intermedios.
 - **Aserciones**– Verificaciones del estado esperado del navegador.
- Los comandos de navegación mas habituales son:
 - **open**: abre una página empleando la URL.
 - **click/clickAndWait**: simula la acción de click, y opcionalmente espera a que una nueva pagina se cargue.
 - **waitForPageToLoad**: para la ejecución hasta que la pagina esperada se carga. Es llamada por defecto automáticamente cuando se invoca **clickAndWait**.

Selenium IDE

- Los comandos de navegación mas habituales son:
 - **waitForElementPresent**: para la ejecución hasta que el UIElement esperado, esta definido por un tag HTML presente en la pagina.
 - **chooseCancelOnNextConfirmation**: Predispone a seleccionar en la próxima ventana de confirmación el botón de Cancel.
- Los comandos de almacenamiento mas habituales son:
 - **store**: Almacena en la variable el valor.
 - **storeElementPresent**: Almacenara True o False, dependiendo de si encuentra el UI Element.
 - **storeText**: Almacena el texto encontrado. Es usado para localizar un texto en un lugar de la pagina especifico.

Selenium IDE

- Los comandos de verificación mas habituales son:
 - **verifyTitle/assertTitle**: verifica que el titulo de la pagina es el esperado.
 - **verifyTextPresent**: verifica que el texto esperado esta en alguna parte de la pagina.
 - **verifyElementPresent**: verifica que un UI element esperado, esta definido como tag HTML en la presente pagina.
 - **verifyText**: verifica si el texto esperado y su tag HTML están presentes en la pagina.
 - **assertAlert**: verifica si sale un alert con el texto esperado.
 - **assertConfirmation**: verifica si sale una ventana de confirmación con el texto esperado.

Ejercicio I

- Utilizando Selenium IDE, graba un test en el que:
 - Accedes a la página principal de la wikipedia.
 - Compruebas que el título de la página es el esperado

Ejercicio I

The screenshot shows the Selenium IDE interface running a test case named "testTitle". The test case contains three commands: "open /wiki/Wikipedia:Portada", "waitForTitle Wikipedia, la enciclopedia libre", and "assertTitle Wikipedia, la enciclopedia libre". The "Base URL" is set to "https://es.wikipedia.org/". A context menu is open over the page content, showing options like "Anterior", "Siguiente", "Recargar", "Añadir esta página a marcadores", "Guardar como...", "Ver imagen de fondo", "Seleccionar todo", "Ver código fuente de la página", "Ver información de la página", "Inspeccionar elemento", "Inspecionar elemento con Firebug", and "Show All Available Commands".

```
open /wiki/Wikipedia:Portada
assertTitle Wikipedia, la enciclopedia libre
assertValue
assertText id=content exact:Wikipedia:Portada De Wikipedia, la enciclopedia libre ...
assertTable
assertElementPresent id=content

verifyTitle Wikipedia, la enciclopedia libre
verifyValue
verifyText id=content exact:Wikipedia:Portada De Wikipedia, la enciclopedia libre S...
verifyTable
verifyElementPresent id=content

waitForTitle Wikipedia, la enciclopedia libre
waitForValue
waitForText id=content exact:Wikipedia:Portada De Wikipedia, la enciclopedia libre ...
waitForTable
waitForElementPresent id=content

storeTitle Wikipedia, la enciclopedia libre
storeValue
storeText id=content exact:Wikipedia:Portada De Wikipedia, la enciclopedia libre S...
storeTable
storeElementPresent id=content
```

testTitle (WikipediaSuitCase) - Selenium IDE 2.9.1 *

Base URL https://es.wikipedia.org/

Test Case testTitle *

testSearch register user

Command	Target	Value
open	/wiki/Wikipedia:Portada	
waitForTitle	Wikipedia, la enciclopedia libre	
assertTitle	Wikipedia, la enciclopedia libre	

Command open

Target /wiki/Wikipedia:Portada

Value

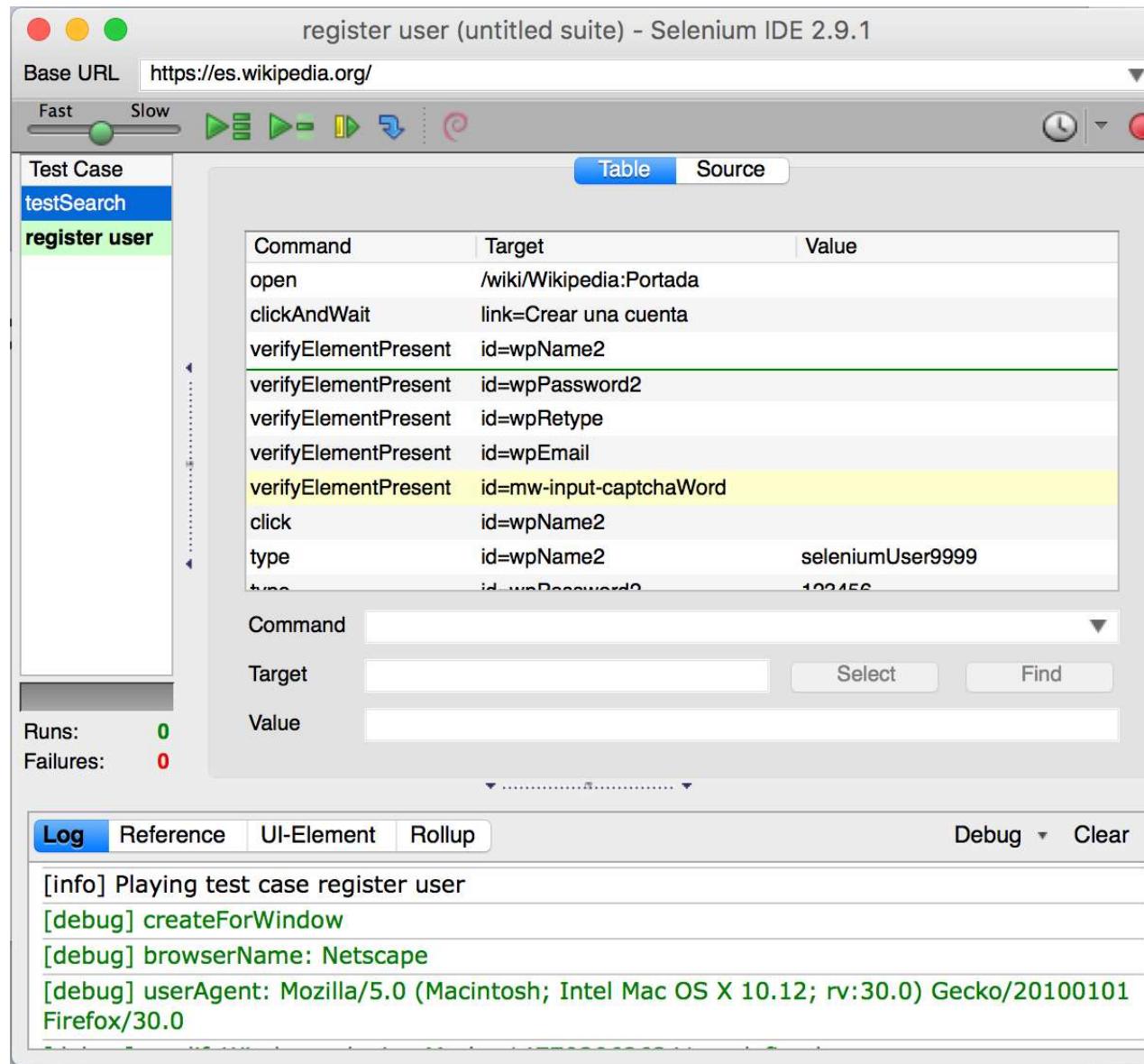
Runs: 1

Failures: 0

Ejercicio II

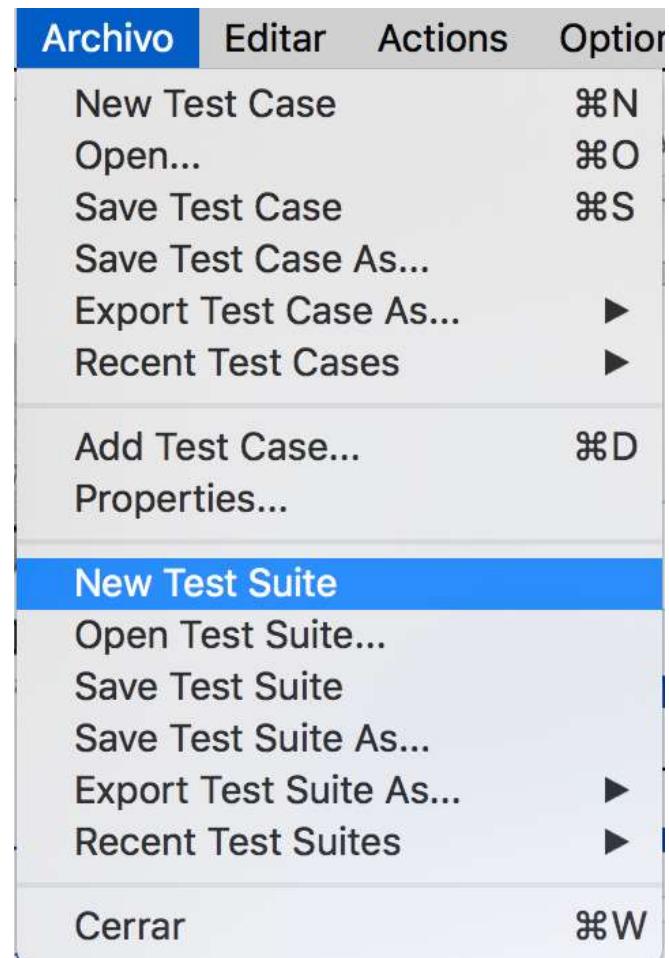
- Utilizando Selenium IDE, graba un test en el que:
 - Accedes a la página principal de la wikipedia.
 - Acceder a crear cuenta.
 - Comprobar que existen las cajas de usuario, contraseña, email y el captcha.
 - Rellenar todos los datos necesarios.
 - Intentar registrar el usuario.
 - Comprobar que aparece el error al haber llenado mal el captcha.

Ejercicio II



Selenium IDE

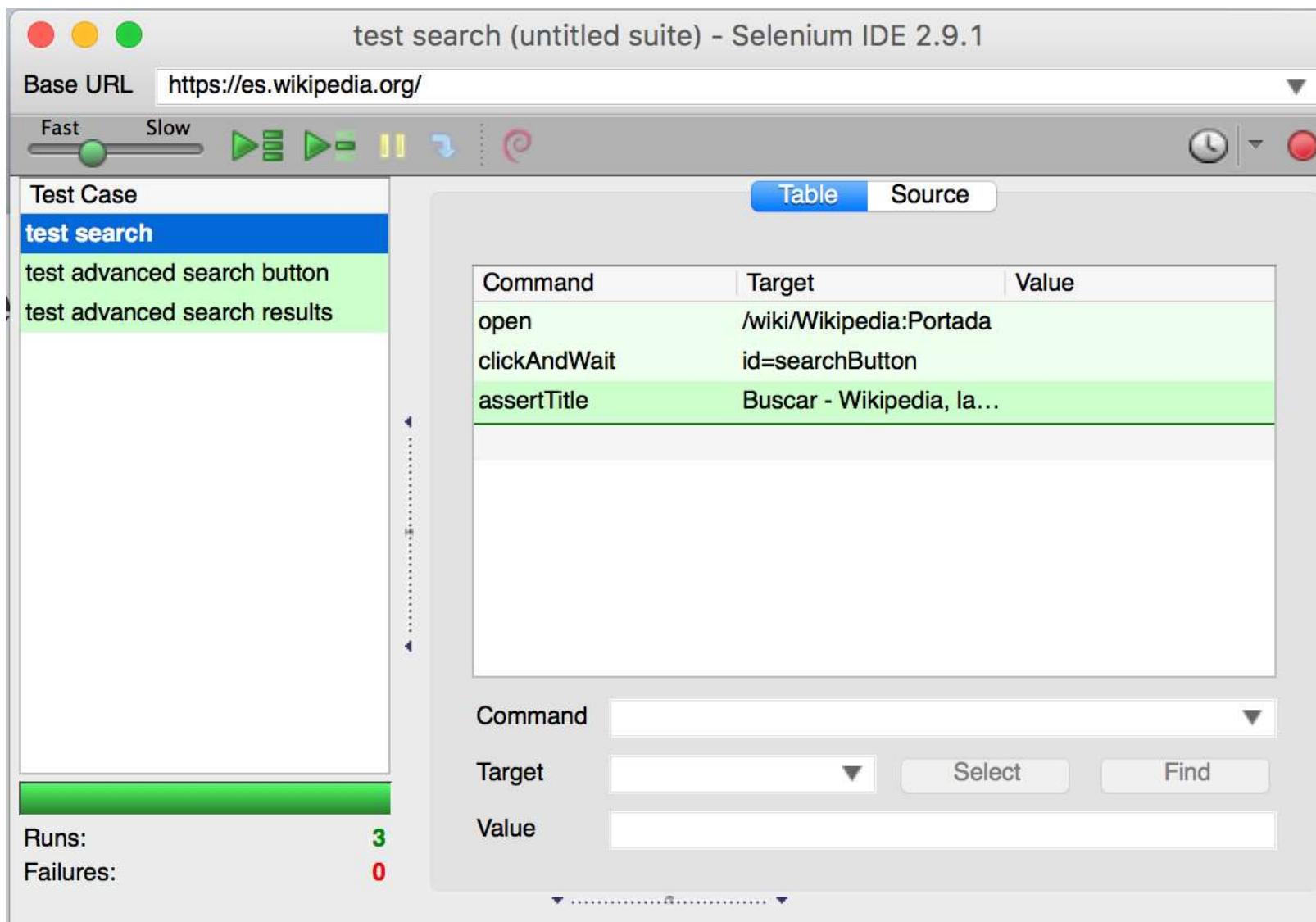
- Selenium nos permite generar un banco de pruebas, compuesto por los pequeños tests que hemos ido haciendo.
- Podemos añadir pruebas a nuestro banco.
- Ejecutarlas por separado o todas juntas.
- No es necesario que todas las pruebas sean independientes, pero hay que tener cuidado con los comandos “open”



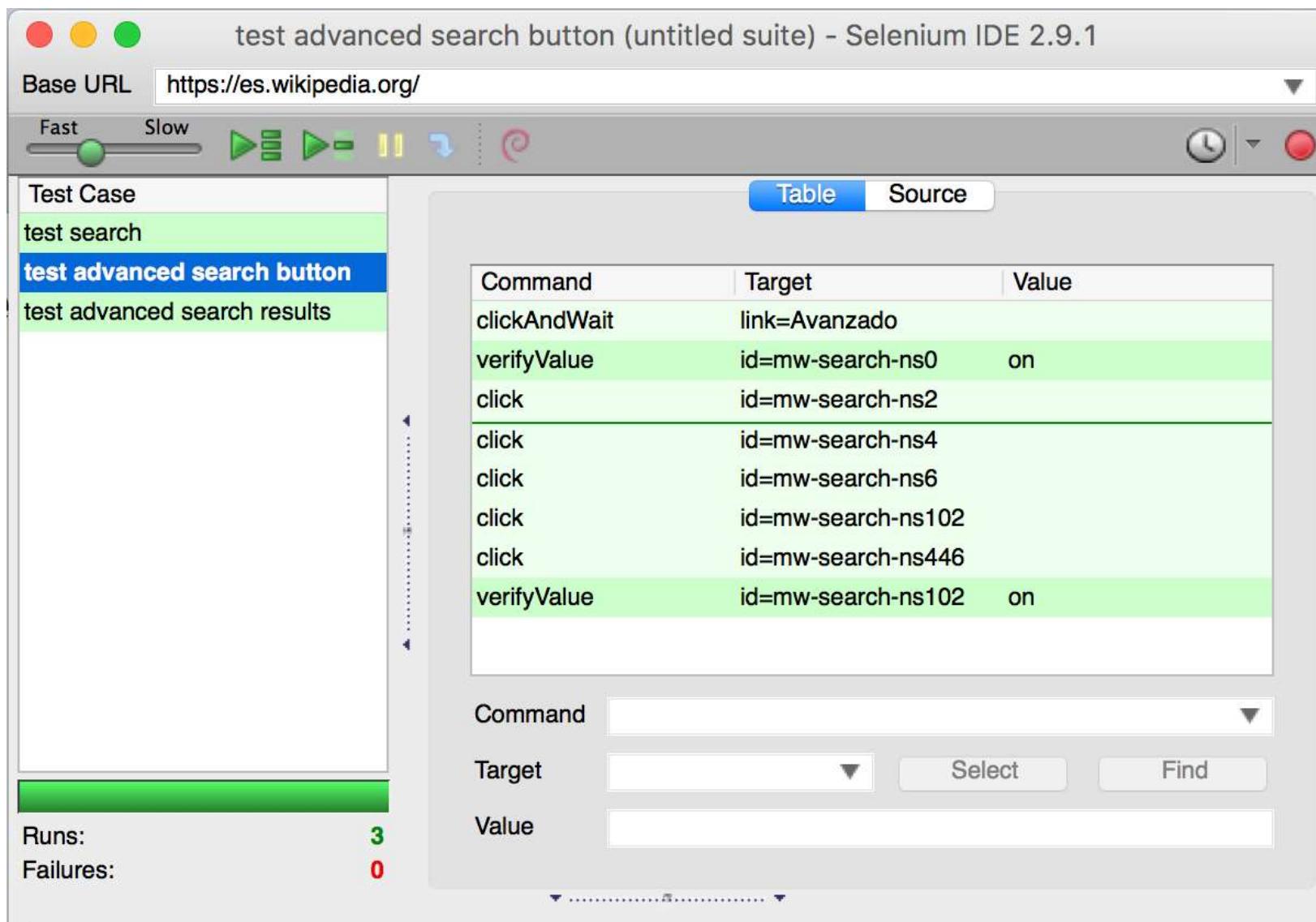
Ejercicio III

- Genera 3 test para testar las búsquedas avanzadas en wikipedia.
- **Primero:** carga la página de búsqueda avanzada y testa su título.
- **Segundo:** Click en el botón de “avanzado” y selecciona algún checkbox.
- **Tercero:** Escribe “selenium” en el cuadro de búsqueda y comprueba que devuelve mas de un resultado.

