

En esta ocasión aprenderemos a generar XPath y CSS personalizados desde los atributos HTML, identificaremos objetos usando los Locators de ambos y finalizaremos con un par de ejercicios en los que veremos la integración de Sikuli con Selenium y WebDriver.

No es objeto de este tutorial profundizar en Sikuli aunque sí se darán unas pinceladas en los comentarios del código para entender mejor los ejercicios.

1. Cómo generar XPath personalizados desde los atributos HTML.

Un XPath como hemos anteriormente es la ruta entre las etiquetas html que debemos seguir para identificar a un elemento concreto de la web.

El formato para generar un XPath personalizado es:

```
//TagName[@attribute='Value']
```

```
//TagName[@attribute="Value"]
```

Ambos casos son válidos, con la única diferencia de que en el segundo se ha utilizado el caracter de escape para poder usar las comillas dobles en Java.

Continuando con el ejemplo que venimos usando desde los post anteriores y en la ventana principal, donde se nos solicita ingresar nuestro correo y password, vamos a generar el XPath personalizado para el objeto que nos solicita el email.

A screenshot of a login interface. It features two input fields: 'Correo electrónico o teléfono' (Email or phone) and 'Contraseña' (Password). The first field is highlighted with a red rectangle. To the right of the password field is an 'Entrar' (Login) button. Below the fields is a link that says '¿Has olvidado los datos de la cuenta?' (Forgot your account details?).

Echando un vistazo al código del objeto desde la Herramientas para el Desarrollador observamos:

```
sole Sources Network Performance Memory Application Security Audits AdBlock
td>
<input type="email" class="inputtext" name="email" id="email" data-testid="royal_email" style="background-image: url('data:image/png;
base64,iVBORw0KGgoAAAANSUuEUGAAACAAAAAAYAAABzenr0AAAAAXNSR0IArs4c6QAAAJZJREFUWAntV7vKI1EMPT5d0y/
NID9WIoKKAsiWqFFmLra4j2IoKPsL24YKwFD7Ai+ASCLY2yNioWgjcKfzLrCn46XSKNXDmJPkyWd0ZnCE+CdfuP3G9RCxVhHrDapFNVUuZYXr1YX1+anmlwUv3D9xPVu+YEFFRNeqCX
Ku6tf6i2IALXGMDvVvLS+EPg/+tAuVwMuyGR+/tHsgUCiKdThHgCs/
egosFouIRCIiEAjoY9n2+XwikUgIp9PJpQ1itVpFLBYTfr9f63608zs6GAzCeDwGKb1eD1wuF+OYARqNBpzPZzVkuVxCNpu9wuPx0EynU5kCWq0W2Gw2jhk0h1Cr1dgmVmyMRI0+USrNZpNx
PD/
p5n29XoPb7VZjTCYTTCTYxqRSqV04h54AD6HX6xXJ2PKmTbLcJn35fJ51qWbXkUqLVBm7KEKhkIR41+Zg50VhAofDQ5BjPS72+z37tDo7Udntdqopdy1G+qP7CGMC2+1WdLtd8l0JniHb7Xb
7h1Set8CjU2Nms5kYDAYCLxVtDunT7nw+Ho8H0p0YDdgs9lAv4Hs9nM0N4EpVIJ5v050og0M+Fw+ApXFAX6/T4cj0dYrVZQrVavcP0MEJGrALJpammg7mH5S3A4nuJ2u/
```

Nuestro **TagName** es *input*, como **attribute** podemos seleccionar cualquiera de los disponibles en color marrón. Seleccionamos el *id*. Por último, añadimos el **Value** (valor) del atributo seleccionado. Por tanto, nuestro **XPath personalizado** queda:

