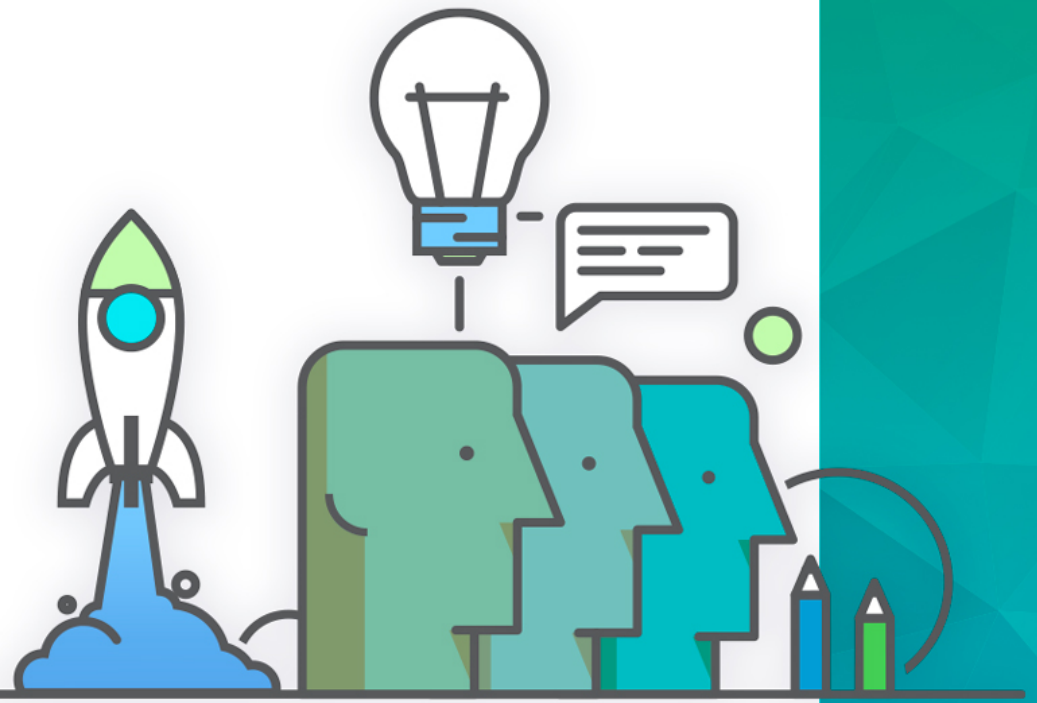
campus
Periferia IT

Periferia IT
Group



Automatización

Módulo 8: Inspección de objetos

Un curso del Área DevOps
Periferia IT Group

Periferia IT
Corp

CONTACT&
BUSINESSIT

COOLEXPRESS

Contenido

- Inspección de objetos
- Localizadores tradicionales
- Localizadores relativos
- Localizadores

INSPECCIÓN DE OBJETOS

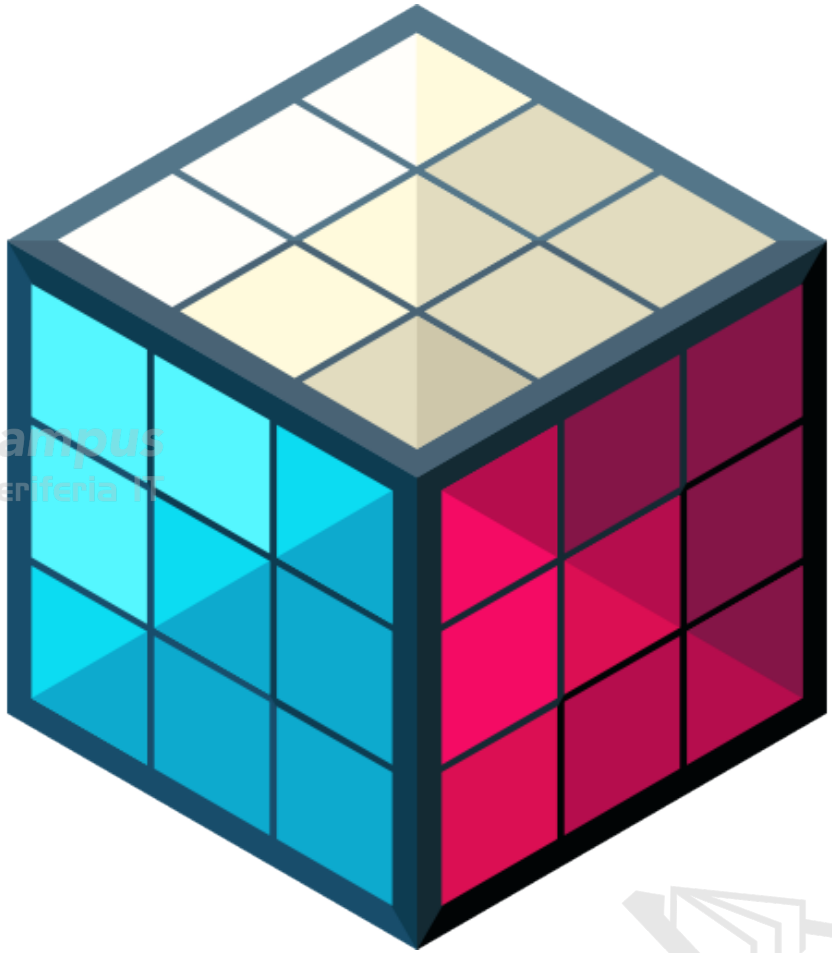
Un localizador es una forma de identificar elementos en una página.