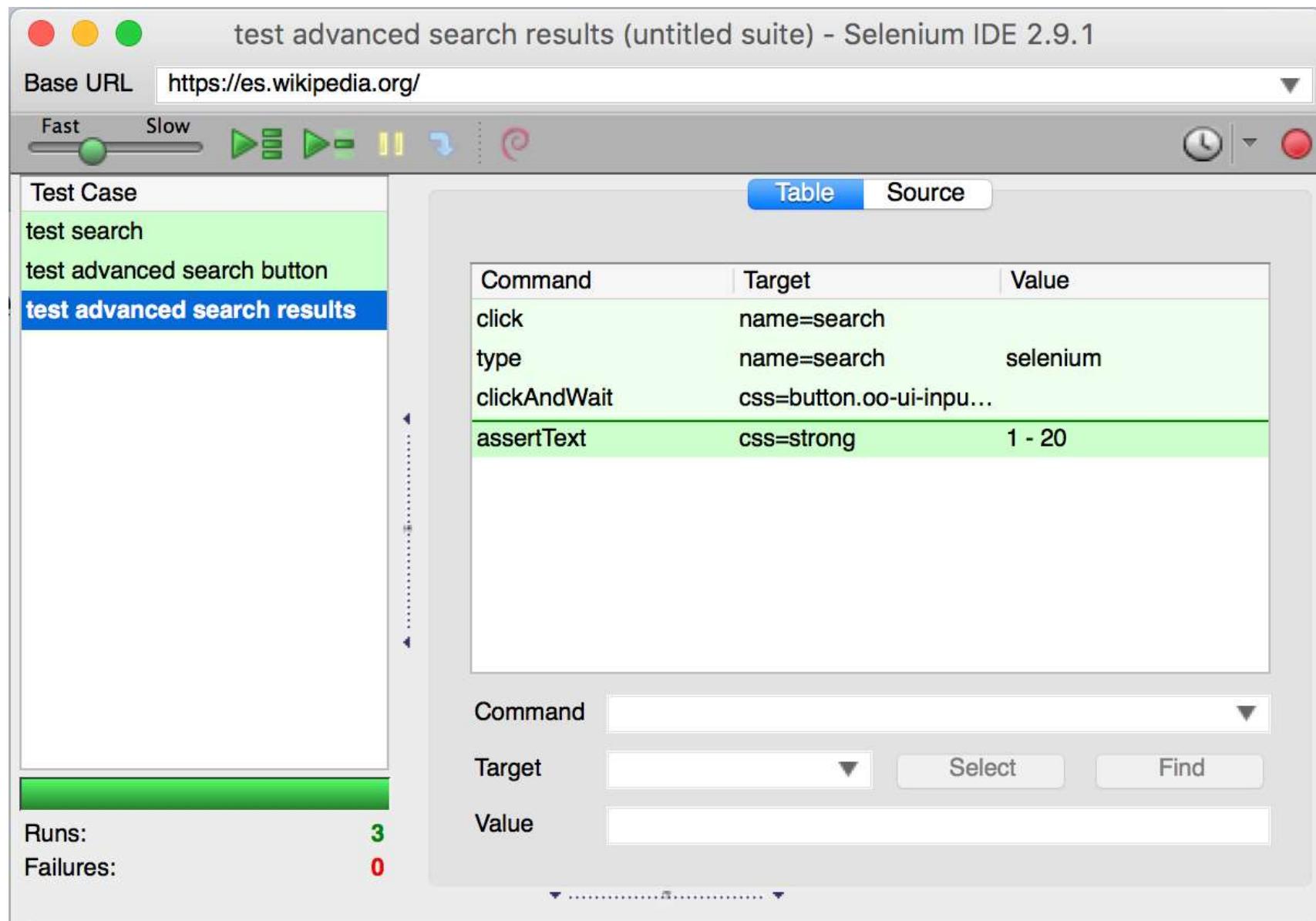
Ejercicio III



Ejercicio III



Ejercicio III

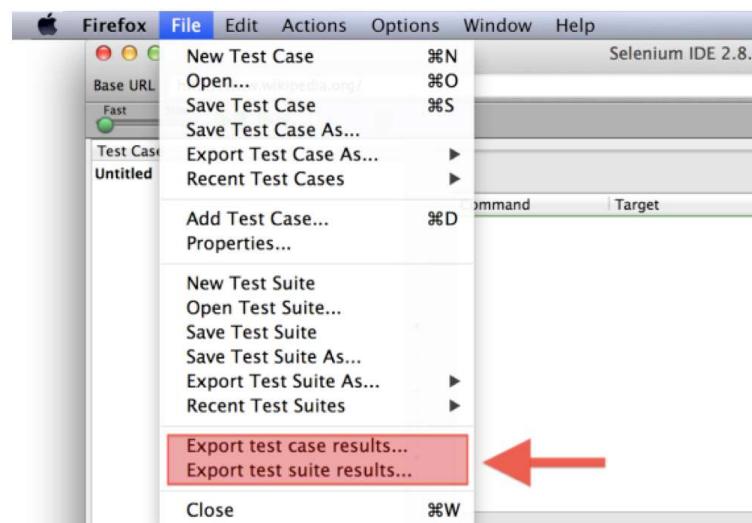


Selenium IDE

- Para la generación de informes con selenium IDE, necesitamos instalar un plugin a firefox



- Una vez instalado podemos generar automáticamente los informes de los tests



Selenium IDE

- Página HTML con los informes de los resultados

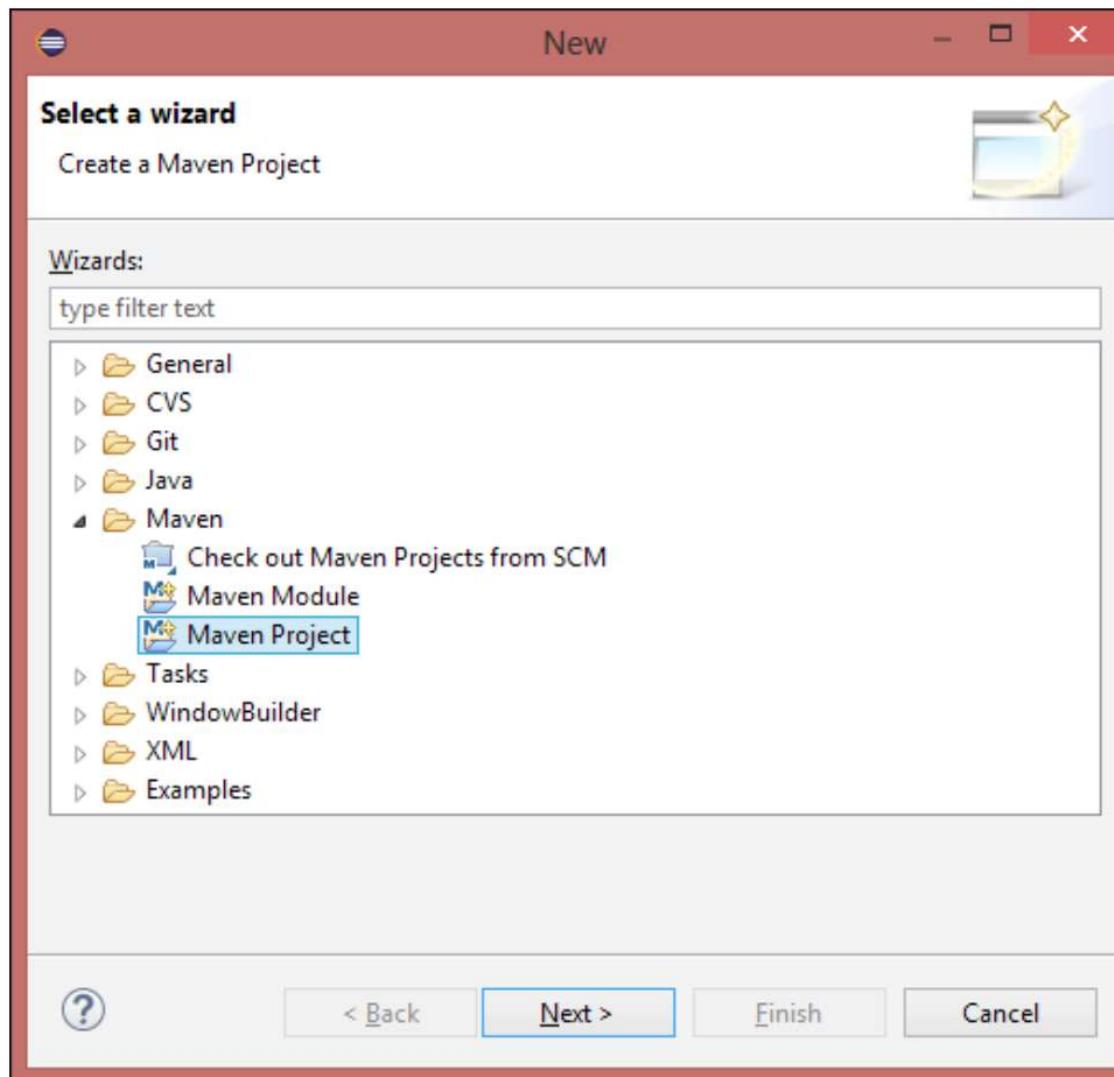
Test Suite Results		
Untitled		
Untitled 2		
open	/allPlans.action;jsessionid=m494ip3wu8hnne6bk5dwz804	
verifyText	css=h1	OpenQA Bamboo
verifyText	//table[@id='dashboard']/tbody[10]/tr/td[2]/a	Selenium IDE
open	/allPlans.action;jsessionid=m494ip3wu8hnne6bk5dwz804	
clickAndWait	//table[@id='dashboard']/tbody[10]/tr/td[2]/a	
verifyText	id=breadcrumb:IDE	Selenium IDE
clickAndWait	id=viewBuild:IDE-EDITOR	
verifyText	css=p.success	Plan Status: #177 was successful

Selenium Web Driver

- Selenium Web Driver es el motor de pruebas automatizadas de Selenium.
- Proporciona una API para poder realizar las pruebas y la automatización del los navegadores web.
- Necesitamos un entorno de desarrollo y algunas dependencias externas para poder construir un marco de pruebas con Selenium Web Driver.
- Para poder construir este marco de pruebas vamos a usar Eclipse junto con Maven.

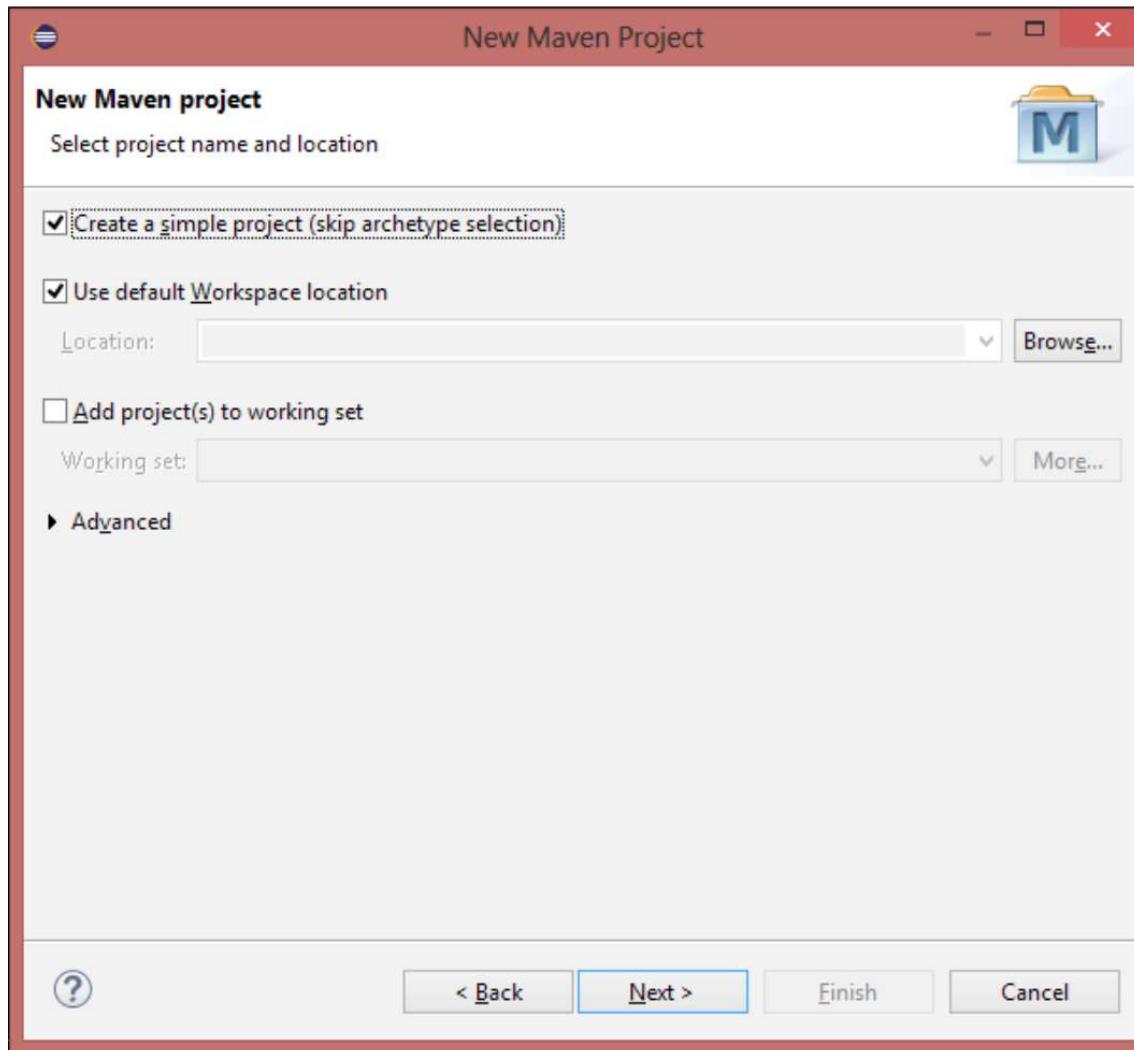
Selenium Web Driver

- Creamos un proyecto Maven con Eclipse



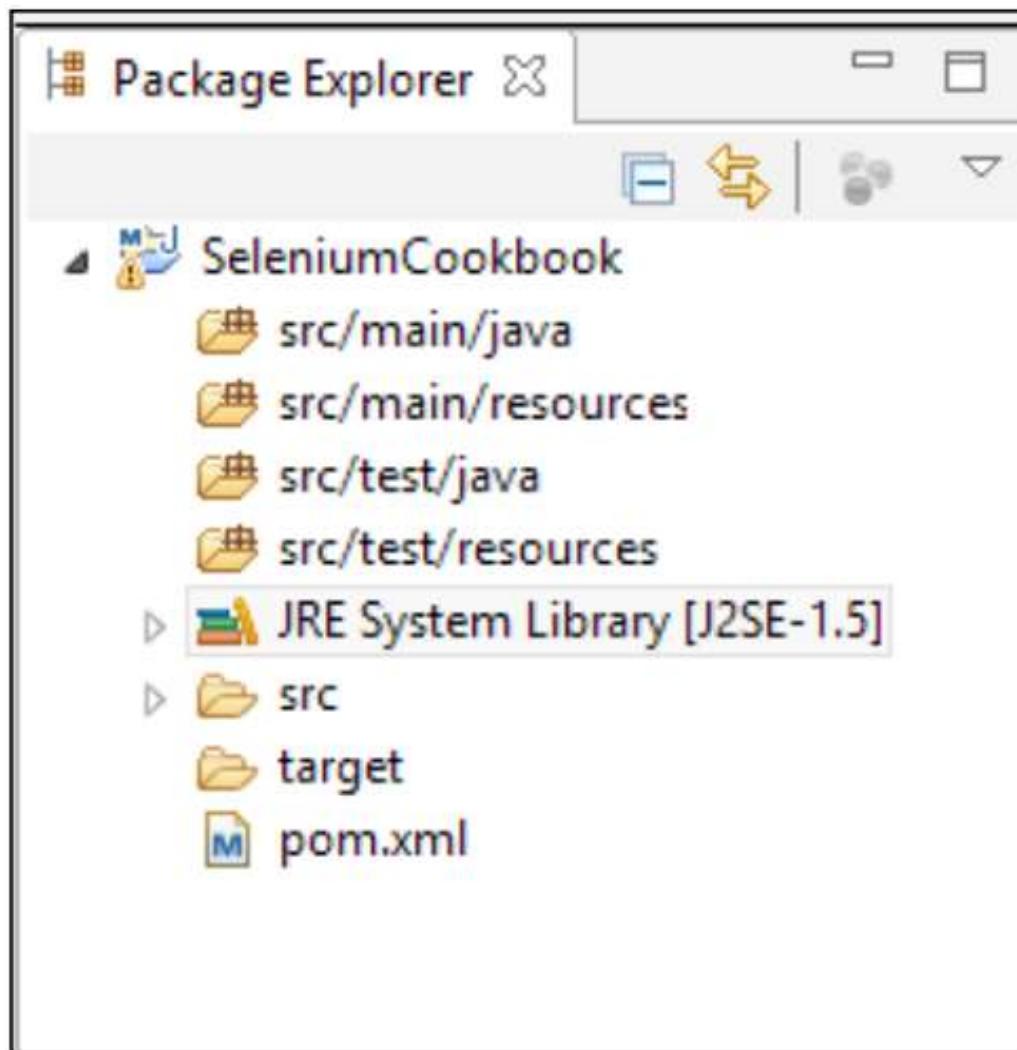
Selenium Web Driver

- Creamos un proyecto Maven con Eclipse



Selenium Web Driver

- Una vez añadidos el Group Id y el Artifact Id deberíamos tener un proyecto como este



Selenium Web Driver

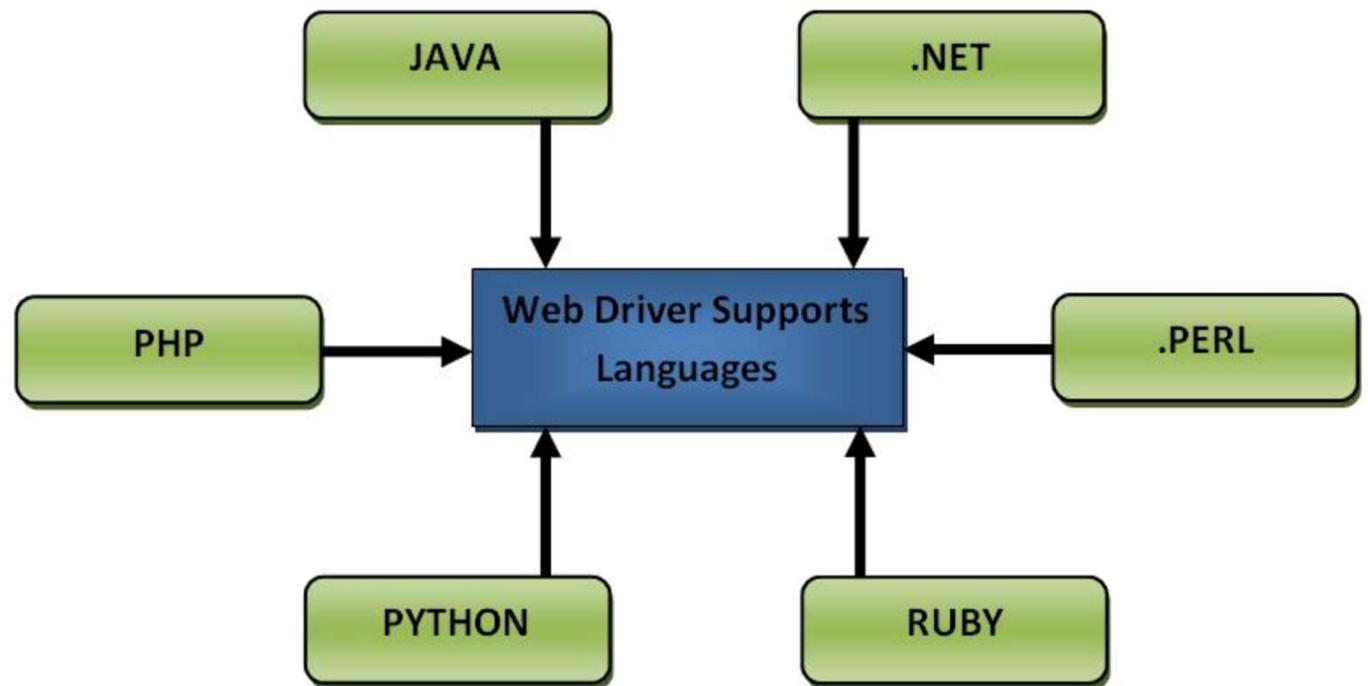
- Añadimos en el *pom.xml* las dependencias de *selenium* y *junit*

```
<dependencies>
    <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>2.53.0</version>
        <scope>test</scope>
    </dependency>

    <!-- https://mvnrepository.com/artifact/junit/junit -->
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

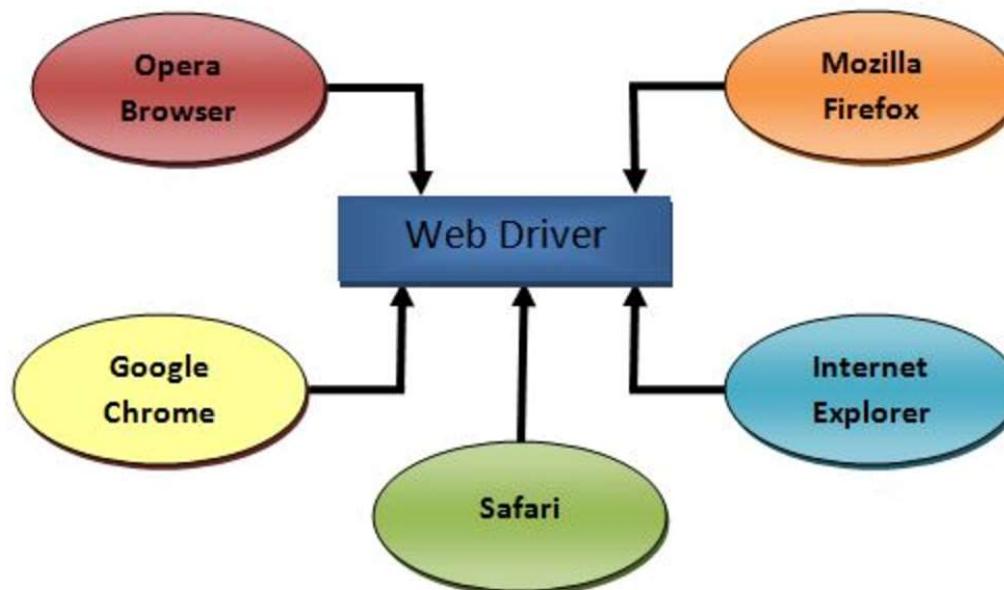
Objeto Web Driver

- Interfaz que nos permite crear instrucciones para automatizar un navegador Web.
- A través del objeto Web Driver podemos acceder a todo el contenido Web.
- Soporta varios lenguajes



Objeto Web Driver

- Selenium tiene implementado diferentes Web Drivers para dar soporte a distintos navegadores



Junit

- Framework para pruebas unitarias
 - Pruebas de funcionamiento de una clase
- Paquetes junit.* (JUnit 3) y org.junit.* (JUnit 4)
- Disponible en Eclipse (JUnit 3 y 4)
 - Eclipse nos proporciona la creación de Junit Test Case (caso de prueba) y Junit Test Suite (conjunto de casos de prueba).
- JUnit 4 admite timeout, excepciones esperadas, tests ignorables, ...

JUnit

- Paquetes de pruebas (Suites)
 - Permite ejecutar pruebas de varias clases en conjunto desde una lanzadora
 - En JUnit 3 se añaden clases a TestSuite
 - **suite.addTest(ClaseTest.suite());**
 - **suite.addTestSuite(ClaseTest.class);**
 - En JUnit 4 se usan anotaciones de clase
 - **@RunWith(Suite.class)**
 - **@SuiteClasses({C1Test.class, C2Test.class})**

Junit 4

- Antes de Java 5
 - Los métodos debían llamarse **testXXX()**
 - Se admitían los métodos **setUp()** y **tearDown()** para inicializar y liberar objetos (pre-post condiciones) antes de cada Test.
- Despues de Java 5
 - Se utilizan anotaciones **@Before**, **@Test**, **@After**, **@AfterClass** y **@BeforeClass**.
 - Aparecen **@AfterClass** y **@BeforeClass**, que son dos métodos que se ejecutarán una única vez antes de empezar las pruebas y justo después de terminarlas.
 - No importan los nombres de los métodos ni hay que heredar **TestCase**.

Junit 4

Anotación	Descripción
BeforeClass	Sólo puede haber un método con este marcador. Este método será invocado una vez al principio del lanzamiento de todas las pruebas. Se suele utilizar para inicializar los atributos comunes a todas las pruebas.
AfterClass	Sólo puede haber un método con este marcador. Este método será invocado una sola vez cuando finalicen todas las pruebas.
Before	Los métodos marcados con esta anotación, serán invocados antes de iniciarse cada prueba (antes de que se ejecute cada método).
After	Los métodos marcados con esta anotación, serán invocados después de ejecutarse cada prueba (después de que se ejecute cada método).
Test	Representa un test que se debe ejecutar. Puede tener dos tipos de modificadores: <code>Test(timeout=X)</code> que significa que el test será válido si se ejecuta en un tiempo máximo de X milisegundos. <code>Test(expected = java.util.NoSuchElementException.class)</code> . Que significa que el test será válido si se lanza una determinada excepción.
Ignore	Los métodos marcados con esta anotación no serán ejecutados. Se suelen poner para desactivar las pruebas que no pueden ser realizadas por algún motivo.

Testeando excepciones

- En JUnit 4, se proporciona un parámetro **expected**, junto con la anotación **@Test**, que permite indicar que en la ejecución de un test, se espera, es decir, el resultado bueno, es que se lance una excepción.