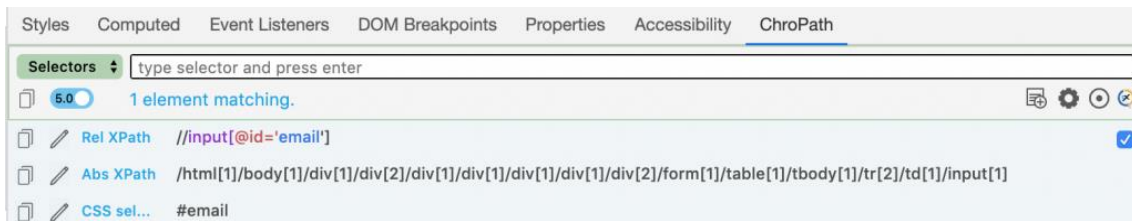
```
//input[@id='email']
```

Otras variantes validas también pueden ser:

```
//input[@class='email']
```

```
//input[@name=\»email\»]
```

Si usamos ChroPath, la extensión del navegador que nos facilita la tarea podemos visualizar que el XPath es el indicado en el primer ejemplo. Además, podemos visualizar la ruta absoluta del objeto:



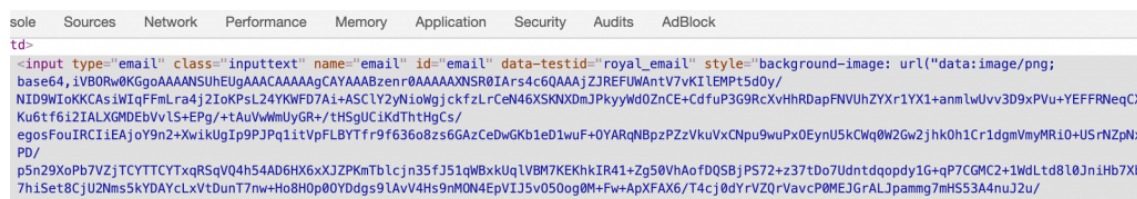
2. Cómo generar CSS personalizados desde los atributos HTML

Una vez que se conoce como generar el XPath personalizado generar el CSS no tiene mucho misterio, este se hace de la misma forma pero pierde la @:

//TagName[attribute='Value']

//TagName[attribute="Value"]

Continuando con el ejemplo anterior de Facebook:



Comprobamos que generar el CSS no tiene mucha historia:

//input[id='email']

3. Identificando objetos con Texto usando XPath Locators y LinkText

A veces nos interesará buscar objetos por su texto o bien por un enlace que tengan asignado para estas situación podremos encontrar el objeto con:

//*[text()='Texto_a_buscar']

Sólo se recomienda utilizar cuando no se tengan atributos. Tened en cuenta que estáis dependiendo de un texto variable.

En caso de un enlace que se quiera buscar podemos usar el Locator LinkText. Volviendo al ejemplo de Facebook, suponiendo que queramos hacer click sobre la frase «¿Has olvidado los datos de la cuenta?» (que es un enlace), se utilizará:

`driver.findElement(By.linkText("¿Has olvidado los datos de la cuenta?")).click();`



4. Identificando objetos con CSS Selectors Locators

Ya vimos anteriormente que la estructura de que debe mantener nuestro CSS Locator personalizado debe ser:

TagName[attribute='Value']

con la particularidad de que en CSS se hace más simple aún al no ser necesario indicar el TagName:

[attribute='Value']

En el ejemplo anterior de Facebook:

[class='inputtext']

Con lo cual si queremos introducir de forma automatizada nuestro email en la caja correspondiente podríamos hacerlo así:

```
public static void main(String[] args) {  
  
    System.setProperty("webdriver.chrome.driver","/Users/Fbogas/Desktop/Sikuli/chromedriver");  
  
  
  
  
    WebDriver driver = new ChromeDriver();  
  
    driver.get("http://facebook.com"); //URL in the browser  
  
    driver.findElement(By.cssSelector("[class='inputtext']")).sendKeys("micorreo@autentia.com");  
  
}
```

5. Ejercicio Práctico I. Integración de Sikuli con Selenium y WebDriver

Para poder realizar el siguiente ejercicio se puede utilizar tanto IntelliJ como SikuliX, siendo con este último una automatización mucho más rápida (apenas lleva 2 minutos).

