En este tutorial se mostrará cómo crear una prueba automatizada con el lenguaje de programación Java y el framework de pruebas automatizadas Selenium Webdriver. Para el tutorial, utilizaremos el IDE Intellij IDEA.

## Objetivo de la prueba

La prueba consiste en abrir un sitio web en el navegador de Internet, en este caso, el sitio será http://blogs.masterhacks.net. Después, se buscará la barra de búsqueda del blog y se escribirá en ella.

## Desarrollo de la prueba

Una vez creado nuestro paquete «*base*» en Intellij, crearemos un Setup para establecer qué página web se va a abrir utilizando webdriver. Si no recuerdas cómo se hace esto, puedes ver este tutorial y observar los detalles sobre la inicialización de la prueba.

Se puede observar que se está utilizando TestNG, un framework que nos permite agregar más funcionalidades al código para mejorar los procesos. Para utilizar TestNG en la prueba, es necesario agregar la dependencia en el archivo *pom.xml* en el IDE.

```
<dependency>
            <groupId>org.testng</groupId>
            <artifactId>testng</artifactId>
            <version>6.14.3</version>
            <scope>test</scope>
        </dependency>
```

```
    </dependencies>
</project>
```

Con esto, ya se puede utilizar las clases *@BeforeClass y @AfterClass,* que se utilizan en el archivo Base para inicializar y finalizar la prueba.

Ahora, se crea otra clase Java, en este caso con el nombre *SearchForm*, donde se escribirá el código para la prueba requerida.

Para acceder al elemento de barra de búsqueda, vamos a crear un Webelement que busque dicho elemento a partir del id del CSS. Para conocer este ID, se da clic secundario sobre el elemento del menú y en *Inspeccionar*.

Como se observa en la imagen, el ID que se va a utilizar es *«search-toggle»*. El código para posicionarnos sobre este elemento con Selenium es:

```
WebElement cliclink = driver.findElement(By.id("search-
toggle"));
```

Se crea la variable *cliclink* para almacenar la información del *findelement*, para posteriormente, dar clic en el elemento utilizando dicha variable.

Aquí se utilizarán tres elementos más, uno para situarse en la barra de búsqueda para enviar el texto a buscar y otro para dar clic en el botón que realizará la búsqueda. Para esto, será un poco complicado encontrar el nombre del elemento, y por el diseño del sitio web, no se podrá utilizar id, en este caso, será mediante *CSSSelector*. Para esto, en la misma ventana de inspeccionar, se puede posicionar el cursor sobre la barra de búsqueda para obtener el selector CSS.

El código completo del test queda de la siguiente forma:

```
@Test
    public void Buscar(){

        WebElement cliclink =
driver.findElement(By.id("search-toggle"));
        cliclink.click();
        WebElement escribir =
driver.findElement(By.cssSelector("input.search-field"));
        escribir.sendKeys("arduino");
        driver.findElement(By.cssSelector("button.search-
submit")).click();
    }
}
```

Como se puede observar en el código, se creó otra variable llamada *«escribir»,* con la búsqueda del elemento de la barra de búsqueda mediante selector CSS. Posteriormente, se utiliza *«sendkeys»* para enviar la frase que se va a buscar. Para realizar la búsqueda, se utiliza el botón *«search-submit»,* cuyo Selector CSS es *«button.search-submit»*. Aquí se puede notar que no se utilizó una variable para la búsqueda del elemento, ya que es otra

forma de realizar la prueba sin crear variables, pues para esta prueba, no resultan tan necesarias, así que se puede modificar el código eliminando las dos variables anteriores.

Al ejecutar el test, se ejecutará la inicialización del archivo base, donde se abrirá el navegador y el sitio web, posteriormente, se ejecuta el test y al terminar, se ejecuta la clase @AfterClass, igual en el archivo base, que lo que hace es cerrar la ventana del navegador.

Si tienes alguna duda o comentario, déjalo aquí abajo!