```
@Test(expected=InvalidIngresoException.class)  
public void comprobamosQueLanzamosException() throws InvalidIngresoException{  
}
```

Restricciones Temporales

- También JUnit 4, proporciona otro parámetro **timeout**, junto con la anotación **@Test**, que permite indicar el tiempo máximo que debe tardar la ejecución de un test.

```
@Test(timeout=12000)  
public void testDeRendimiento() {  
}
```

Test paramétricos

- JUnit 4 proporciona un mecanismo, para poder ejecutar un Test Case, con juegos de datos distintos, sin tener que codificar la prueba varias veces.
- Para emplear este mecanismo hay que
 - Anotar la clase de Test con

```
@RunWith(Parameterized.class)
```

Test paramétricos

- Crear un método estático público, con la siguiente firma, donde cada array de objetos, representa un juego de datos, y cada objeto del array es un dato de prueba, ya sea de entrada o esperado

```
@Parameters  
public static Collection<Object[]> data() {}
```

- Atributos de clase de la misma tipología que los objetos de los array retornados por el anterior método.
- Constructor con tantos parámetros como atributos de clase.

Asertos

- Tanto en JUnit 3 como en 4, se proporciona una clase con métodos estáticos, que permiten realizar comprobaciones (asertos) del estado actual de los datos que participan en un test.
- En JUnit 3

junit.framework.Assert

- En JUnit 4

org.junit.Assert

Asertos

- Entre los métodos que se encuentran en la clase están
 - **assertEquals**
 - **assertFalse**
 - **assertTrue**
 - **assertNotNull**
 - **assertNull**
 - **assertNotSame**
 - **assertSame**
 - **fail**

Objeto Web Driver

```
import static org.junit.Assert.assertEquals;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class SimpleTest {

    private WebDriver driver;

    @Before
    public void setUp() {

        // Launch a new Firefox instance
        driver = new FirefoxDriver();
        // Maximize the browser window
        driver.manage().window().maximize();
        // Navigate to Google
        driver.get("http://www.google.com");
    }

    @Test
    public void testGooglePageTitle() {

        assertEquals("Google", driver.getTitle());
    }

    @After
    public void tearDown() throws Exception {

        // Close the browser
        driver.quit();
    }
}
```

```
import static org.junit.Assert.assertEquals;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class SimpleTest {

    private WebDriver driver;

    @Before
    public void setUp() {

        System.setProperty("webdriver.chrome.driver", "/Path");

        // Launch a new Firefox instance
        driver = new ChromeDriver();
        // Maximize the browser window
        driver.manage().window().maximize();
        // Navigate to Google
        driver.get("http://www.google.com");
    }

    @Test
    public void testGooglePageTitle() {

        assertEquals("Google", driver.getTitle());
    }

    @After
    public void tearDown() throws Exception {

        // Close the browser
        driver.quit();
    }
}
```

Elementos Web

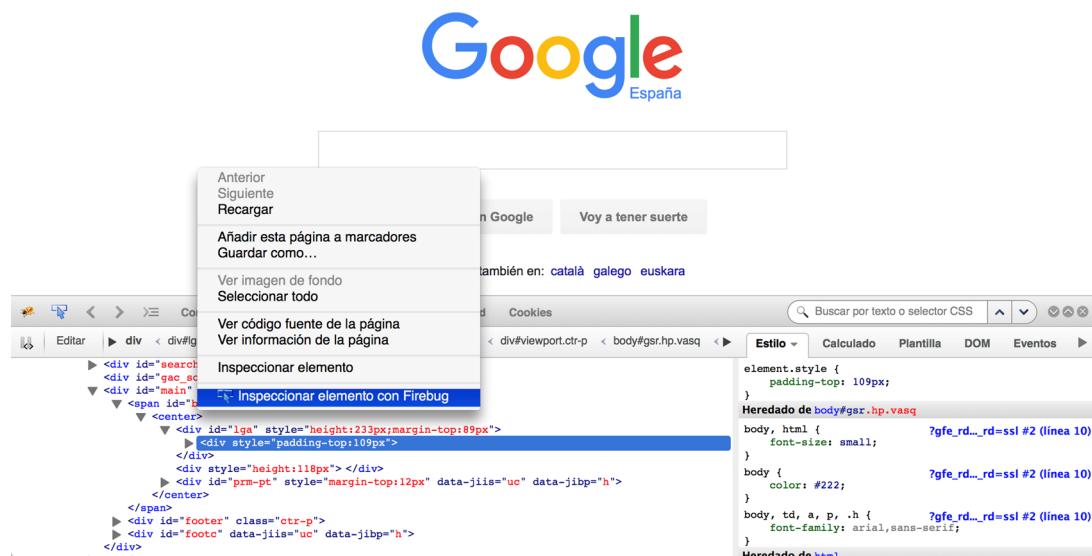
- Herramientas Web del Navegador para inspeccionar elementos.
- Elementos Web con findElement.
- Links.
- Elementos por tag name
- CSS selectors
- XPath
- jQuery selectors

Herramientas Web

- Para inspeccionar los elementos Web en Firefox vamos usar el plugin de firebug

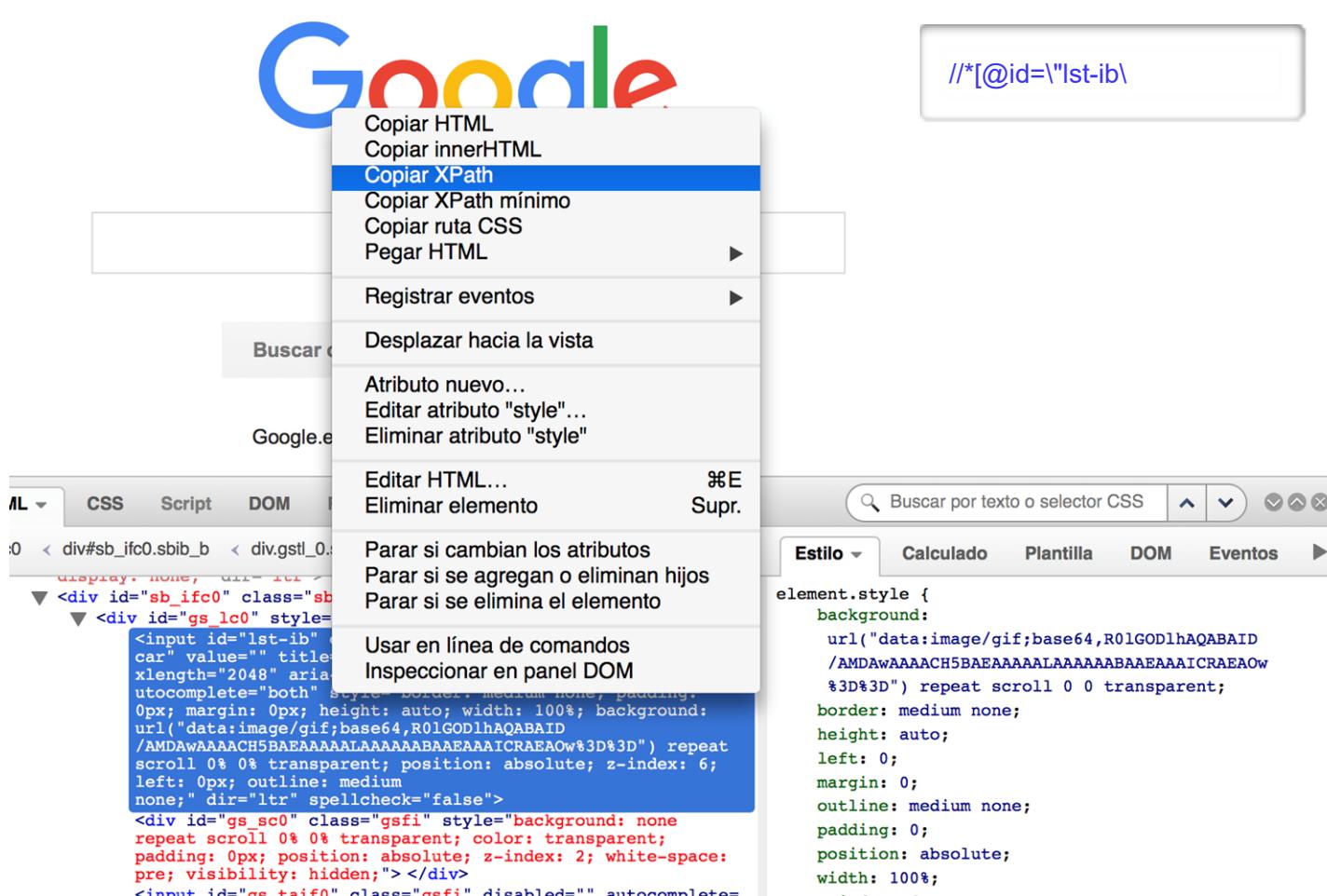


- Dentro de la web, inspeccionamos los elementos con el botón derecho



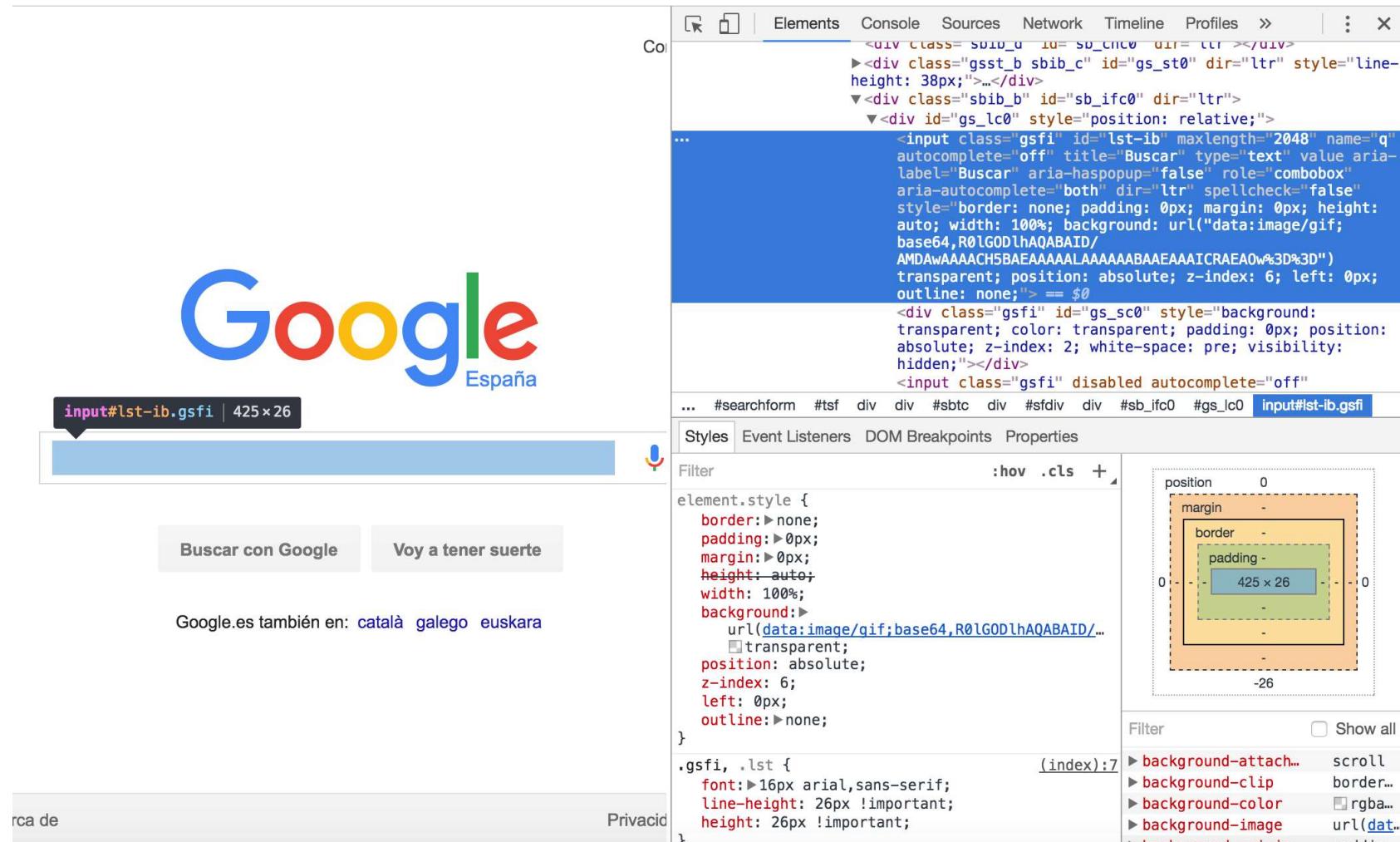
Herramientas Web

- Firebug nos proporciona varias maneras de poder acceder a la ruta de los elementos web, tanto su XPath como su CSS selector.



Herramientas Web

- Con Google chrome usamos las Chrome dev tools, nos permiten hacer prácticamente lo mismo que firebug.



Find Element

- Para acceder a los elementos de la página web lo hacemos a través de los métodos de las interfaces WebDriver y WebElement.
- Usamos los métodos findElement() y findElements()
- Estos métodos devuelven una instancia de un WebElement, solo si se ha encontrado según el criterio de búsqueda especificado.
- Si no se encuentra el WebElement, en el caso de findElement() se lanzará una excepción NoSuchElementException exception, en el de findElements() se devolverá una lista vacía.

Find Element

- Por atributo “id”. Podemos buscar cualquier elemento de la web que tenga su atributo id.
- Proporciona la estrategia de localización mas rápida.

```
<form name="loginForm">
    <label for="username">UserName: </label>
    <input type="text" id="username" /><br/>
    <label for="password">Password: </label>
    <input type="password" id="password" /><br/>
    <input name="login" type="submit" value="Login" />
</form>
```

```
WebElement username = driver.findElement(By.id("username"));
WebElement password = driver.findElement(By.id("password"));
```

Find Element

- No siempre todos los elementos de la web disponen de un atributo “id”.
- Podemos buscar también por el atributo “name”

```
<form name="loginForm">
    <label for="username">UserName: </label>
    <input type="text" name="username" /><br/>
    <label for="password">Password: </label>
    <input type="password" name="password" /><br/>
    <input name="login" type="submit" value="Login" />
</form>

WebElement username = driver.findElement(By.name("username"));
WebElement password = driver.findElement(By.name("password"));
```

Find Element

- También podemos buscar por su clase

```
<form name="loginForm">
    <label for="username">UserName: </label>
    <input type="text" class="username" /><br>
    <label for="password">Password: </label>
    <input type="password" class="password" /><br>
    <input name="login" type="submit" value="Login" />
</form>

WebElement username = driver.findElement(By.className("username"));
WebElement password = driver.findElement(By.className("password"));
```

Find Element

- Como la interface WebElement también soporta los métodos findElement, podemos buscar entre los hijos de un elemento web, filtrar los resultados.

```
WebElement div = driver.findElement(By.id("div1"));
WebElement topLink = div.findElement(By.linkText("top"));

WebElement topLink = driver.findElement(By.id("div1")).findElement(By.linkText("top"));
```

Ejercicio

- Abrir la pagina de la Wikipedia y buscar dentro del div con id “p-navigation”, los divs con ids “n-mainpage-description” y “n-portal”
- Comprobar sus valores “Portada” y “Portal de la comunidad”

Ejercicio

```
15 public class WikipediaNavigationBarTest {  
16  
17     private static WebDriver driver;  
18  
19     @BeforeClass  
20     public static void setUp() {  
21         //initialize web driver object  
22         //set https://es.wikipedia.org/ web page  
23     }  
24  
25  
26     @Test  
27     public void testMainPageNavigationBarTitles() {  
28         //get elements in p-navigation div by ID  
29         //test that elements exists  
30     }  
31  
32  
33     @AfterClass  
34     public static void tearDown() throws Exception {  
35         driver.quit();  
36     }  
37  
38  
39 }
```

Ejercicio

```
15 public class WikipediaNavigationBarTest {  
16  
17     private static WebDriver driver;  
18  
19     @BeforeClass  
20     public static void setUp() {  
21  
22         driver = new FirefoxDriver();  
23         driver.manage().window().maximize();  
24         driver.get("https://es.wikipedia.org/");  
25     }  
26  
27     @Test  
28     public void testMainPageNavigationBarTitles() {  
29  
30         WebElement portada = driver.findElement(By.id("n-mainpage-description"));  
31         WebElement portal = driver.findElement(By.id("n-portal"));  
32  
33         WebElement ayuda = driver.findElement(By.id("n-help"));  
34         WebElement donaciones = driver.findElement(By.id("n-sitesupport"));  
35  
36  
37         assertEquals(portada.getText(), "Portada");  
38         assertEquals(portal.getText(), "Portal de la comunidad");  
39  
40         assertEquals(ayuda.getText(), "Ayuda");  
41         assertEquals(donaciones.getText(), "Donaciones");  
42     }  
43  
44     @AfterClass  
45     public static void tearDown() throws Exception {  
46  
47         driver.quit();  
48     }  
49  
50 }
```

Find Element

- El WebDriver nos proporciona el método `findElements()`, lo usamos cuando necesitamos buscar mas de un elemento que coincide con nuestro criterio de búsqueda.
- Este método nos devuelve una lista de `WebElements`

```
List<WebElement> elements = driver.findElements(By.class("body"));
```

- Recorremos la lista de elementos pudiendo llamar a cualquier método del interface `WebElement`

```
for (WebElement element : elements) {  
    assertNotNull(element);  
}
```

Find Element

- Podemos buscar por el tag HTML del elemento, por ejemplo buscar todos los links (`<a>`) de una misma página.

```
List<WebElement> links = driver.findElements(By.tagName("a"));
```

- O todas las filas de una tabla

```
WebElement table = driver.findElement(By.id("summaryTable"));
List<WebElement> rows = table.findElements(By.tagName("tr"));
assertEquals(10, rows.size());
```

Ejercicio

- Abrir la pagina de la Wikipedia y buscar todos los nodos de lista ordenada dentro del div con id “p-navigation”.
- Comprobar que el valor es 9.

Ejercicio

```
15 public class WikipediaNavigationBarTest {  
16  
17     private static WebDriver driver;  
18  
19     @BeforeClass  
20     public static void setUp() {  
21  
22         driver = new FirefoxDriver();  
23         driver.manage().window().maximize();  
24         driver.get("https://es.wikipedia.org/");  
25     }  
26  
27     @Test  
28     public void testMainPageNavigationBarLinks() {  
29  
30         //find all the elements in p-navigation  
31         //test the size  
32     }  
33  
34     @AfterClass  
35     public static void tearDown() throws Exception {  
36  
37         driver.quit();  
38     }  
39  
40 }
```

Ejercicio

```
15 public class WikipediaNavigationBarTest {  
16  
17     private static WebDriver driver;  
18  
19     @BeforeClass  
20     public static void setUp() {  
21  
22         driver = new FirefoxDriver();  
23         driver.manage().window().maximize();  
24         driver.get("https://es.wikipedia.org/");  
25     }  
26  
27     @Test  
28     public void test MainPageNavigationBarLinks() {  
29  
30         List<WebElement> elements = driver.findElement(By.id("p-navigation"))  
31             .findElement(By.className("body"))  
32             .findElements(By.tagName("li"));  
33  
34         for (WebElement element : elements) {  
35             assertNotNull(element);  
36         }  
37  
38         assertEquals(elements.size(), 9); //8 + 1 for one bug  
39     }  
40  
41     @AfterClass  
42     public static void tearDown() throws Exception {  
43  
44         driver.quit();  
45     }  
46  
47 }
```

Find Element

- Selenium proporciona unos métodos específicos para poder buscar links dentro de una página web.
- Buscar link por su texto
 - Buscamos un link según el texto que se muestra en la página mediante el método linkText() de la clase By.

```
WebElement gmailLink = driver.findElement(By.linkText("Gmail"));
assertEquals("http://mail.google.com/", gmailLink.getAttribute("href"));
```

- Buscamos un link según un texto parcial mediante partialLinkText(), puede resultar muy útil cuando el texto cambia, como por ejemplo el número de resultados de una búsqueda.