En nuestro caso vamos a resolverlo con IntelliJ empleando las librerías tanto de Selenium como de Sikuli y veremos que ambos se integran perfectamente sin generar conflictos.

El ejercicio consiste en acceder a una Web aceptando las Cookies y cambiando la configuración de la misma a modo oscuro. La web en cuestión es: www.finofilipino.org

Previamente, vamos a capturar las imágenes que se van a utilizar en el proceso de automatización (emplead para ello la capturadora de SikuliX o bien cualquier herramienta para el caso ya que las nativas de Windows y OSX suelen dar problemas). Y posteriormente se almacenan en una carpeta de fácil acceso.

En mi caso sólo necesito las siguientes:



El código para automatizar el proceso es:

```
package com.autentia.sikulix;
```

```
import org.openqa.selenium.WebDriver;
```

```
import org.openqa.selenium.chrome.ChromeDriver;
```

```
import org.sikuli.basics.Settings;
```

```
import org.sikuli.script.FindFailed;
```

```
import org.sikuli.script.Pattern;
```

```
import org.sikuli.script.Screen;
```

```
public class Sikuli {
```

```
public static void main(String[] args) throws FindFailed, InterruptedException {
```

```
//Primero creamos un objeto Screen para interactuar con la pantalla
```

```
Screen screen = new Screen();
```

```
//A continuación generamos un patrón por cada imagen con la que vamos a interactuar
```

```
//en nuestro caso con la imagen ACEPTO, para aceptar las Cookies y la imagen de la
```

```
//bombilla que cambiara al modo oscuro.
```

```
Pattern pattern = new Pattern("/Users/Fbogas/Desktop/Images/acepto.png");
```

```
Pattern pattern2 = new Pattern("/Users/Fbogas/Desktop/Images/bombilla.png");
```

```
//Creamos nuestro WebDriver de Chrome
```

```
System.setProperty("webdriver.chrome.driver","/Users/Fbogas/Desktop/Sikuli/chromedriver");
```

```
WebDriver driver = new ChromeDriver();
```

```
//Accedemos a la web indicada
```

```

driver.get("https://finofilipino.org/");

System.out.println(driver.getTitle());

System.out.println(driver.getCurrentUrl());

System.out.println(driver.getPageSource());

//Le indicamos que haga click en el primer patrón, la imagen de ACEPTO

screen.click(pattern);

Settings.MoveMouseDelay = 3; //Da a la aplicación 3 segundos antes de hacer la siguiente acción

//Hacemos doble click sobre la imagen de la bombilla, patrón 2 y cambiamos al modo oscuro.

screen.doubleClick(pattern2);

}

}

```

6. Ejercicio Práctico II. Acceder a la cuenta de Gmail y borrar la carpeta Spam

En este otro ejercicio lo que se pide es automatizar el borrado de la carpeta Spam de una cuenta de correo de Gmail utilizando Sikuli y Selenium bajo el IDE de IntelliJ.

Para este ejercicio las imágenes que se han capturado han sido:



El lector puede detenerse aquí e intentar realizar el ejercicio por su cuenta antes de leer la solución.

Para la solución mostrada, se da por hecho que las credenciales de acceso a la cuenta de gmail están almacenadas y por tanto no se deben introducir. Siendo el acceso a la bandeja de entrada del correo directo.

Solución:

```

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.sikuli.basics.Settings;

import org.sikuli.script.FindFailed;

import org.sikuli.script.Pattern;

import org.sikuli.script.Screen;