```
WebElement inboxLink = driver.findElement(By.partialLinkText("Recibidos"));
```

Ejercicio

```
--  
17 public class WikipediaMainPageLinkTest {  
18  
19     private static WebDriver driver;  
20  
21     @BeforeClass  
22     public static void setUp() {  
23  
24         driver = new FirefoxDriver();  
25         driver.manage().window().maximize();  
26         driver.get("https://es.wikipedia.org/");  
27     }  
28  
29     @Test  
30     public void testNavigationMenuLinks() {  
31  
32         //find all link in p-personal div and test it  
33     }  
34  
35     @Test  
36     public void testDateLinks() {  
37  
38         //find some dynamic text links with dates and test it  
39     }  
40  
41  
42     @AfterClass  
43     public static void tearDown() throws Exception {  
44  
45         driver.quit();  
46     }  
47
```

Ejercicio

```
17 public class WikipediaMainPageLinkTest {
18
19     private static WebDriver driver;
20
21     @BeforeClass
22     public static void setUp() {
23
24         driver = new FirefoxDriver();
25         driver.manage().window().maximize();
26         driver.get("https://es.wikipedia.org/");
27     }
28
29     @Test
30     public void testNavigationMenuLinks() {
31
32         //find all link in p-personal div and test it
33
34         WebElement discussion = driver.findElement(By.linkText("Discusión"));
35         WebElement contributions = driver.findElement(By.linkText("Contribuciones"));
36         WebElement account = driver.findElement(By.linkText("Crear una cuenta"));
37         WebElement access = driver.findElement(By.linkText("Acceder"));
38
39         assertNotNull(discussion);
40         assertNotNull(contributions);
41         assertNotNull(account);
42         assertNotNull(access);
43
44     }
45
46     @Test
47     public void testDateLinks() {
48
49         //find some dynamic text links with dates and test it
50
51         WebElement month = driver.findElement(By.partialLinkText(currentMonthName()));
52         WebElement week = driver.findElement(By.partialLinkText(currentMonthName()));
53
54         assertNotNull(month);
55         assertNotNull(week);
56     }
57
58
59     @AfterClass
60     public static void tearDown() throws Exception {
61
62         driver.quit();
63     }
64 }
```

Find Element, XPath

- Lenguaje que permite construir expresiones que recorren y procesan un documento xml.
- La mayor diferencia de XPath con CSS, es que nos permite navegar en todas direcciones entre el xml; padres-hijos hijos-padres.
- Terminología:
 - **Nodo**: Cada elemento del xml.
 - <html> root node
 - <title> element node
 - id="identifier" attribute or value node
 - **Atomic value**: Nodos sin hijos ni padres (texto de un nodo).
 - **Padres**: Cada element node tiene un parent node que “cuelga”.
 - **Hijo**: Nodos que pertenecen a un mismo parent node.
 - **Siblings**: Nodos que están al mismo nivel (hermanos).

Find Element, XPath

- XPath usa una serie de expresiones para poder acceder a un elemento determinado
 - ***nodename***: selecciona todos los nodos dado el nombre.
 - **/**: si aparece el primero indica la ruta absoluta /html. Si aparece en medio indica la relación padre-hijo entre los nodos html/body/table
 - **//**: indica la ruta relativa al nodo. //table, //a//img (todos los elementos img dentro de a)
 - **.** : representa el nodo de la posición actual
 - **..** : representa el nodo padre de la posición actual
 - **@** : se utiliza para indicar un atributo //img/@alt (todas las img con atributo alt)

Find Element, XPath

- Ruta absoluta. Indicamos la ruta completa del nodo al que queremos acceder.
- Esta estrategia tiene sus limitaciones, si cambia la posición del elemento, no encontraremos el nodo indicado.

```
WebElement userName = driver.findElement(By.xpath("/html/body/div/div/input"));
```

- Ruta relativa. Indicamos la ruta relativa del elemento independientemente de dónde se encuentre dentro del DOM.

```
WebElement logo = driver.findElement(By.xpath("//*[@id='p-logo']"));
```

- Predicados

```
WebElement userName = driver.findElement(By.xpath("/html/body/div[4]/div[2]/div[1]"));
```

Ejercicio

```
12
13 public class NavigationBarMenuTitleTest {
14
15     private static WebDriver driver;
16
17     @BeforeClass
18     public static void setUp() {
19
20         driver = new FirefoxDriver();
21         driver.manage().window().maximize();
22         driver.get("https://es.wikipedia.org/");
23     }
24
25     @Test
26     public void testElementsWithAbsolutePath() {
27
28         //find all titles in mw-panel using absolute XPATH
29     }
30
31
32     @Test
33     public void testElementsWithRelativePath() {
34
35         //find all titles in mw-panel using relative XPath
36     }
37
38
39     @AfterClass
40     public static void tearDown() throws Exception {
41
42         driver.quit();
43     }
44 }
```

Ejercicio

```
13 public class NavigationBarMenuTitleTest {
14
15     private static WebDriver driver;
16
17     @BeforeClass
18     public static void setUp() {
19
20         driver = new FirefoxDriver();
21         driver.manage().window().maximize();
22         driver.get("https://es.wikipedia.org/");
23     }
24
25     @Test
26     public void testElementsWithAbsolutePath() {
27
28         WebElement print = driver.findElement(By.xpath("/html/body/div[4]/div[2]/div[3]/h3"));
29         WebElement otherProjects = driver.findElement(By.xpath("/html/body/div[4]/div[2]/div[4]/h3"));
30         WebElement tools = driver.findElement(By.xpath("/html/body/div[4]/div[2]/div[5]/h3"));
31         WebElement languages = driver.findElement(By.xpath("/html/body/div[4]/div[2]/div[6]/h3"));
32
33         assertNotNull(print);
34         assertNotNull(otherProjects);
35         assertNotNull(tools);
36         assertNotNull(languages);
37     }
38
39
40     @Test
41     public void testElementsWithRelativePath() {
42
43         WebElement print = driver.findElement(By.xpath("//*[@id=\"p-coll-print-export-label\"]"));
44         WebElement otherProjects = driver.findElement(By.xpath("//*[@id=\"p-wikibase-otherprojects-label\"]"));
45         WebElement tools = driver.findElement(By.xpath("//*[@id=\"p-tb-label\"]"));
46         WebElement languages = driver.findElement(By.xpath("//*[@id=\"p-lang-label\"]"));
47
48         assertNotNull(print);
49         assertNotNull(otherProjects);
50         assertNotNull(tools);
51         assertNotNull(languages);
52     }
53
54
55     @AfterClass
56     public static void tearDown() throws Exception {
57
58         driver.quit();
59     }
60 }
```

Find Element, XPath

- Funciones en XPath

```
WebElement previousButton = driver.findElement(By.xpath("//input[@type='submit' and @value='Login']"));
WebElement previousButton = driver.findElement(By.xpath("//input[@type='submit' or @value='Login']"));
List<WebElement> imagesWithAlt = driver.findElements(By.xpath("//img[not(@alt)]"));
```

Syntax	Example	Description
starts-with()	input [starts-with(@id, 'ctrl')]	Starting with: For example, if the ID of an element is <code>ctrl_12</code> , this will find and return elements with <code>ctrl</code> at the beginning of the ID.
ends-with()	input [ends-with(@id, '_userName')]	Ending with: For example, if the ID of an element is <code>a_1_userName</code> , this will find and return elements with <code>_userName</code> at the end of the ID.
contains()	Input [contains(@id, 'userName')]	Containing: For example, if the ID for an element is <code>panel_login_userName_textfield</code> , this will use the <code>userName</code> part in the middle to match and locate the element.

Find Element, XPath

- Usando un valor determinado

```
WebElement userName = driver.findElement(By.xpath("//input[@*='username']"));
```

Expression	Description
/table/tr[1]	This will select the first <code>tr</code> (row) element that is the child of the <code>table</code> element.
/table/tr[last()]	This will select the last <code>tr</code> (row) element that is the child of the <code>table</code> element.
/table/tr[last()-1]	This will select the second last <code>tr</code> (row) element that is the child of the <code>table</code> element.
/table/tr[position()>4]	This will select the three <code>tr</code> (rows) elements that are child of the <code>table</code> element.
//tr[td>40]	This will select all the <code>tr</code> (rows) elements that have one of their children <code>td</code> with value greater than 40.

Ejercicio

```
11  
18 private static WebDriver driver;  
19  
20@  
20@BeforeClass  
21 public static void setUp() {  
22  
23     driver = new FirefoxDriver();  
24     driver.manage().window().maximize();  
25     driver.get("http://www.w3schools.com/html/html_tables.asp");  
26 }  
27  
28  
29@  
29@Test  
30 public void testTable() {  
31  
32     //find the customers table and test its elements  
33     //table not null  
34     //table has contents  
35     //the table is full of content  
36 }  
37  
38  
39@  
39@AfterClass  
40 public static void tearDown() throws Exception {  
41     driver.quit();  
42 }  
43 }  
44 }
```

Ejercicio

```
15 public class TableTest {  
16  
17  
18     private static WebDriver driver;  
19  
20     @BeforeClass  
21     public static void setUp() {  
22  
23         driver = new FirefoxDriver();  
24         driver.manage().window().maximize();  
25         driver.get("http://www.w3schools.com/html/html_tables.asp");  
26     }  
27  
28     @Test  
29     public void testTable() {  
30  
31         WebElement table = driver.findElement(By.xpath("//*[@id='customers']"));  
32         List<WebElement> rows = table.findElements(By.tagName("tr"));  
33         List<WebElement> lastRowColumns = driver.findElements(By.xpath("//*[@id='customers']/tbody/tr[last()]/td"));  
34         List<WebElement> twoLastRows = driver.findElements(By.xpath("//*[@id='customers']/tbody/tr[position()>5]"));  
35  
36         List<WebElement> tableElements = table.findElements(By.tagName("td"));  
37  
38         assertNotNull(table);  
39         assertEquals(rows.size(), 7);  
40         assertEquals(lastRowColumns.size(), 3);  
41         assertEquals(twoLastRows.size(), 2);  
42  
43         //check all table are fill  
44         for (WebElement element : tableElements) {  
45             assertFalse(element.getText().length() == 0);  
46         }  
47     }  
48  
49  
50     @AfterClass  
51     public static void tearDown() throws Exception {  
52         driver.quit();  
53     }  
54 }  
55 }
```

Find Element, CSS

- CSS es un lenguaje de estilo que se usa para describir la presentación de un elemento en HTML o XML.
- Indica que patrón de estilo se debe aplicar a un determinado elemento web.
- ruta absoluta

```
WebElement userName = driver.findElement(By.cssSelector("html body div div form input"));
WebElement userName = driver.findElement(By.cssSelector("html > body > div > div > form > input"));
```

- ruta relativa (teniendo en cuenta que devolverá el primer elemento)

```
WebElement userName = driver.findElement(By.cssSelector("input"));
```

Find Element, CSS

- usando su clase

```
WebElement loginButton = driver.findElement(By.cssSelector("input.login"));
WebElement loginButton = driver.findElement(By.cssSelector("input.login.textfield"));
WebElement loginButton = driver.findElement(By.cssSelector(".login"));
```

- usando su atributo id

```
WebElement userName = driver.findElement(By.cssSelector("input#username"));
WebElement userName = driver.findElement(By.cssSelector("#username"));
```

- otros atributos

```
WebElement userName = driver.findElement(By.cssSelector("input[name=username]"));
WebElement previousButton = driver.findElement(By.cssSelector("img[alt='Previous']"));
```

Find Element, CSS

- expresiones regulares en atributos

Syntax	Example	Description
<code>^=</code>	<code>input [id^='ctrl']</code>	Starting with: For example, if the ID of an element is <code>ctrl_12</code> , this will find and return elements with <code>ctrl</code> at the beginning of the ID.
<code>\$=</code>	<code>input [id\$='_userName']</code>	Ending with: For example, if the ID for an element is <code>a_1_userName</code> , this will find and return elements with <code>_userName</code> at the end of the ID.
<code>*=</code>	<code>input [id*= 'userName']</code>	Containing: For example, if the ID of an element is <code>panel_login_userName_textfield</code> , this will use the <code>userName</code> part in the middle to match and find the element.

Ejercicio

```
17 public class AdvancedSearchTest {  
18     private static WebDriver driver;  
20  
21     @BeforeClass  
22     public static void setUp() {  
23         driver = new FirefoxDriver();  
25         driver.manage().window().maximize();  
26         driver.get("https://es.wikipedia.org/w/index.php?title=Especial:Buscar&search=&fulltext=Buscar&profile=advanced");  
27     }  
28  
29  
30     @Test  
31     public void findAdvancedSearchComponentsWithCSSselectors() {  
32         //find all checkbox inside advanced search ( 30 )  
34         //find the search input and the search button  
35     }  
36  
37  
38     @AfterClass  
39     public static void tearDown() throws Exception {  
40         driver.quit();  
41     }  
42  
43 }
```

Ejercicio

```
16
17 public class AdvancedSearchTest {
18
19     private static WebDriver driver;
20
21     @BeforeClass
22     public static void setUp() {
23
24         driver = new FirefoxDriver();
25         driver.manage().window().maximize();
26         driver.get("https://es.wikipedia.org/w/index.php?title=Especial:Buscar&search=&fulltext=Buscar&profile=advanced");
27     }
28
29
30     @Test
31     public void findAdvancedSearchComponentsWithCSSselectors() {
32
33         List<WebElement> checkBoxes = driver.findElements(By.cssSelector("#mw-searchoptions input[name^='ns']"));
34         WebElement searchBox = driver.findElement(By.cssSelector("#searchText .oo-ui-inputWidget-input"));
35         WebElement searchButton = driver.findElement(By.cssSelector("button[type='submit'][role='button']"));
36
37         assertEquals(checkBoxes.size(), 30);
38         assertNotNull(searchBox);
39         assertNotNull(searchButton);
40     }
41
42
43
44     @AfterClass
45     public static void tearDown() throws Exception {
46         driver.quit();
47     }
48
49 }
```

Find Element, CSS

- Características avanzadas
 - selectores CSS para búsquedas de hijos

```
WebElement userName = driver.findElement(By.cssSelector("form#loginForm :nth-child(2)"));
```

Pseudo-class	Example	Description
:first-child	form#loginForm :first-child	This will find the first element under the form, that is, the label for username.
:last-child	form#loginForm :last-child	This will find the last element under the form, that is, the Login button.
:nth-child(2)	form#loginForm :nth-child(2)	This will find the second child element under the form, that is, the Username field.

- selectores CSS para búsquedas de hermanos

```
WebElement productDescription = driver.findElement(By.cssSelector("div#top5 > p + p"));
```

p + p	div#top5 > p + p	Immediately following sibling. This will locate Description for Product 2.
p + * + p	div#top5 > p + * + p	Following sibling with one intermediary. This will locate Description for Product 3.

Find Element, CSS

- Características avanzadas
 - selectores CSS pseudo-clases

```
WebElement productDescription = driver.findElement(By.cssSelector("input:focus"));
```

Pseudo-class	Example	Description
:enabled	input :enabled	This will find all the elements that are enabled for user input.
:disabled	input :enabled	This will find all the elements that are disabled for user input.
:checked	input :checked	This will find all the elements (checkboxes) that are checked.

Interacciones

- Cajas de texto y botones
- Testando textos y atributos CSS
- Desplegables y listas
- Botones de opciones (radio buttons)
- Checkboxes

Interacciones, Cajas de texto

- Para limpiar el texto utilizamos la función clear() del WebElement.
- Para introducir texto usamos sendKeys()
- Para enviar el formulario submit()

```
WebElement element = driver.findElement(By.name("q"));
element.clear();
element.sendKeys("Selenium testing tools cookbook");
element.submit();
```

- También podemos buscar el botón de enviar y hacer click() sobre él

```
WebElement element = driver.findElement(By.name("btnG"));
element.click();
```

Ejercicio

```
--  
13 public class SimpleSearchTest {  
14  
15     private static WebDriver driver;  
16  
17     @BeforeClass  
18     public static void setUp() {  
19  
20         driver = new FirefoxDriver();  
21         driver.manage().window().maximize();  
22         driver.get("https://es.wikipedia.org/wiki/Wikipedia:Portada");  
23     }  
24  
25     @Test  
26     public void testSimpleSearch() {  
27  
28         //find the search text box, fill with 'Selenium' and search  
29         //test the result  
30     }  
31  
32  
33     @AfterClass  
34     public static void tearDown() throws Exception {  
35         driver.quit();  
36     }  
37 }
```

Ejercicio

```
13 public class SimpleSearchTest {  
14  
15     private static WebDriver driver;  
16  
17     @BeforeClass  
18     public static void setUp() {  
19  
20         driver = new FirefoxDriver();  
21         driver.manage().window().maximize();  
22         driver.get("https://es.wikipedia.org/wiki/Wikipedia:Portada");  
23     }  
24  
25     @Test  
26     public void testSimpleSearch() {  
27  
28         WebElement searchTextbox = driver.findElement(By.cssSelector("#searchInput"));  
29  
30         searchTextbox.clear();  
31         searchTextbox.sendKeys("Selenium");  
32         searchTextbox.submit();  
33  
34 //         WebElement searchButton = driver.findElement(By.cssSelector("#searchButton"));  
35 //         searchButton.click();  
36  
37         assertEquals("Selenium - Wikipedia, la enciclopedia libre", driver.getTitle());  
38     }  
39  
40  
41     @AfterClass  
42     public static void tearDown() throws Exception {  
43         driver.quit();  
44     }  
45 }
```

Interacciones, Cajas de texto

- Con getText() podemos obtener el texto de cualquier elemento web.

```
WebElement message = driver.findElement(By.id("message"));
String messageText = message.getText();
```

Interacciones, Atributos CSS

- Podemos recoger cualquier atributo a través del método `getAttribute()` del interface WebElement.
- Si queremos comprobar un atributo CSS, utilizamos el método `getCssValue()`

```
WebElement message = driver.findElement(By.id("message"));
assertEquals("justify", message.getAttribute("align"));

String width = message.getCssValue("width");
assertEquals("150px", width);
```

```
25 @Test
26 public void testSimpleSearch() {
27
28     WebElement searchTextbox = driver.findElement(By.cssSelector("#searchInput"));
29
30     //test placeholder attribute
31     //test font-size CSS Style attribute
32
33
34     searchTextbox.clear();
35     searchTextbox.sendKeys("Selenium");
36     searchTextbox.submit();
37
38     assertEquals("Selenium - Wikipedia, la enciclopedia libre", driver.getTitle());
39 }
```

Ejercicio

```
25@  
26    @Test  
27    public void testSimpleSearch() {  
28        WebElement searchTextbox = driver.findElement(By.cssSelector("#searchInput"));  
29        String placeholderSearch = searchTextbox.getAttribute("placeholder");  
30        assertEquals("Buscar en Wikipedia", placeholderSearch);  
31        String fontSizeSearch = searchTextbox.getCssValue("font-size");  
32        assertEquals("13px", fontSizeSearch);  
33        searchTextbox.clear();  
34        searchTextbox.sendKeys("Selenium");  
35        searchTextbox.submit();  
36        assertEquals("Selenium - Wikipedia, la enciclopedia libre", driver.getTitle());  
37    }  
38
```

Interacciones, Desplegables

- Selenium WebDriver nos proporciona una manera para poder trabajar con las listas desplegables creadas como elementos HTML <select>.
- Para ello tiene la clase *Select*, con la que podemos interactuar directamente con estos elementos.
- Si solo podemos seleccionar un elemento de la lista

```
Select make = new Select(driver.findElement(By.name("make")));
assertFalse(make.isMultiple());
assertEquals(4, make.getOptions().size());

make.selectByVisibleText("Honda");
assertEquals("Honda", make.getFirstSelectedOption().getText());

make.selectByValue("audi");
assertEquals("Audi", make.getFirstSelectedOption().getText());

make.selectByIndex(0);
assertEquals("BMW", make.getFirstSelectedOption().getText());
```

Interacciones, Desplegables

- Si podemos seleccionar varios

```
Select color = new Select(driver.findElement(By.name("color")));
assertTrue(color.isMultiple());
assertEquals(5, color.getOptions().size());

color.selectByVisibleText("Black");
color.selectByVisibleText("Red");
color.selectByVisibleText("Silver");

color.deselectByValue("rd");
assertEquals(2, color.getAllSelectedOptions().size());

color.deselectByVisibleText("Black");
assertEquals(1, color.getAllSelectedOptions().size());

color.deselectByIndex(2);
assertEquals(0, color.getAllSelectedOptions().size());
```

- Cuando seleccionamos/deseleccionamos por índice tenemos que tener cuidado, puede que los elementos sean dinámicos y obtengamos fallos inesperados al testar.

Ejercicio

```
--  
13 public class DropdownsTest {  
14  
15     private static WebDriver driver;  
16  
17     @BeforeClass  
18     public static void setUp() {  
19  
20         driver = new FirefoxDriver();  
21         driver.manage().window().maximize();  
22         driver.get("https://es.wikipedia.org/wiki/Especial:Contribuciones/");  
23     }  
24  
25     @Test  
26     public void testContributionsDropdown() {  
27  
28         //test the dropdown list set some values and check it  
29     }  
30  
31     @AfterClass  
32     public static void tearDown() throws Exception {  
33         driver.quit();  
34     }  
35 }
```

Ejercicio

```
13 public class DropdownsTest {
14
15     private static WebDriver driver;
16
17     @BeforeClass
18     public static void setUp() {
19
20         driver = new FirefoxDriver();
21         driver.manage().window().maximize();
22         driver.get("https://es.wikipedia.org/wiki/Especial:Contribuciones/");
23     }
24
25     @Test
26     public void testContributionsDropdown() {
27
28         Select namespace = new Select(driver.findElement(By.cssSelector("#namespace")));
29
30         assertFalse(namespace.isMultiple());
31         assertEquals(32, namespace.getOptions().size());
32
33         namespace.selectByVisibleText("Usuario");
34         assertEquals("Usuario", namespace.getFirstSelectedOption().getText());
35
36         namespace.selectByValue("8");
37         assertEquals("MediaWiki", namespace.getFirstSelectedOption().getText());
38
39         namespace.selectByIndex(13);
40         assertEquals("Ayuda", namespace.getFirstSelectedOption().getText());
41     }
42
43     @AfterClass
44     public static void tearDown() throws Exception {
45         driver.quit();
46     }
47 }
```

Interacciones, Desplegables

- También podemos testar si todas las opciones del desplegable son las que esperamos.

```
50     @Test  
51     public void testMonthDropdown() {  
52         Select month = new Select(driver.findElement(By.cssSelector("#month")));  
53         List<String> actualValues = new ArrayList<String>();  
54  
55         for (WebElement element : month.getOptions()) {  
56             actualValues.add(element.getText());  
57         }  
58  
59         assertEquals(monthDropDownList().toArray(), actualValues.toArray());  
60     }  
61 }
```

Interacciones, Radio buttons

- Selenium permite automatizar los radio buttons a través de los métodos del WebElement `click()`, para seleccionarlos, y `isSelected()` para comprobar si están seleccionados

```
WebElement petrol = driver.findElement(By.xpath("//input[@value='Petrol']"));
if (!petrol.isSelected()) {
    petrol.click();
}
```

Ejercicio

```
13 public class RadioButtonsTest {  
14  
15     private static WebDriver driver;  
16  
17     @BeforeClass  
18     public static void setUp() {  
19  
20         driver = new FirefoxDriver();  
21         driver.manage().window().maximize();  
22         driver.get("https://es.wikipedia.org/wiki/Especial:Contribuciones/");  
23     }  
24  
25     @Test  
26     public void testContributionsSearchRadioButtons() {  
27  
28         //test two radio buttons in contributions page, select and test it  
29     }  
30  
31     @AfterClass  
32     public static void tearDown() throws Exception {  
33         driver.quit();  
34     }  
35 }
```

Ejercicio

```
13 public class RadioButtonsTest {  
14  
15     private static WebDriver driver;  
16  
17     @BeforeClass  
18     public static void setUp() {  
19  
20         driver = new FirefoxDriver();  
21         driver.manage().window().maximize();  
22         driver.get("https://es.wikipedia.org/wiki/Especial:Contribuciones/");  
23     }  
24  
25     @Test  
26     public void testContributionsSearchRadioButtons() {  
27  
28         WebElement newbie = driver.findElement(By.cssSelector("#newbie"));  
29         WebElement user = driver.findElement(By.cssSelector("#user"));  
30  
31         assertFalse(newbie.isSelected());  
32         assertTrue(user.isSelected());  
33  
34         newbie.click();  
35  
36         assertTrue(newbie.isSelected());  
37         assertFalse(user.isSelected());  
38     }  
39  
40     @AfterClass  
41     public static void tearDown() throws Exception {  
42         driver.quit();  
43     }  
44 }
```

Interacciones, Checkboxes

- De la misma manera que utilizamos los radio buttons podemos usar los elementos checkbox de una web.

```
WebElement airbags = driver.findElement(By.xpath("//input[@value='Airbags']"));
if (!airbags.isSelected()) {
    airbags.click();
}
assertTrue(airbags.isSelected());
```

Ejercicio

```
15 public class CheckBoxTest {
16
17
18     private static WebDriver driver;
19
20     @BeforeClass
21     public static void setUp() {
22
23         driver = new FirefoxDriver();
24         driver.manage().window().maximize();
25         driver.get("https://es.wikipedia.org/w/index.php?title=Especial:Buscar&search=&fulltext=Buscar&profile=advanced");
26     }
27
28     @Test
29     public void testCheckboxes() {
30
31         //find some checkboxes select and test it
32     }
33
34     @AfterClass
35     public static void tearDown() throws Exception {
36         driver.quit();
37     }
38 }
```

Ejercicio

```
14  
15 public class CheckBoxTest {  
16  
17     private static WebDriver driver;  
18  
20@ BeforeClass  
21     public static void setUp() {  
22  
23         driver = new FirefoxDriver();  
24         driver.manage().window().maximize();  
25         driver.get("https://es.wikipedia.org/w/index.php?title=Especial:Buscar&search=&fulltext=Buscar&profile=advanced");  
26     }  
27  
28@ Test  
29     public void testCheckboxes() {  
30  
31         WebElement mainCheckBox = driver.findElement(By.cssSelector("#mw-search-ns0"));  
32         assertTrue(mainCheckBox.isSelected());  
33  
34         List<WebElement> checkboxes = driver.findElements(By.cssSelector("#mw-searchoptions input[name^='ns']"));  
35  
36         for (WebElement webElement : checkboxes) {  
37  
38             if (!webElement.isSelected()) {  
39                 webElement.click();  
40             }  
41         }  
42  
43         for (WebElement webElement : checkboxes) {  
44             assertTrue(webElement.isSelected());  
45         }  
46     }  
47  
48@ AfterClass  
49     public static void tearDown() throws Exception {  
50         driver.quit();  
51     }  
52 }
```

Selenium API

- Comprobar presencia de elementos
- Comprobar el estado de los elementos
- Interacciones avanzadas
- Ejecutar código javascript
- Capturas de pantalla
- Manejo de cookies
- Eventos del Web Driver

Selenium API. Presencia

- Es una buena práctica detectar la presencia de un elemento dentro de una web antes de realizar alguna acción sobre él.
- Para ello vamos a escribir un método muy simple que capture la excepción NoSuchElementException si no encuentra el elemento.

```
private boolean isElementPresent(By by) {  
    try {  
        driver.findElement(by);  
        return true;  
  
    } catch (NoSuchElementException e) {  
        return false;  
    }  
}
```

```
if (isElementPresent(By.name("airbags"))) {  
    WebElement airbag = driver.findElement(By.name("airbags"));  
    if (!airbag.isSelected()) {  
        airbag.click();  
    }  
  
} else {  
  
    fail("Airbag Checkbox doesn't exists!!");  
}
```

Selenium API. Estado

- Comprobar el estado de los WebElement
- Muchas veces un test falla porque intentamos interactuar con un elemento no visible de la página o deshabilitado .
- Para corregir esto Selenium nos permite detectar el estado del elemento web mediante los siguiente métodos.

Method	Purpose
isEnabled()	This method checks if an element is enabled. It returns <code>true</code> if enabled, else <code>false</code> if disabled.
isSelected()	This method checks if an element is selected (radio button, checkbox, and so on). It returns <code>true</code> if selected, else <code>false</code> if deselected.
isDisplayed()	This method checks if an element is displayed.

Ejercicio

```
15 public class PresentAndStateElementTest {  
16  
17     private static WebDriver driver;  
18  
19     @BeforeClass  
20     public static void setUp() {  
21  
22         driver = new FirefoxDriver();  
23         driver.manage().window().maximize();  
24         driver.get("https://es.wikipedia.org/w/index.php?title=Especial:Crear_una_cuenta");  
25     }  
26  
27     @Test  
28     public void testElementsPresence() {  
29  
30         //test the presence of some elements  
31         //find one hidden and test its state  
32     }  
33  
34     @AfterClass  
35     public static void tearDown() throws Exception {  
36         driver.quit();  
37     }  
..
```

Ejercicio

```
15 public class PresentAndStateElementTest {
16
17     private static WebDriver driver;
18
19     @BeforeClass
20     public static void setUp() {
21
22         driver = new FirefoxDriver();
23         driver.manage().window().maximize();
24         driver.get("https://es.wikipedia.org/w/index.php?title=Especial:Crear_una_cuenta");
25     }
26
27     @Test
28     public void testElementsPresence() {
29
30         if(!isElementPresent(By.name("esteNoEstaNiDeBroma"))){
31             System.out.println("LO SABÍA !!");
32         }
33
34         if(!isElementPresent(By.cssSelector("#userloginForm"))){
35             fail("El elemento #userloginForm no se encuentra en la página");
36         }
37
38         if(isElementPresent(By.cssSelector("#wpEditToken"))){
39             WebElement editToken = driver.findElement(By.cssSelector("#wpEditToken"));
40
41             assertFalse(editToken.isDisplayed());
42         }
43     }
44
45     @AfterClass
46     public static void tearDown() throws Exception {
47         driver.quit();
48     }
49
50     private boolean isElementPresent(By by) {
51         return isElementPresent(by, driver);
52     }
53
54     private boolean isElementPresent(By by, SearchContext in) {
55         try {
56             in.findElement(by);
57             return true;
58         } catch (NoSuchElementException e) {
59             return false;
60         }
61     }
62 }
```

Selenium API. Actions

- Gracias a la API de Selenium podemos automatizar interacciones avanzadas, como mandar keystrokes o automatizar un doble click o drag and drop.
- Para ello disponemos de la clase Actions, en la que podemos automatizar una acción o un grupo de acciones.

```
Actions builder = new Actions(driver);
builder.click(tableRows.get(1))
    .keyDown(Keys.CONTROL)
    .click(tableRows.get(3))
    .keyUp(Keys.CONTROL)
    .build().perform();
```

Selenium API. Doble Click

- Doble click

```
Actions builder = new Actions(driver);
builder.doubleClick(message).perform();
```

```
14 public class DoubleClickTest {
15
16     private static WebDriver driver;
17
18     @BeforeClass
19     public static void setUp() {
20
21         driver = new FirefoxDriver();
22         driver.manage().window().maximize();
23         driver.get("http://cookbook.seleniumacademy.com/DoubleClickDemo.html");
24     }
25
26     @Test
27     public void testDoubleClick() throws Exception {
28
29         WebElement message = driver.findElement(By.id("message"));
30
31         // Verify color is Blue
32         assertEquals("rgba(0, 0, 255, 1)", message.getCssValue("background-color"));
33
34         Actions builder = new Actions(driver);
35         builder.doubleClick(message).perform();
36
37         // Verify Color is Yellow
38         assertEquals("rgba(255, 255, 0, 1)", message.getCssValue("background-color"));
39     }
40
41     @AfterClass
42     public static void tearDown() throws Exception {
43         driver.quit();
44     }
45 }
```

Selenium API. Drag And Drop

- Drag and drop.
- Utilizamos la función dragAndDrop de la clase Actions
- A esta función deberemos pasarle el origen y el destino, los cuales deberán ser WebElements.

```
WebElement source = driver.findElement(By.id("draggable"));
WebElement target = driver.findElement(By.id("droppable"));
Actions builder = new Actions(driver);
builder.dragAndDrop(source, target) .perform();
```

```
14 public class DragAndDropTest {
15
16     private static WebDriver driver;
17
18     @BeforeClass
19     public static void setUp() {
20
21         driver = new FirefoxDriver();
22         driver.manage().window().maximize();
23         driver.get("http://cookbook.seleniumacademy.com/DragDropDemo.html");
24     }
25
26     @Test
27     public void testDragAndDrop() {
28
29         WebElement source = driver.findElement(By.id("draggable"));
30         WebElement target = driver.findElement(By.id("droppable"));
31
32         Actions builder = new Actions(driver);
33         builder.dragAndDrop(source, target) .perform();
34         assertEquals("Dropped!", target.getText());
35     }
36
37     @AfterClass
38     public static void tearDown() throws Exception {
39         driver.quit();
40     }
41 }
```

Selenium API. Javascript

- Javascript.
- Podemos ejecutar código javascript igual que lo hacíamos cuando buscábamos elementos con jQuery.
- Con la clase JavascriptExecutor ejecutamos todo el código.

```
@Test
public void testJavaScriptCalls() throws Exception {
    WebDriver driver = new ChromeDriver();
    driver.get("http://www.google.com");
    try {
        JavascriptExecutor js = (JavascriptExecutor) driver;
        String title = (String) js.executeScript("return document.title");
        assertEquals("Google", title);

        long links = (Long) js.executeScript("var links = document.getElementsByTagName('A'); return links.length");
        assertEquals(42, links);

    } finally {
        driver.quit();
    }
}
```

Selenium API. Javascript. jQuery

- Podemos usar todo el API de jQuery con selenium.
- Para poder hacerlo necesitamos una instancia *Javascript* del WebDriver.
- Una vez la tenemos, inyectando código *javascript*, podemos acceder a la API de jQuery.

```
46 @Test
47 public void findAdvancedSearchComponentsWithjQuerySelectors() {
48     JavascriptExecutor js = (JavascriptExecutor)driver;
49
50     @SuppressWarnings("unchecked")
51     List<WebElement> elementsChecked = (List<WebElement>)js.executeScript("return jQuery.find(':checked')");
52
53     @SuppressWarnings("unchecked")
54     List<WebElement> elementsUnChecked = (List<WebElement>)js.executeScript("return jQuery.find('input:checkbox:not(:checked)'");
55
56     assertEquals(elementsChecked.size(), 3);
57     assertEquals(elementsUnChecked.size(), 27);
58 }
59
60
```

Selenium API. Screenshot

- TakesScreenshot nos permite sacar una captura de pantalla de la página web.
- Esto nos permite ver en qué estado se encontraba la página cuando ha fallado un test.

```
File scrFile = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);
```

Ejercicio

```
--+
22 public class ScreenShotTest {
23
24     private static WebDriver driver;
25
26     @BeforeClass
27     public static void setUp() {
28
29         driver = new FirefoxDriver();
30         driver.manage().window().maximize();
31         driver.get("https://es.wikipedia.org/w/index.php?title=Especial:Buscar&search=&fulltext=Buscar&profile=advanced");
32     }
33
34     @Test
35     public void testCheckBoxes() throws IOException {
36
37         WebElement mainCheckBox = driver.findElement(By.cssSelector("#mw-search-ns0"));
38         assertTrue(mainCheckBox.isSelected());
39
40         List<WebElement> checkBoxes = driver.findElements(By.cssSelector("#mw-searchoptions input[name^='ns']"));
41
42         //take a screenshot before select all the checkboxes
43
44         for (WebElement webElement : checkBoxes) {
45
46             if (!webElement.isSelected()) {
47                 webElement.click();
48             }
49         }
50
51         //take a screenshot after select all the checkboxes
52     }
53
54     @AfterClass
55     public static void tearDown() throws Exception {
56         driver.quit();
57     }
58 }
```

Ejercicio

```
21
22 public class ScreenShotTest {
23
24     private static WebDriver driver;
25
26     @BeforeClass
27     public static void setUp() {
28
29         driver = new FirefoxDriver();
30         driver.manage().window().maximize();
31         driver.get("https://es.wikipedia.org/w/index.php?title=Especial:Buscar&search=&fulltext=Buscar&profile=advanced");
32     }
33
34     @Test
35     public void testCheckBoxes() throws IOException {
36
37         WebElement mainCheckBox = driver.findElement(By.cssSelector("#mw-search-ns0"));
38         assertTrue(mainCheckBox.isSelected());
39
40         List<WebElement> checkBoxes = driver.findElements(By.cssSelector("#mw-searchoptions input[name^='ns']"));
41
42         File scrFile = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);
43         FileUtils.copyFile(scrFile, new File("src/test/resources/checkBox_before.png"));
44
45         for (WebElement webElement : checkBoxes) {
46
47             if (!webElement.isSelected()) {
48                 webElement.click();
49             }
50         }
51
52         File after = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);
53         FileUtils.copyFile(after, new File("src/test/resources/checkBox_after.png"));
54     }
55
56     @AfterClass
57     public static void tearDown() throws Exception {
58         driver.quit();
59     }
60 }
```

Selenium API. Cookies

- Manejo de cookies.
- Selenium nos permite acceder, añadir y borrar cookies a través de su interface WebDriver.Options

Method	Description
addCookie(Cookie cookie)	This method adds a cookie.
getCookieNamed(String name)	This method returns the cookie with a specified name.
getCookies()	This method returns all the cookies for current domain.
deleteCookieNamed(String name)	This method deletes the cookie with a specified name.
deleteCookie(Cookie cookie)	This method deletes a cookie.
deleteAllCookies()	This method deletes all the cookies for current domain.

Ejercicio

```
14 public class CookiesTest {  
15  
16     private static WebDriver driver;  
17  
18     @BeforeClass  
19     public static void setUp() {  
20  
21         driver = new FirefoxDriver();  
22         driver.manage().window().maximize();  
23         driver.get("https://es.wikipedia.org/wiki/Wikipedia:Portada");  
24     }  
25  
26     @Test  
27     public void testCookies() {  
28  
29         //get, delete and test some cookies  
30     }  
31  
32     @AfterClass  
33     public static void tearDown() throws Exception {  
34         driver.quit();  
35     }  
36 }  
37
```

Ejercicio

```
13
14 public class CookiesTest {
15
16     private static WebDriver driver;
17
18     @BeforeClass
19     public static void setUp() {
20
21         driver = new FirefoxDriver();
22         driver.manage().window().maximize();
23         driver.get("https://es.wikipedia.org/wiki/Wikipedia:Portada");
24     }
25
26     @Test
27     public void testCookies() {
28
29         Cookie geoIPCookie = driver.manage().getCookieNamed("GeoIP");
30         assertNotNull(geoIPCookie);
31
32         Set<Cookie> cookies = driver.manage().getCookies();
33         assertEquals(cookies.size() > 0, true);
34
35         driver.manage().deleteAllCookies();
36
37         cookies = driver.manage().getCookies();
38         assertEquals(cookies.size() == 0, true);
39     }
40
41     @AfterClass
42     public static void tearDown() throws Exception {
43         driver.quit();
44     }
45 }
```

Selenium API. IFrame

- Cuando las páginas están compuestas por otras páginas, es habitual el uso de **IFrame**, esto introduce nuevos árboles HTML, que evitan que se pueda seleccionar de forma sencilla los elementos dentro de estos subárboles.
- Para ello Selenium ofrece dentro del API, la opción de cambiar al IFrame deseado, con el método `switchTo(nameFrame)`

Selenium API. Windows

- Cuando las páginas abren otras páginas a través de enlaces, para situar el foco del objeto WebDriver sobre la nueva página, se ha de emplear el API de WindowHandle.

Selenium API. Eventos

- Se pueden escuchar eventos sobre la interacción con el objeto WebDriver.
- Selenium nos proporciona la clase EventFiringWebDriver que escucha eventos como pulsar un link, buscar un elemento, cambiar algún valor o cuando se lanza una excepción.

Event	Description
beforeNavigateTo	This method is called before the get (String url) or the navigate () .to (String url) method is called.
afterNavigateTo	This method is called after the get (String url) or the navigate () .to (String url) method is called.
beforeNavigateBack	This method is called before the navigate () .back () method.
afterNavigateBack	This method is called after the navigate () .back () method.
beforeNavigateForward	This method is called before the navigate () .forward () method.
afterNavigateForward	This method is called after the navigate () .forward () method.
beforeFindBy	This method is called before the following methods: <ul style="list-style-type: none">▶ WebDriver.findElement (...)▶ WebDriver.findElements (...)▶ WebElement.findElement (...)▶ WebElement.findElements (...)
afterFindBy	This method is called after the following methods: <ul style="list-style-type: none">▶ WebDriver.findElement (...)▶ WebDriver.findElements (...)▶ WebElement.findElement (...)▶ WebElement.findElements (...)

Event	Description
beforeScript	This method is called before the RemoteWebDriver .executeScript (java.lang.String, java.lang.Object []) method.
afterScript	This method is called after the RemoteWebDriver .executeScript (java.lang.String, java.lang.Object []) method.
onException	This method is called whenever an exception would be thrown.
beforeChangeValueOf	This method is called before the WebElement.clear () or the WebElement.sendKeys (...) method.
afterChangeValueOf	This method is called after the WebElement.clear () or the WebElement.sendKeys (...) method.
beforeClickOn	This method is called before the WebElement.click () method.
afterClickOn	This method is called after the WebElement.click () method.

Selenium API.

Eventos

```
16
17 public class TraceListener implements WebDriverEventListener {
18
```

```
14 public class EventTest {
15
16
17     private static WebDriver driver;
18
19     @BeforeClass
20     public static void setUp() {
21
22         driver = new FirefoxDriver();
23         driver.manage().window().maximize();
24     }
25
26     @Test
27     public void testEvents() {
28
29         EventFiringWebDriver eventDriver = new EventFiringWebDriver(driver);
30         TraceListener myListener = new TraceListener();
31         eventDriver.register(myListener);
32
33         eventDriver.get("https://es.wikipedia.org/w/index.php?title=Especial:Buscar&search=&fulltext=Buscar&profile=advanced");
34
35         List<WebElement> checkBoxes = eventDriver.findElements(By.cssSelector("#mw-searchoptions input[name^='ns']"));
36
37         for (WebElement webElement : checkBoxes) {
38
39             if (!webElement.isSelected()) {
40                 webElement.click();
41             }
42         }
43
44         WebElement searchTextbox = eventDriver.findElement(By.cssSelector("#searchText .oo-ui-inputWidget-input"));
45
46         searchTextbox.clear();
47         searchTextbox.sendKeys("Selenium");
48         searchTextbox.submit();
49
50         eventDriver.findElement(By.cssSelector("excepción para que la capture el evento y haga una captura"));
51     }
52
53
54     @AfterClass
55     public static void tearDown() {
56         driver.quit();
57     }
58 }
```

Sincronización

- Sincronización con wait
- Sincronización con condiciones esperadas
- Sincronización con FluentWait

Sincronización

- Wait implícito.
- Cuando definimos este tipo de wait, dentro de nuestros test, el sistema esperará una determinada cantidad de tiempo, mientras busca el elemento en el DOM de la página que le preceda.
- Si ha pasado ese tiempo y el elemento no aparece lanza una excepción NoSuchElementException.