```

```
import static org.sikuli.script.Mouse.WHEEL_UP;
```

```
public class Sikuli_Gmail {
```

```
    public static void main(String[] args) throws FindFailed, InterruptedException {
```

```
        Screen screen = new Screen();
```

```
        Pattern pattern = new Pattern("/Users/Fbogas/Desktop/SikuliX/Spam/recibidos.png");
```

```
        Pattern pattern2 = new Pattern("/Users/Fbogas/Desktop/SikuliX/Spam/mas.png");
```

```
        Pattern pattern3 = new Pattern("/Users/Fbogas/Desktop/SikuliX/Spam/menos.png");
```

```
        Pattern pattern4 = new Pattern("/Users/Fbogas/Desktop/SikuliX/Spam/spam.png");
```

```
        Pattern pattern5 = new Pattern("/Users/Fbogas/Desktop/SikuliX/Spam/selector.png");
```

```
        Pattern pattern6 = new Pattern("/Users/Fbogas/Desktop/SikuliX/Spam/eliminar.png");
```

```
        System.setProperty("webdriver.gecko.driver","/Users/Fbogas/Desktop/Sikuli/geckodriver")
        ;
```

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

```
        driver.get("http://gmail.com"); //Suponemos que ya están introducidas las
        credenciales.
```

```
        screen.wheel(pattern, WHEEL_UP, 10); //Se va a la imagen recibidos y hace un Scroll
        hasta ver el "Más"
```

```
        screen.click(pattern2); //Hacemos click en el "Más"
```

```
        Settings.MoveMouseDelay = 2; //Da a la aplicación 2 segundos antes de hacer la
        siguiente acción
```

```
        screen.wheel(pattern3, WHEEL_UP, 10); //Ahora que se ha convertido en "Menos"
        buscamos "SPAM"
```

```
        screen.click(pattern4);    //Hacemos click en SPAM
```

```
        Settings.MoveMouseDelay = 1; //Esperamos 1 segundo
```

```
        screen.click(pattern5);    //Hacemos click en el selector
```

```
        Settings.MoveMouseDelay = 1; //Esperamos 1 segundo
```

```
        screen.click(pattern6);    //Hacemos click en "eliminar definitivamente"
```

```

Settings.MoveMouseDelay = 1; //Esperamos un segundo

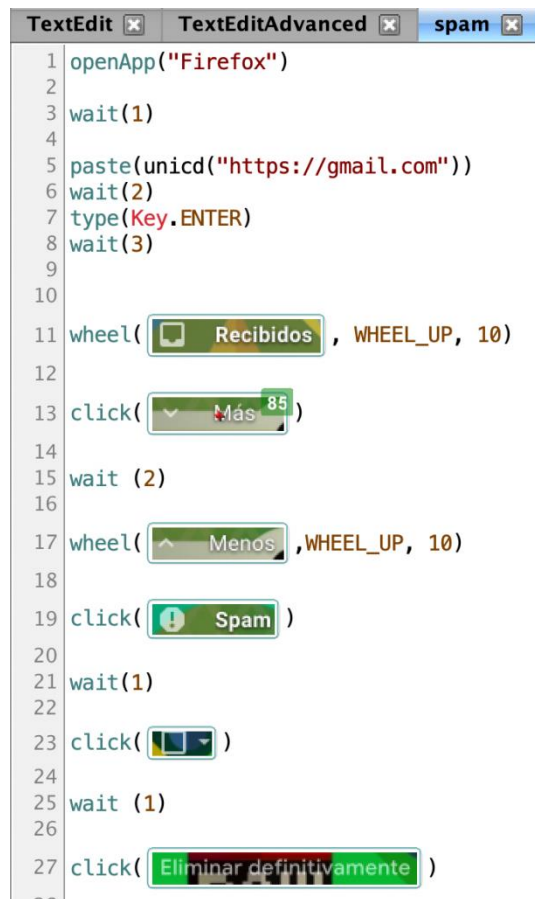
driver.close();           //Cerramos el navegador

}







}

```

Solución desde el IDE Sikulix:



```

1  openApp("Firefox")
2
3  wait(1)
4
5  paste(unicd("https://gmail.com"))
6  wait(2)
7  type(Key.ENTER)
8  wait(3)
9
10
11 wheel(  , WHEEL_UP, 10)
12
13 click(  )
14
15 wait (2)
16
17 wheel(  ,WHEEL_UP, 10)
18
19 click(  )
20
21 wait(1)
22
23 click(  )
24
25 wait (1)
26
27 click(  )
28

```