```
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
```

- Aunque no pongamos este wait, el sistema por defecto espera 10 segundos si no encuentra el elemento dentro de un findElement()

Sincronización

- Wait explícito.
- Proporciona una manera mucho mas óptima de sincronización.
- Para ello disponemos de WebDriverWait y ExpectedCondition.
- Si la condición se cumple antes del tiempo, deja de esperar.
- Condiciones predefinidas.

Predefined condition	Selenium method
An element is visible and enabled	elementToBeClickable(By locator)
An element is selected	elementToBeSelected(WebElement element)
Presence of an element	presenceOfElementLocated(By locator)
Specific text present in an element	textToBePresentInElement(By locator, java.lang.String text)
Element value	textToBePresentInElementValue(By locator, java.lang.String text)
Title	titleContains(java.lang.String title)

Sincronización

- Wait explícito.

```
WebDriverWait wait = new WebDriverWait(driver, 10);
wait.until(ExpectedConditions.titleContains("selenium"));

new WebDriverWait(driver, 10).until(new ExpectedCondition<Boolean>() {
    public Boolean apply(WebDriver d) {
        return d.getTitle().toLowerCase().startsWith("selenium");
        //return d.findElement(By.id("page4"));
    }
});
```

Sincronización

- FluentWait. Implementación del interface Wait.
- Nos permite indicar al sistema una cantidad de tiempo a esperar y la frecuencia con la que va a buscar un elemento en el DOM de la página web.
- También podemos indicarle que ignore cierto tipo de excepciones como NoSuchElementException

```
Wait<WebDriver> wait = new FluentWait<WebDriver>(driver)
    .withTimeout(10, TimeUnit.SECONDS)
    .pollingEvery(2, TimeUnit.SECONDS)
    .ignoring(NoSuchElementException.class);

WebElement message = wait.until(new Function<WebDriver, WebElement>() {

    public WebElement apply(WebDriver d) {
        return d.findElement(By.id("page4"));
    }
});
```

Ejercicio

```
20 public class SincronizationTest {  
21  
22     private static WebDriver driver;  
23  
24     @BeforeClass  
25     public static void setUp() {  
26  
27         driver = new FirefoxDriver();  
28         driver.manage().window().maximize();  
29         driver.get("https://es.wikipedia.org/wiki/Wikipedia:Portada");  
30     }  
31  
32     @Test  
33     public void testSincronizationWithWaits() {  
34  
35         //make an advanced search, with an explicit wait and a Fluent wait  
36         //test the results  
37     }  
38  
39  
40     @AfterClass  
41     public static void tearDown() {  
42         driver.quit();  
43     }  
44 }
```

Data-Driven testing

- Con JUnit
- Con TestNG
- Archivos CSV
- Archivos Excel con JUnit y Apache POI

Data-Driven testing

- Gracias al testing basado en los datos, podemos usar el mismo código de test para testar diferentes condiciones (datos).
- Podemos testar, por ejemplo toda una tabla, sin tener que hacer un test para cada registro.
- Selenium por si solo no proporciona ningún mecanismo para realizar este tipo de test, pero vamos a ver diferentes formas de hacerlo.

Data-Driven testing

- JUnit.
- Teniendo en cuenta la siguiente tabla

Height (centimeters)	Weight (kilograms)	BMI	Category
160	45	17.6	Underweight
168	70	24.8	Normal
181	89	27.2	Overweight
178	100	31.6	Obesity

Data-Driven testing

- JUnit.
- Primero declaramos una clase con la anotación @RunWith

```
16  @RunWith(Parameterized.class)
17  public class JUnitDataDrivenTest {
```

- Añadimos las variables necesarias, los diferentes datos de nuestro test y un método con los distintos valores de los mismos.

```
21      private String height;
22      private String weight;
23      private String bmi;
24      private String bmiCategory;
25
26  @Parameters
27  public static List<String[]> testData() {
28      return Arrays.asList(new String[][] {
29          { "160", "45", "17.6", "Underweight" },
30          { "168", "70", "24.8", "Normal" },
31          { "181", "89", "27.2", "Overweight" },
32          { "178", "100", "31.6", "Obesity" } });
33  }
```

Data-Driven testing

- JUnit.
- Constructor por parámetros

```
35  public JUnitDataDrivenTest(String height, String weight, String bmi, String bmiCategory) {  
36      this.height = height;  
37      this.weight = weight;  
38      this.bmi = bmi;  
39      this.bmiCategory = bmiCategory;  
40  }  
..  
42  @Test  
43  public void testBMICalculator() throws Exception {  
44  
45      // Get the Height element and set the value using height variable  
46      WebElement heightField = driver.findElement(By.name("heightCMS"));  
47      heightField.clear();  
48      heightField.sendKeys(height);  
49  
50      // Get the Weight element and set the value using weight variable  
51      WebElement weightField = driver.findElement(By.name("weightKg"));  
52      weightField.clear();  
53      weightField.sendKeys(weight);  
54  
55      // Click on Calculate Button  
56      WebElement calculateButton = driver.findElement(By.id("Calculate"));  
57      calculateButton.click();  
58  
59      // Get the Bmi element and verify its value using bmi variable  
60      WebElement bmiLabel = driver.findElement(By.name("bmi"));  
61      assertEquals(bmi, bmiLabel.getAttribute("value"));  
62  
63      // Get the Bmi Category element and verify its value using  
64      // bmiCategory variable  
65      WebElement bmiCategoryLabel = driver.findElement(By.name("bmi_category"));  
66      assertEquals(bmiCategory, bmiCategoryLabel.getAttribute("value"));  
67 }
```



Ejercicio

```
20  
21 @RunWith(Parameterized.class)  
22 public class JUnitDataDrivenTest {  
23  
24     /**  
25      * Using Data driven testing test the create account use case  
26      */  
27  
28     private static WebDriver driver;  
29  
30     @BeforeClass  
31     public static void setUp() {  
32  
33         driver = new FirefoxDriver();  
34         driver.manage().window().maximize();  
35         driver.get("https://es.wikipedia.org/w/index.php?title=Especial:Crear_una_cuenta");  
36     }  
37  
38     @Parameters  
39     public static List<String[]> testData() {  
40         return Arrays.asList(new String[][] {  
41             {}  
42         });  
43     }  
44  
45     @Test  
46     public void testRegister() {  
47     }  
48  
49     @AfterClass  
50     public static void tearDown() {  
51         driver.quit();  
52     }  
53 }
```

Ejercicio

```
20
21 @RunWith(Parameterized.class)
22 public class JUnitDataDrivenTest {
23
24     private static WebDriver driver;
25
26     @BeforeClass
27     public static void setUp() {
28
29         driver = new FirefoxDriver();
30         driver.manage().window().maximize();
31         driver.get("https://es.wikipedia.org/w/index.php?title=Especial:Crear_una_cuenta");
32     }
33
34     private String username;
35     private String password;
36     private String confirmPassword;
37     private String email;
38     private String captcha;
39
40     @Parameters
41     public static List<String[]> testData() {
42         return Arrays.asList(new String[][] {
43             { "manolo",      "1357",      "1357",      "manolo@gmail.com",      "drfhhs" },
44             { "fulano",      "2468",      "2468",      "fulano@gmail.com",      "asdhwetv" },
45             { "mengano",     "asdfghj",   "asdfghj",   "mengano@gmail.com",   "dsfyrthjgh" },
46             { "pepe",        "qwerty",    "qwerty",    "pepe@gmail.com",      "dfet5yhhgs" } });
47     }
48
49     public JUnitDataDrivenTest(String username, String password, String confirmPassword, String email, String captcha) {
50         this.username = username;
51         this.password = password;
52         this.confirmPassword = confirmPassword;
53         this.email = email;
54         this.captcha = captcha;
55     }
```

Ejercicio

```
--  
57@  @Test  
58  public void testRegister() {  
59  
60      WebElement usernameElement = driver.findElement(By.cssSelector("#wpName2"));  
61      usernameElement.clear();  
62      usernameElement.sendKeys(username);  
63  
64      WebElement passElement = driver.findElement(By.cssSelector("#wpPassword2"));  
65      passElement.clear();  
66      passElement.sendKeys(password);  
67  
68      WebElement confirmPassElement = driver.findElement(By.cssSelector("#wpRetype"));  
69      confirmPassElement.clear();  
70      confirmPassElement.sendKeys(confirmPassword);  
71  
72      WebElement emailElement = driver.findElement(By.cssSelector("#wpEmail"));  
73      emailElement.clear();  
74      emailElement.sendKeys(email);  
75  
76      WebElement captchaElement = driver.findElement(By.cssSelector("#mw-input-captchaWord"));  
77      captchaElement.clear();  
78      captchaElement.sendKeys(captcha);  
79  
80      WebElement submitButton = driver.findElement(By.cssSelector("#wpCreateaccount"));  
81      submitButton.click();  
82  
83      FluentWait<By> fluentWait = new FluentWait<By>(By.cssSelector(".error"))  
84          .withTimeout(10, TimeUnit.SECONDS)  
85          .pollingEvery(2, TimeUnit.SECONDS);  
86  
87@      fluentWait.until(new Predicate<By>() {  
△88@          public boolean apply(By by) {  
89              return driver.findElement(by).isDisplayed();  
90          }  
91      });  
92  }  
93  
94@  @AfterClass  
95  public static void tearDown() {  
96      driver.quit();  
97  }
```

Data-Driven testing

- TestNG
- Añadimos las dependencias del framework

```
<dependency>
  <groupId>org.testng</groupId>
  <artifactId>testng</artifactId>
  <version>6.9.4</version>
  <scope>test</scope>
</dependency>
```

- Su funcionamiento es muy parecido al de JUnit, tiene un array con los datos del test y un método que recoge esos datos para testar.

TestNG

```
2
3@import org.testng.annotations.*;
4
5 import com.google.common.base.Predicate;
6 import java.util.concurrent.TimeUnit;
7 import org.openqa.selenium.By;
8 import org.openqa.selenium.WebDriver;
9 import org.openqa.selenium.WebElement;
10 import org.openqa.selenium.firefox.FirefoxDriver;
11 import org.openqa.selenium.support.ui.FluentWait;
12
13 public class TestNGDataDrivenTest {
14
15     private static WebDriver driver;
16
17@    @BeforeClass
18    public static void setUp() {
19
20        driver = new FirefoxDriver();
21        driver.manage().window().maximize();
22        driver.get("https://es.wikipedia.org/w/index.php?title=Especial:Crear_una_cuenta");
23    }
24
25@    @DataProvider
26    public Object[][] testData() {
27
28        return new Object[][] {
29            { "manolo",      "1357",      "1357",      "manolo@gmail.com",      "drfhrs" },
30            { "fulano",      "2468",      "2468",      "fulano@gmail.com",      "asdhwetv" },
31            { "mengano",     "asdfghj",   "asdfghj",   "mengano@gmail.com",   "dsfyrthjgh" },
32            { "pepe",         "qwerty",    "qwerty",    "pepe@gmail.com",      "dfet5yhhgs" } };
33    }
34}
```

TestNG

```
34
35 @TestdataProvider = "testData")
36 public void testRegister (String username, String password, String confirmPassword, String email, String captcha) {
37
38     WebElement usernameElement = driver.findElement(By.cssSelector("#wpName2"));
39     usernameElement.clear();
40     usernameElement.sendKeys(username);
41
42     WebElement passElement = driver.findElement(By.cssSelector("#wpPassword2"));
43     passElement.clear();
44     passElement.sendKeys(password);
45
46     WebElement confirmPassElement = driver.findElement(By.cssSelector("#wpRetype"));
47     confirmPassElement.clear();
48     confirmPassElement.sendKeys(confirmPassword);
49
50     WebElement emailElement = driver.findElement(By.cssSelector("#wpEmail"));
51     emailElement.clear();
52     emailElement.sendKeys(email);
53
54     WebElement captchaElement = driver.findElement(By.cssSelector("#mw-input-captchaWord"));
55     captchaElement.clear();
56     captchaElement.sendKeys(captcha);
57
58     WebElement submitButton = driver.findElement(By.cssSelector("#wpCreateaccount"));
59     submitButton.click();
60
61     FluentWait<By> fluentWait = new FluentWait<By>(By.cssSelector(".error"))
62         .withTimeout(10, TimeUnit.SECONDS)
63         .pollingEvery(2, TimeUnit.SECONDS);
64
65     fluentWait.until(new Predicate<By>() {
66         public boolean apply(By by) {
67             return driver.findElement(by).isDisplayed();
68         }
69     });
70 }
71
72 @AfterClass
73 public static void tearDown() {
74     driver.quit();
75 }
76 }
```

Data-Driven testing

- Archivos CSV.
- Primero para leer estos archivos CSV vamos a utilizar la librería OpenCSV, por lo que debemos añadir sus dependencias

```
<dependency>
  <groupId>com.opencsv</groupId>
  <artifactId>opencsv</artifactId>
  <version>3.4</version>
  <scope>test</scope>
</dependency>
```

- Realizaremos los test con JUnit exactamente igual que como hemos visto hasta ahora. La única diferencia es que ahora leeremos los datos de un archivo .csv

Data-Driven testing

```
18
19 @RunWith(Parameterized.class)
20 public class CSVDataDrivenTest {
21
22     private static WebDriver driver;
23
24     private String username;
25     private String password;
26     private String confirmPassword;
27     private String email;
28     private String captcha;
29
30     @Parameters
31     public static List<String[]> testData() throws IOException {
32         return getTestData("./src/test/resources/testData/registerData.csv");
33     }
34
35     public CSVDataDrivenTest(String username, String password, String confirmPassword, String email, String captcha) {
36         this.username = username;
37         this.password = password;
38         this.confirmPassword = confirmPassword;
39         this.email = email;
40         this.captcha = captcha;
41     }
42
43     public static List<String[]> getTestData(String fileName) throws IOException {
44         CSVReader reader = new CSVReader(new FileReader(fileName));
45         List<String[]> myEntries = reader.readAll();
46         reader.close();
47         return myEntries;
48     }
49
50     @BeforeClass
51     public static void setUp() {
52
53         driver = new FirefoxDriver();
54         driver.manage().window().maximize();
55         driver.get("https://es.wikipedia.org/w/index.php?title=Especial:Crear_una_cuenta");
56     }
--
```

Data-Driven testing

```
57
58@Test
59public void testRegister() {
60
61    WebElement usernameElement = driver.findElement(By.cssSelector("#wpName2"));
62    usernameElement.clear();
63    usernameElement.sendKeys(username);
64
65    WebElement passElement = driver.findElement(By.cssSelector("#wpPassword2"));
66    passElement.clear();
67    passElement.sendKeys(password);
68
69    WebElement confirmPassElement = driver.findElement(By.cssSelector("#wpRetype"));
70    confirmPassElement.clear();
71    confirmPassElement.sendKeys(confirmPassword);
72
73    WebElement emailElement = driver.findElement(By.cssSelector("#wpEmail"));
74    emailElement.clear();
75    emailElement.sendKeys(email);
76
77    WebElement captchaElement = driver.findElement(By.cssSelector("#mw-input-captchaWord"));
78    captchaElement.clear();
79    captchaElement.sendKeys(captcha);
80
81    WebElement submitButton = driver.findElement(By.cssSelector("#wpCreateaccount"));
82    submitButton.click();
83
84    FluentWait<By> fluentWait = new FluentWait<By>(By.cssSelector(".error"))
85        .withTimeout(10, TimeUnit.SECONDS)
86        .pollingEvery(2, TimeUnit.SECONDS);
87
88    fluentWait.until(new Predicate<By>() {
89        public boolean apply(By by) {
90            return driver.findElement(by).isDisplayed();
91        }
92    });
93}
94
95@AfterClass
96public static void tearDown() throws Exception {
97    // Close the browser
98    driver.quit();
99}
```

Data-Driven testing

- Archivos Excel.
- Para leer archivos de excel vamos a necesitar varias librerías.

```
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi</artifactId>
    <version>3.15</version>
    <scope>test</scope>
</dependency>

<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-ooxml</artifactId>
    <version>3.15</version>
</dependency>
```

Data-Driven testing

- Archivos Excel.
- Una vez añadidas estas librerías necesitaremos una clase que nos lea el archivo y nos genere una lista para poder añadir a nuestros tests.

```
17 public class SpreadsheetData {  
18     private transient Collection<Object[]> data = null;  
19  
20     public SpreadsheetData(final InputStream excelInputStream) throws IOException, InvalidFormatException {  
21         this.data = loadFromSpreadsheet(excelInputStream);  
22     }  
23  
24     public Collection<Object[]> getData() {  
25         return data;  
26     }  
27  
28     private Collection<Object[]> loadFromSpreadsheet(final InputStream excelFile) throws IOException, InvalidFormatException {  
29         Workbook workbook = WorkbookFactory.create(excelFile);  
30  
31         data = new ArrayList<Object[]>();  
32         Sheet sheet = workbook.getSheetAt(0);  
33  
34         int numberofColumns = countNonEmptyColumns(sheet);  
35         List<Object[]> rows = new ArrayList<Object[]>();  
36         List<Object> rowData = new ArrayList<Object>();  
37  
38         for (Row row : sheet) {  
39             if (isEmpty(row)) {  
40                 break;  
41             } else {  
42                 if (row.getRowNum() != 0) //Row 0 will be Header Row  
43                 {  
44                     rowData.clear();  
45                     for (int column = 0; column < numberofColumns; column++) {  
46                         Cell cell = row.getCell(column);  
47                         rowData.add(objectFrom(workbook, cell));  
48                     }  
49                     rows.add(rowData.toArray());  
50                 }  
51             }  
52         }  
53     }  
54     return rows;  
55 }  
56  
57 //(...)  
58
```

Data-Driven testing

- Archivos Excel.

```
20
21 @RunWith(Parameterized.class)
22 public class ExcelDataDrivenTest {
23
24     private static WebDriver driver;
25
26     private String username;
27     private String password;
28     private String confirmPassword;
29     private String email;
30     private String captcha;
31
32     @BeforeClass
33     public static void setUp() {
34
35         driver = new FirefoxDriver();
36         driver.manage().window().maximize();
37         driver.get("https://es.wikipedia.org/w/index.php?title=Especial:Crear_una_cuenta");
38     }
39
40     @SuppressWarnings("rawtypes")
41     @Parameters
42     public static Collection testData() throws Exception {
43         InputStream spreadsheet = new FileInputStream("./src/test/resources/testData/registerData.xlsx");
44         return new SpreadsheetData(spreadsheet).getData();
45     }
46
47     public ExcelDataDrivenTest(String username, String password, String confirmPassword, String email, String captcha) {
48         this.username = username;
49         this.password = password;
50         this.confirmPassword = confirmPassword;
51         this.email = email;
52         this.captcha = captcha;
53     }
54
```

Data-Driven testing

- Archivos Excel.

```
57
58@  @Test
59  public void testRegister() {
60
61      WebElement usernameElement = driver.findElement(By.cssSelector("#wpName2"));
62      usernameElement.clear();
63      usernameElement.sendKeys(username);
64
65      WebElement passElement = driver.findElement(By.cssSelector("#wpPassword2"));
66      passElement.clear();
67      passElement.sendKeys(password);
68
69      WebElement confirmPassElement = driver.findElement(By.cssSelector("#wpRetype"));
70      confirmPassElement.clear();
71      confirmPassElement.sendKeys(confirmPassword);
72
73      WebElement emailElement = driver.findElement(By.cssSelector("#wpEmail"));
74      emailElement.clear();
75      emailElement.sendKeys(email);
76
77      WebElement captchaElement = driver.findElement(By.cssSelector("#mw-input-captchaWord"));
78      captchaElement.clear();
79      captchaElement.sendKeys(captcha);
80
81      WebElement submitButton = driver.findElement(By.cssSelector("#wpCreateaccount"));
82      submitButton.click();
83
84      FluentWait<By> fluentWait = new FluentWait<By>(By.cssSelector(".error"))
85          .withTimeout(10, TimeUnit.SECONDS)
86          .pollingEvery(2, TimeUnit.SECONDS);
87
88
89@     fluentWait.until(new Predicate<By>() {
90         public boolean apply(By by) {
91             return driver.findElement(by).isDisplayed();
92         }
93     });
94
95@     @AfterClass
96     public static void tearDown() {
97         driver.quit();
98     }
```

Page Object Model

- PageFactory.
- Selenium nos permite agrupar todos los elementos de una página web en una misma clase PageFactory.
- De esta manera podemos reutilizar el código sin preocuparnos de localizar los elementos dentro de la web.
- Primero tenemos que crear una clase que contenga todos los elementos de la página web, nos creamos las variables

```
import org.openqa.selenium.support.FindBy;  
  
@FindBy(id = "bmi_category")  
private WebElement bmiCategory;
```

- Si queremos que el sistema cachee los elementos, utilizamos la anotación @CacheLookup

Page Object Model

- PageFactory.
- Nos creamos un constructor que reciba un WebDriver e inicie los elementos de PageFactory

```
public BmiCalcPage(WebDriver driver) {  
    this.driver = driver;  
    PageFactory.initElements(driver, this);  
}
```

- Métodos de ayuda para manejar los WebElements

```
public void setWeight(String weight) {  
    weightKg.sendKeys(weight);  
}
```

- Test que utilice este objeto

Ejercicio

```
10
11 public class PageFactoryTest {
12
13     /*
14      * Code the WikipediaMainPage Factory to run this test
15      */
16
17     private static WebDriver driver;
18
19     @BeforeClass
20     public static void setUp() {
21
22         driver = new FirefoxDriver();
23         driver.manage().window().maximize();
24         driver.get(WikipediaMainPage.url);
25     }
26
27     @Test
28     public void testSimpleSearch() {
29
30         WikipediaMainPage page = new WikipediaMainPage(driver);
31
32         assertEquals("Buscar en Wikipedia", page.getPlaceholder());
33         page.setSearchText("Selenium");
34         page.search();
35
36         assertEquals("Selenium - Wikipedia, la enciclopedia libre", driver.getTitle());
37     }
38
39     @AfterClass
40     public static void tearDown() {
41         driver.quit();
42     }
43 }
```

Ejercicio

```
/
8  public class WikipediaMainPage {
9
10     public static String url = "https://es.wikipedia.org/wiki/Wikipedia:Portada";
11
12     private WebDriver driver;
13
14     @FindBy(id = "searchInput")
15     private WebElement searchBox;
16
17     @FindBy(id = "searchButton")
18     private WebElement searchButton;
19
20     @FindBy(id = "pt-anoncontribs")
21     private WebElement contributionsLink;
22
23     @FindBy(id = "pt-createaccount")
24     private WebElement createAccountLink;
25
26     public WikipediaMainPage(WebDriver driver) {
27
28         this.driver = driver;
29         PageFactory.initElements(this.driver, this);
30     }
31
32     public void setSearchText(String text) {
33         this.searchBox.clear();
34         this.searchBox.sendKeys(text);
35     }
36
37     public void search() {
38         this.searchButton.click();
39     }
40
41     public void contributors() {
42         this.contributionsLink.click();
43     }
44
45     public void createAccount() {
46         this.createAccountLink.click();
47     }
48
49     public String getPlaceholder() {
50         return searchBox.getAttribute("placeholder");
51     }
52 }
```

Page Object Model

- PageFactory.
- Podemos usar la clase creada para realizar operaciones en ella, de tal manera que el test no tenga que conocer nada de la página.

```
10
11 public class PageFactoryOperationsTest {
12
13     private static WebDriver driver;
14
15     @BeforeClass
16     public static void setUp() {
17
18         driver = new FirefoxDriver();
19         driver.manage().window().maximize();
20     }
21
22     @Test
23     public void testSimpleSearch() {
24
25         WikipediaMainPageOperationsPage page = new WikipediaMainPageOperationsPage(driver);
26         page.load();
27         page.makeSearch("Selenium");
28
29         assertEquals("Selenium - Wikipedia, la enciclopedia libre", driver.getTitle());
30
31         page.close();
32     }
33
34     @AfterClass
35     public static void tearDown() {
36         driver.quit();
37     }
38 }
```

```
32     public void load() {
33         this.driver.get(url);
34     }
35
36     public void close() {
37         this.driver.close();
38     }
39
40     public void makeSearch(String text) {
41
42         this.setSearchText(text);
43         this.search();
44     }
45 }
```

Page Object Model

- LoadableComponent.
- Podemos heredar de esta clase abstracta para proporcionar una manera estándar de cargar nuestras páginas.
- Nos proporciona dos métodos, load() y isLoaded() para determinar si la página está correctamente presentada.
- Una vez implementado, podemos cargar la página simplemente con get()

```
extends LoadableComponent<myPage>

//methods
protected void load() {
    this.driver.get(url);
}

protected void isLoaded() throws Error {
    assertTrue("a title",
        driver.getTitle().equals(title));
}

// Open the Page
page.get();
```

Ejercicio

```
o
9  public class LoadableComponentTest {
10
11⊕   /*
12     * Create Wikipedia create account page with Loadable component to run this test
13     * */
14
15
16  private static WebDriver driver;
17
18⊕ @BeforeClass
19  public static void setUp() {
20      driver = new FirefoxDriver();
21  }
22
23⊕ @Test
24  public void testCreateAccount() {
25
26      WikipediaCreateAccountPage page = new WikipediaCreateAccountPage(driver);
27
28      page.get();
29      page.submitUser("usuario", "12345678", "nouser@gmail.com", "sdfsadfasd");
30  }
31
32⊕ @AfterClass
33  public static void tearDown() {
34      driver.quit();
35  }
36 }
```

Ejercicio

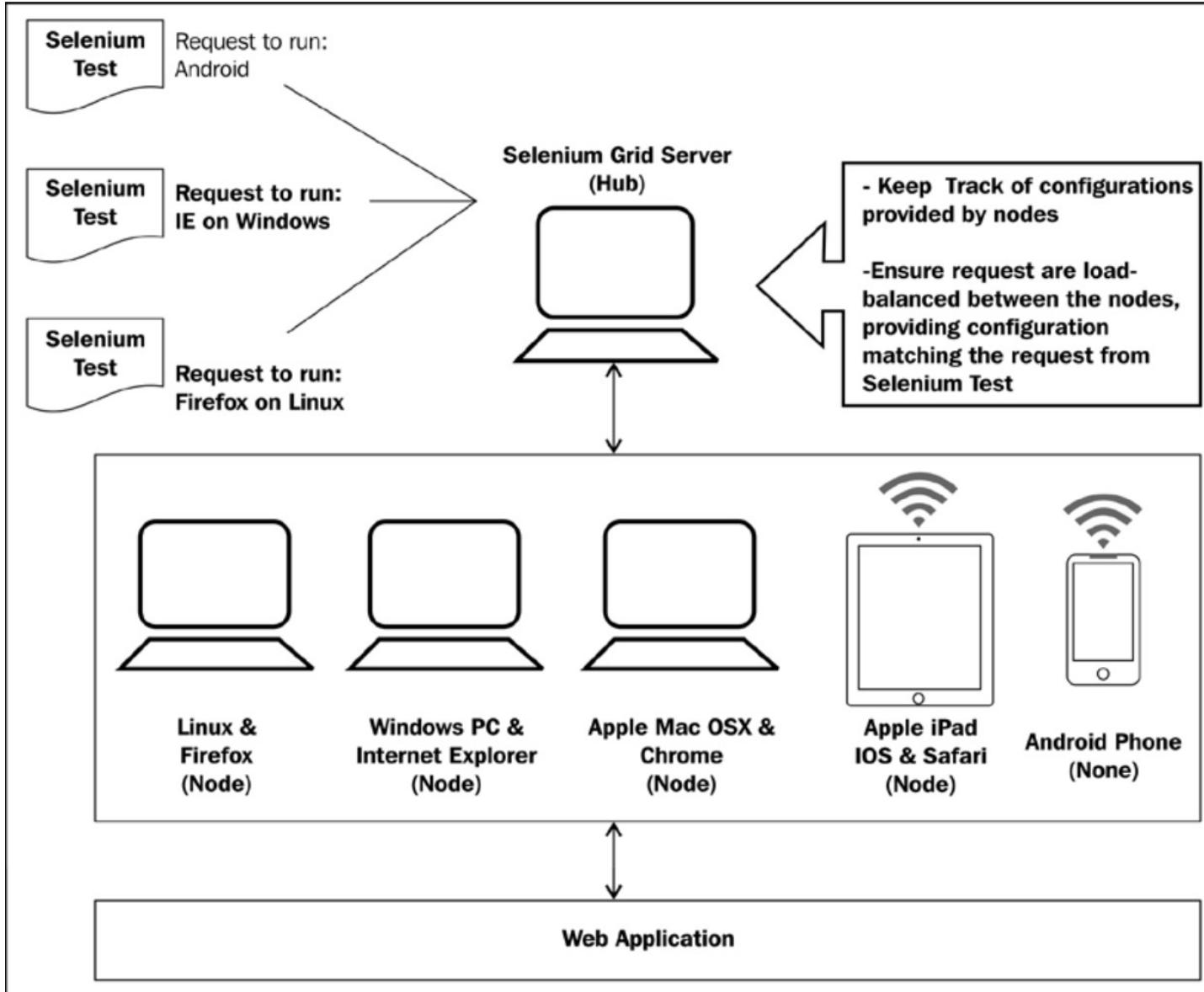
```
 9
10 public class WikipediaCreateAccountPage extends LoadableComponent<WikipediaCreateAccountPage> {
11
12     @FindBy(id = "wpName2")
13     private WebElement usernameElement;
14
15     @FindBy(id = "wpPassword2")
16     private WebElement passElement;
17
18     @FindBy(id = "wpRetype")
19     private WebElement confirmPassElement;
20
21     @FindBy(id = "wpName2")
22     private WebElement emailElement;
23
24     @FindBy(id = "mw-input-captchaWord")
25     private WebElement captchaElement;
26
27     @FindBy(id = "wpCreateaccount")
28     private WebElement submitButton;
29
30     private String url = "https://es.wikipedia.org/w/index.php?title=Especial:Crear_una_cuenta";
31     private WebDriver driver;
32
33     public WikipediaCreateAccountPage(WebDriver driver) {
34
35         this.driver = driver;
36         PageFactory.initElements(this.driver, this);
37     }
38
39     @Override
40     protected void load() {
41         this.driver.get(url);
42     }
43
44     @Override
45     protected void isLoaded() throws Error {
46         assertEquals("Crear una cuenta - Wikipedia, la enciclopedia libre", driver.getTitle());
47     }
48
49     public void submitUser(String username, String password, String email, String captcha) {
50
51         usernameElement.clear();
52         usernameElement.sendKeys(username);
53
54         passElement.clear();
55         passElement.sendKeys(password);
56
57         confirmPassElement.clear();
58         confirmPassElement.sendKeys(password);
59
60         captchaElement.clear();
61         captchaElement.sendKeys(captcha);
62
63         submitButton.click();
64     }
65
66 }
```

Selenium Grid

- Selenium Grid es la herramienta de Selenium por la que vamos a poder configurar todos nuestros test para que se ejecuten paralelamente en todos los navegadores.
- Selenium primero levanta un hub, y a este hub se le añaden todos los nodos (navegadores).
- Una vez levantado el hub y todos los nodos añadimos, configuraremos nuestros test para que se lancen a través de él.
- Para comprobar la compatibilidad de las versiones miramos en el ChangeLog de Selenium:

<https://raw.githubusercontent.com/SeleniumHQ/selenium/master/java/CHANGELOG>

Selenium Grid



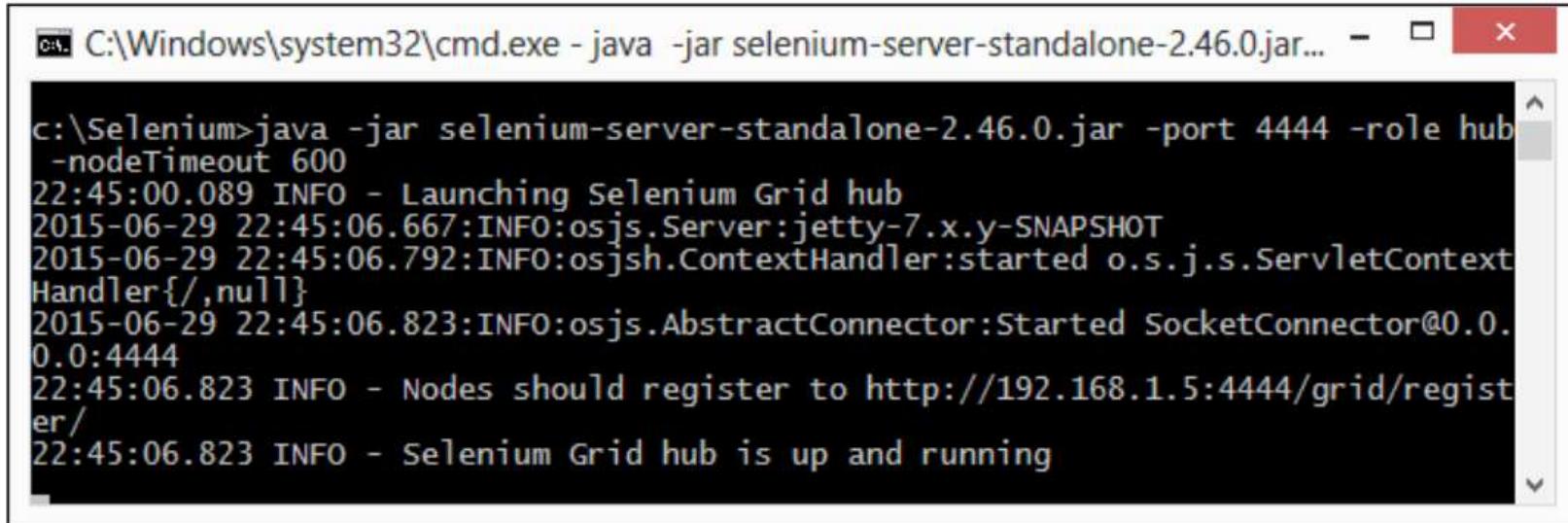
Selenium Grid

- Configuración del hub.
- Primero tenemos que descargarnos el .jar para poder ejecutarlo.
- Para que nos funcione, con nuestro Firefox 30, vamos a descargar la versión 2.53.0
- Página de descarga: <http://docs.seleniumhq.org/download/>
- Versiones: <https://selenium-release.storage.googleapis.com/index.html>
- Ejecutamos el siguiente comando en consola

```
java -jar selenium-server-standalone-2.53.0.jar -port 4444 -role hub
```

Selenium Grid

- Una vez levantado el hub, deberíamos ver algo así



```
C:\Windows\system32\cmd.exe - java -jar selenium-server-standalone-2.46.0.jar... - □ ×  
c:\Selenium>java -jar selenium-server-standalone-2.46.0.jar -port 4444 -role hub  
-nodeTimeout 600  
22:45:00.089 INFO - Launching Selenium Grid hub  
2015-06-29 22:45:06.667:INFO:osjs.Server:jetty-7.x.y-SNAPSHOT  
2015-06-29 22:45:06.792:INFO:osjsh.ContextHandler:started o.s.j.s.ServletContext  
Handler{/,,null}  
2015-06-29 22:45:06.823:INFO:osjs.AbstractConnector:Started SocketConnector@0.0.  
0.0:4444  
22:45:06.823 INFO - Nodes should register to http://192.168.1.5:4444/grid/regis  
ter/  
22:45:06.823 INFO - Selenium Grid hub is up and running
```

- Si accedemos a la dirección dónde lo hemos levantado



Selenium Grid

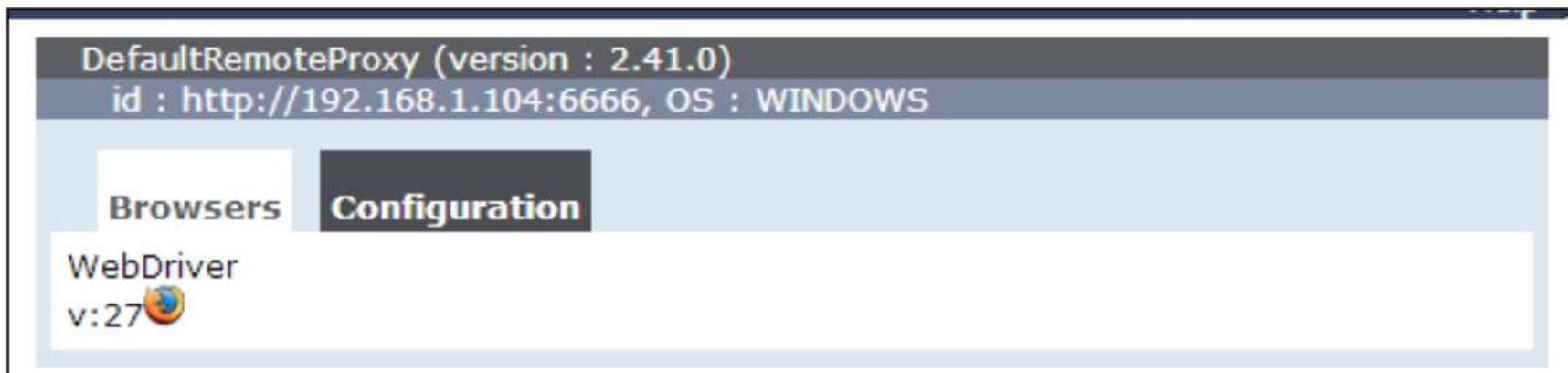
- A la hora de añadir los nodos es muy importante tener en cuenta las versiones, tanto del hub que hemos levantado como la de los navegadores.
- Nos puede pasar que tengamos una versión de selenium-server-standalone que no sea compatible con la del nodo que estamos añadiendo.
- No nos dará ningún error, pero a la hora de ejecutar el test, no nos va a funcionar.

Selenium Grid

- Ahora añadimos los nodos a ese hub
- Para añadir Firefox

```
java -jar selenium-server-standalone-2.53.0.jar -role webdriver -browser  
"browserName=firefox,version=30.0.5,maxinstance=2,platform=WINDOWS" -hubHost localhost –port 6666
```

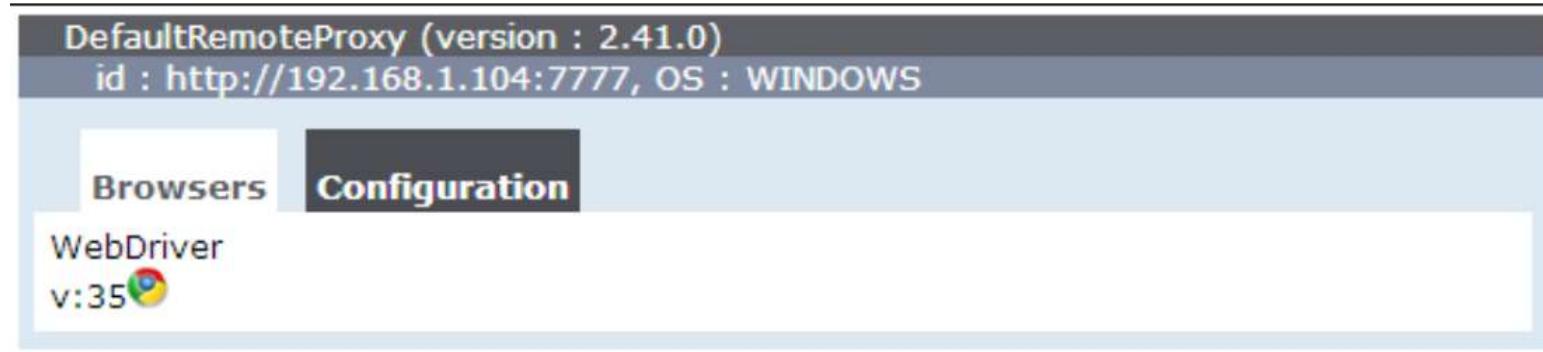
- En nuestra dirección del hub deberíamos ver



Selenium Grid

- Ahora añadimos los nodos a ese hub
- Para añadir Chrome

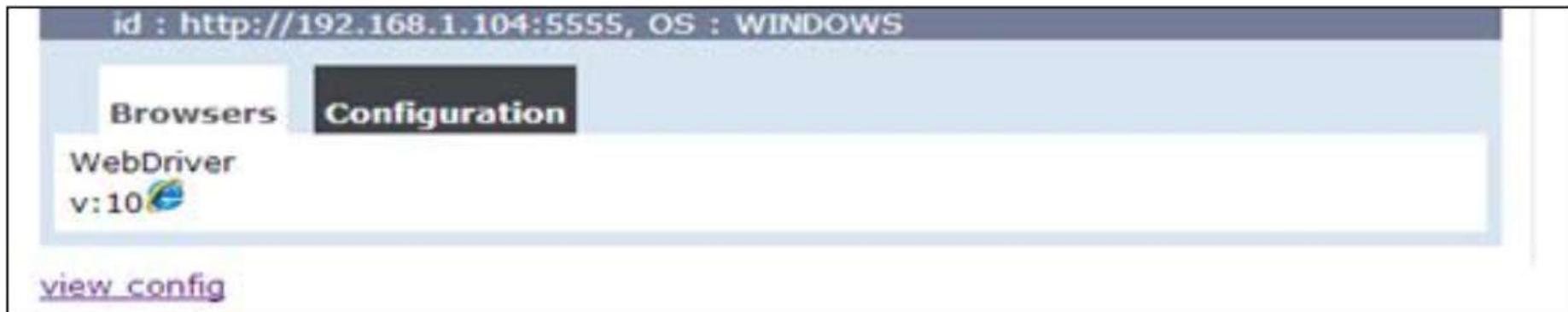
```
java -Dwebdriver.chrome.driver="C:\SeDrivers\chromedriver.exe" -jar
selenium-server-standalone-2.53.0.jar -role webdriver -browser
"browserName=chrome,version=35,maxinstance=2,platform=WINDOWS" -
hubHost localhost -port 7777
```



Selenium Grid

- Ahora añadimos los nodos a ese hub
- Para añadir Internet Explorer

```
java -Dwebdriver.ie.driver="C:\SeDrivers\IEDriverServer.exe" -jar
selenium-server-standalone-2.46.0.jar -role webdriver -browser
"browserName=internet
explorer,version=10,maxinstance=1,platform=WINDOWS" -hubHost
192.168.1.103 –port 5555
```



Selenium Grid

- Ahora añadimos los nodos a ese hub
- Para añadir Safari

```
java -jar selenium-server-standalone-2.47.1.jar -role webdriver -  
browser "browserName=safari,version=7,maxinstance=1,platform=MAC" -  
hubHost 192.168.1.104 -port 8888
```

Selenium Grid

- Una vez añadimos estos cuatro navegadores, nuestro hub quedaría así.

The screenshot shows the Selenium Grid Console interface with four browser instances listed:

- Top Left:** DefaultRemoteProxy (version : 2.41.0), id : http://192.168.1.104:5555, OS : WINDOWS. Browser: WebDriver v:10 (IE icon)
- Top Right:** DefaultRemoteProxy (version : 2.41.0), id : http://192.168.1.104:7777, OS : WINDOWS. Browser: WebDriver v:35 (Chrome icon)
- Bottom Left:** DefaultRemoteProxy (version : 2.41.0), id : http://192.168.1.104:6666, OS : WINDOWS. Browser: WebDriver v:27 (Firefox icon)
- Bottom Right:** DefaultRemoteProxy (version : 2.41.0), id : http://192.168.1.100:8888, OS : MAC. Browser: WebDriver v:7 (Safari icon)

At the bottom left, there is a link labeled [view config](#).

Selenium Grid

- Con nuestro hub ya configurado, vamos a crear nuestro test para que se lance sobre él.
- Dependencias del proyecto

```
<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>2.53.0</version>
    <scope>test</scope>
</dependency>

<!-- https://mvnrepository.com/artifact/junit/junit -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>
```

Selenium Grid

- Configuramos las características de nuestro test, para que se ejecuten los navegadores. Si no indicamos ninguna se ejecutará en todos.

```
DesiredCapabilities caps = new DesiredCapabilities();
caps.setPlatform(org.openqa.selenium.Platform.MAC);
caps = DesiredCapabilities.firefox();
caps.setVersion("30.0");
```

- Configuramos el WebDriver para que se ejecute contra nuestro hub

```
driver = new RemoteWebDriver(new URL("http://localhost:4444/wd/hub"), caps);
driver.get("http://www.google.com");
```

Selenium Grid

- El resto del test lo hacemos como hasta ahora.

```
@Test  
public void testTitle() throws InterruptedException {  
    assertEquals("Google", driver.getTitle());  
}  
  
@After  
public void afterTest() {  
    driver.quit();  
}
```

SouceLabs

- Herramienta de pago, que nos permite crear los test y ejecutarlos en el cloud, para el navegador que queramos.
- Sin preocuparnos de las versiones ni tener que montar nada.
- Nos registramos en la página, disponen de una versión de prueba de 14 días gratuita.
- Una vez registrados, obtendremos una clave de acceso.

Access Key

 Show

Regenerate Access Key

Warning: Regenerating your access key will require update your access key value throughout your configuration.

Commands containing your old access key will fail.

SouceLabs

- Creamos nuestros test y los lanzamos contra su URL

```
DesiredCapabilities caps = new DesiredCapabilities();
caps.setPlatform(org.openqa.selenium.Platform.MAC);
caps = DesiredCapabilities.chrome();

driver = new RemoteWebDriver(new URL(MessageFormat.format("http://{}:{}@ondemand.saucelabs.com:80/wd/hub", "<username>", "<acces-key>")), caps);
driver.get("http://www.google.com");
```

- Definimos nuestros métodos de Test cómo siempre.
- Al ejecutarse lo hará contra sus servidores sin que nosotros hagamos nada.
- El test se ejecutará con normalidad, obteniendo los resultados desde el mismo entorno.

SouceLabs

- Dentro del dashboard de su página, podemos ver cómo se han ejecutado los test, los comando realizados a la web, incluso los videos del navegador ejecutando los tests.

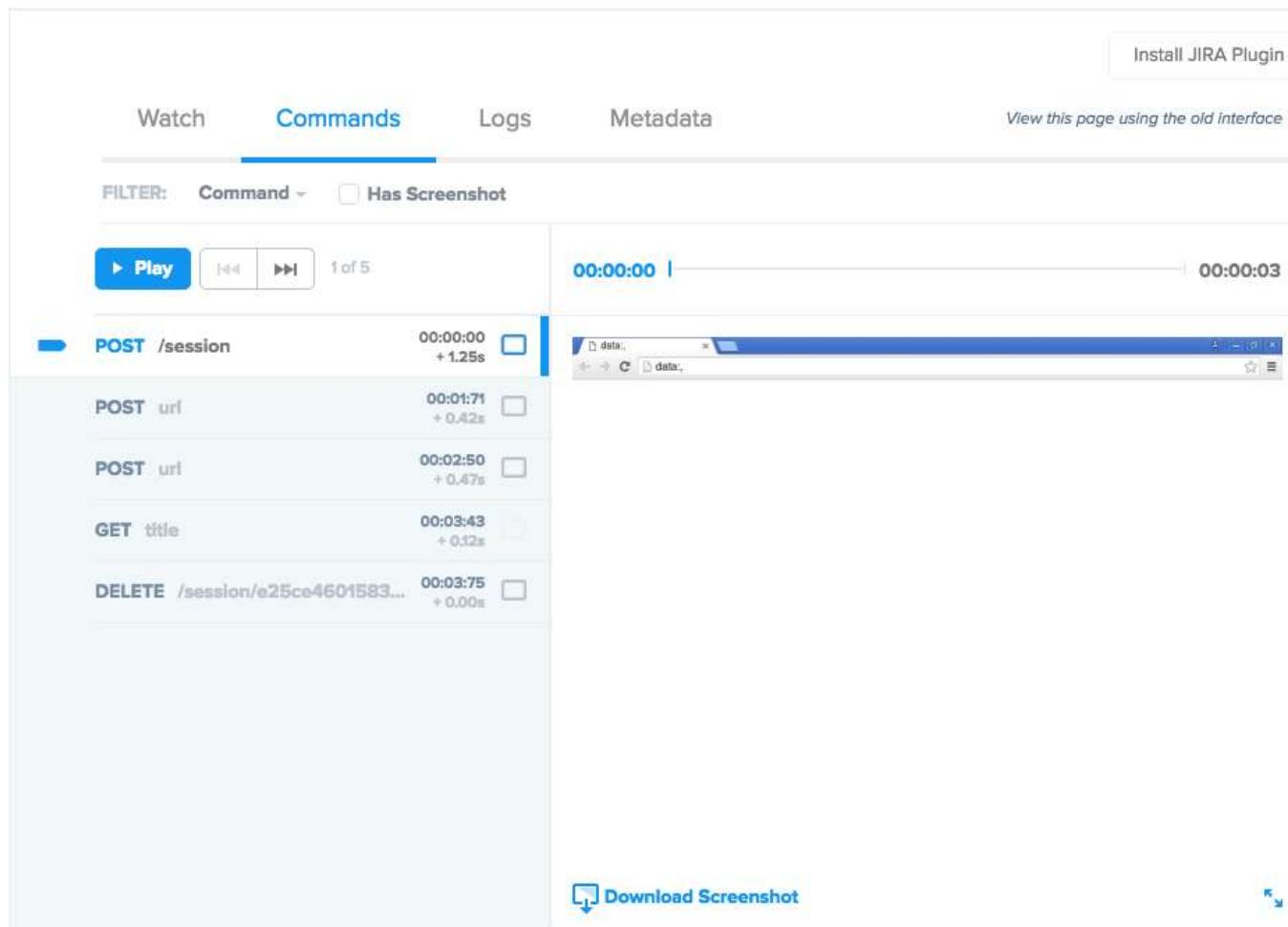
The screenshot shows the SouceLabs dashboard interface. At the top, there is a welcome message: "Welcome to the new interface, optimized for CI." Below this, there are three navigation tabs: "Automated Builds", "Automated Tests" (which is currently selected), and "Manual Tests". The date "Monday, Nov 7th" is displayed below the tabs. The main content area lists three test jobs:

Job Status	Job ID	Started By	Browser Version	Test Count	Run Duration	Result
!	Unnamed Job 24f74476fe714e3e90a51502a8a6f00e	started 9 minutes ago by @borjaigartua	10.10	8	ran for 5m 14s	Error
?	Unnamed job e25ce46015834f67b13f11c3da1c2b9f	started 9 minutes ago by @borjaigartua	*	48	ran for 6s	Complete
?	Unnamed job 3a3b4e750c9145c2b01ffc7f4cf70849	started 11 minutes ago by @borjaigartua	*	30	ran for 9s	Complete

At the bottom center of the dashboard, there is a button labeled "See all in Archives →".

SouceLabs

- Dentro del dashboard de su página, podemos ver cómo se han ejecutado los test, los comando realizados a la web, incluso los videos del navegador ejecutando los tests.

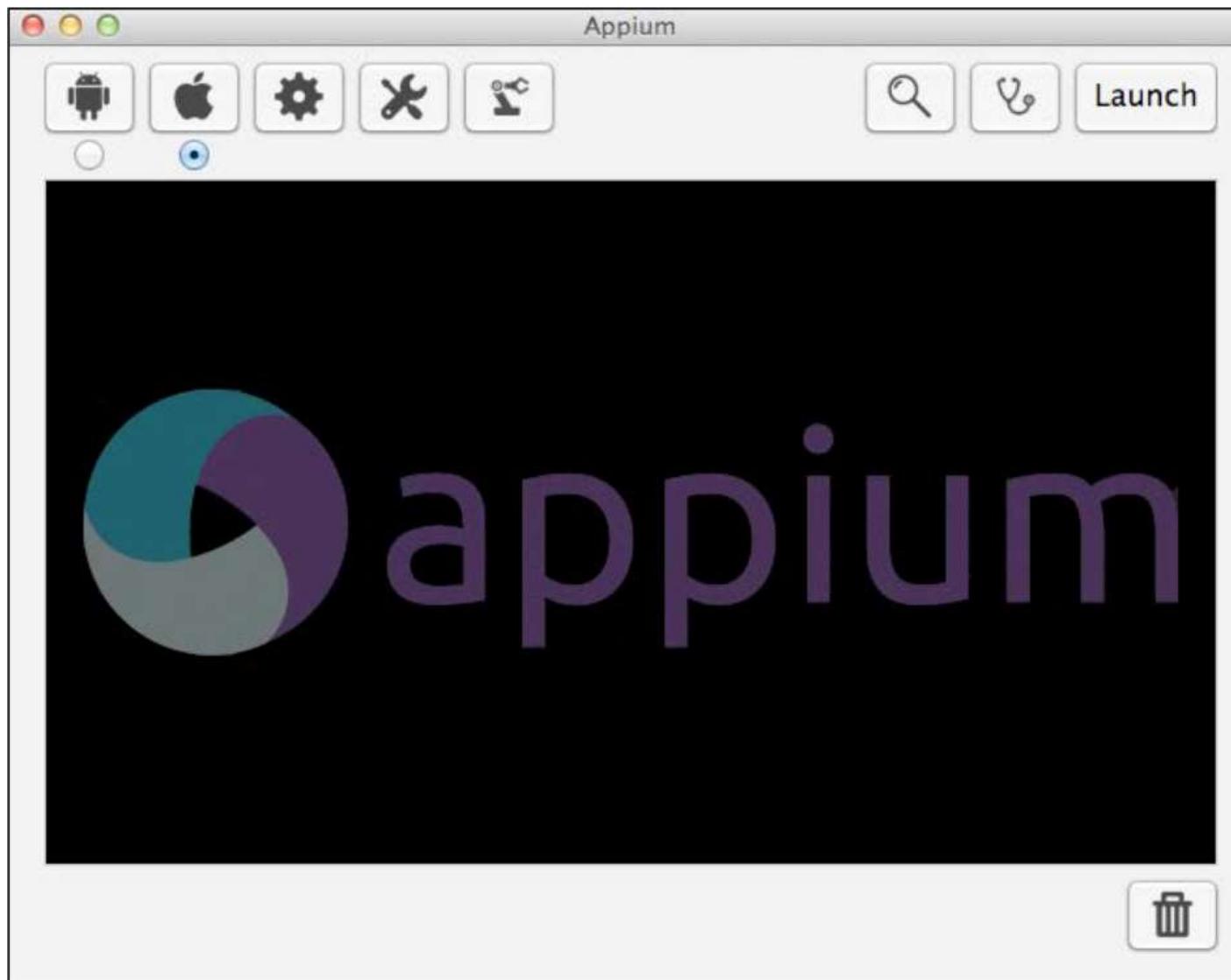


Appium

- Appium es un framework de programación de código abierto de testeo automatizado para el uso con aplicaciones nativas e híbridas.
- Conduce a aplicaciones iOS y Android usando Selenium WebDriver
- Requisitos
 - iOS
 - Mac OSX 10.7+
 - XCode 4.5+ y con las herramientas de línea de comandos
 - Android
 - Mac OSX 10.7+ o Windows 7+ o Linux
 - Android SDK ≥ 16 (SDK < 16 en el modo del Selendroid)

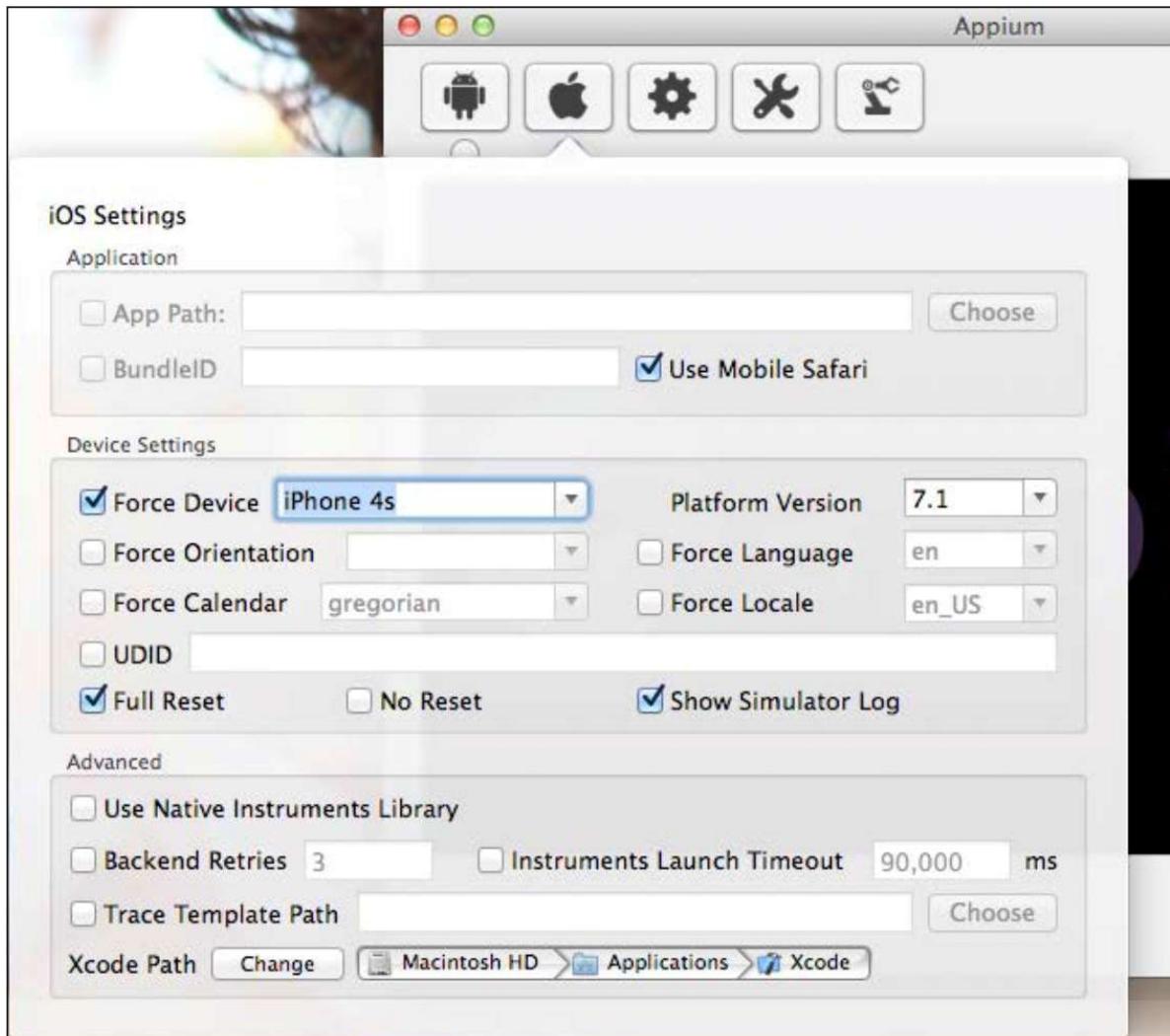
Appium

- Una vez descargado e instalado Appium, podemos elegir qué sistema vamos a usar para correr nuestros test en él.



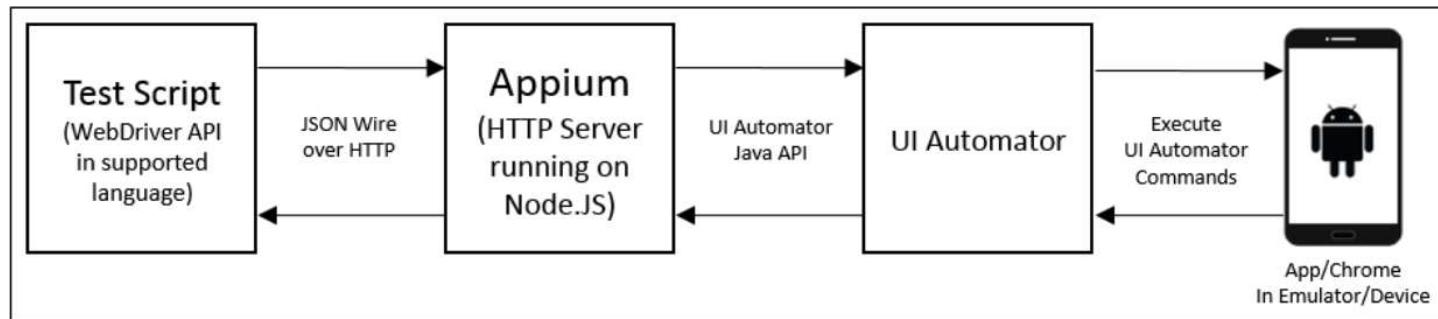
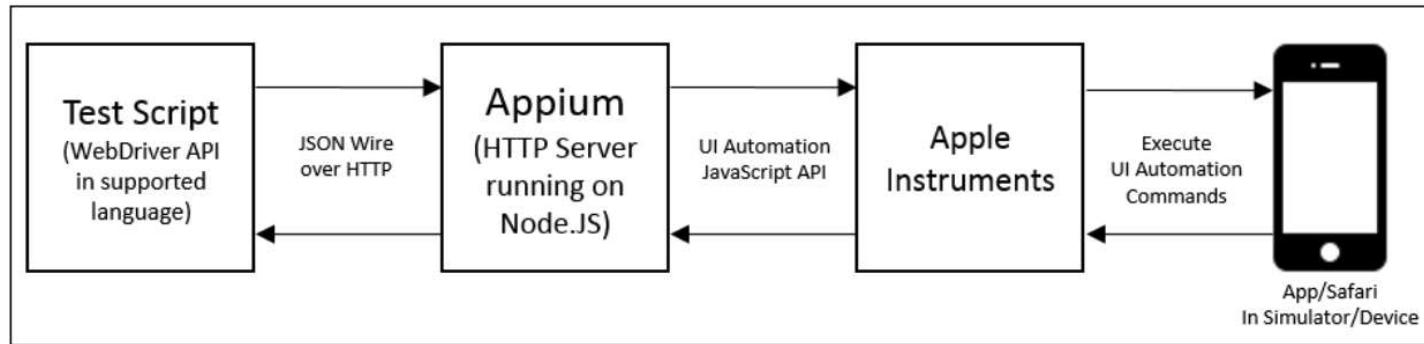
Appium

- Vamos a poder testar tanto aplicaciones nativas, híbridas, como webs dentro de nuestro móvil.



Appium

- Tanto en iOS como en Android, Appium se comunica con las herramientas nativas del sistema operativo.



Appium

- Para ejecutar nuestros test, primero añadimos las dependencias.

```
<dependency>
    <groupId>io.appium</groupId>
    <artifactId>java-client</artifactId>
    <version>3.1.0</version>
    <scope>test</scope>
</dependency>
```

- En nuestro test .
- Indicámos las características de la plataforma.

Appium

- En nuestro test .
- Indicámos las características de la plataforma.

```
@Before
public void setUp() throws Exception {

    // Set the desired capabilities for iOS- iPhone 6
    DesiredCapabilities caps = new DesiredCapabilities();
    caps.setCapability("platformName", "iOS");
    caps.setCapability("platformVersion", "8.4");
    caps.setCapability("deviceName", "iPhone 6");
    caps.setCapability("browserName", "safari");

    // Create an instance of IOSDriver for testing on iOS platform
    // connect to the local Appium server

    driver = new IOSDriver<WebElement>(new URL("http://127.0.0.1:4723"), caps);
    // Open the BMI Calculator Mobile Application
    driver.get("http://google.es");
}
```

Appium

- Los test los creamos como siempre, gracias a Appium, podemos generarlos y que corran bajo una app.
- Para realizar los test de Android lo haríamos de la misma manera, pero cambiando las DesiredCapabilities.