En general, si los ID de HTML están disponibles, son únicos y siempre predecibles, son el método preferido para ubicar un elemento en una página.

Si las ID únicas no están disponibles, un selector de CSS bien escrito es el método preferido para ubicar un elemento.

XPath funciona tan bien como los selectores de CSS.

Las estrategias de selección basadas en linkText y partialLinkText tienen inconvenientes porque solo funcionan en elementos de enlace. Además, llaman a los selectores de XPath internamente



LOCALIZADORES TRADICIONALES

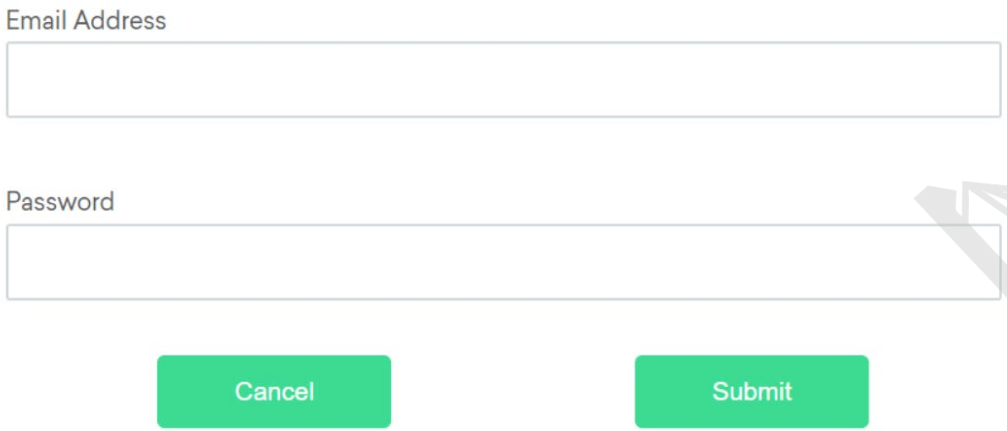
Locator	Descripción
class name	Localiza elementos cuyo nombre de clase contiene el valor de búsqueda (no se permiten nombres de clase compuestos)
css selector	Localiza elementos que coinciden con un selector CSS
id	Localiza elementos cuyo atributo ID coincide con el valor de búsqueda
name	Localiza elementos cuyo atributo NOMBRE coincide con el valor de búsqueda
link text	Localiza elementos de anclaje cuyo texto visible coincide con el valor de búsqueda
partial link text	Localiza elementos de anclaje cuyo texto visible contiene el valor de búsqueda. Si coinciden varios elementos, solo se seleccionará el primero.
tag name	Localiza elementos cuyo nombre de etiqueta coincide con el valor de búsqueda
xpath	Localiza elementos que coinciden con una expresión XPath

LOCALIZADORES RELATIVOS

Estos localizadores son útiles cuando no es fácil construir un localizador para el elemento deseado.

Cómo funciona:

Los métodos de localizador relativo pueden tomar como argumento para el punto de origen, ya sea una referencia de elemento previamente ubicada u otro localizador.



Above (SOBRE):

Si el elemento del campo de texto del correo electrónico no es fácilmente identificable por alguna razón, pero el elemento del campo de texto de la contraseña sí lo es, podemos ubicar el elemento del campo de texto usando el hecho de que es un elemento de "entrada" "arriba" del elemento de la contraseña.

```
By emailLocator = RelativeLocator.with(By.tagName("input")).above(By.id("password"));
```

Below (DEBAJO):

Si el elemento del campo de texto de la contraseña no es fácilmente identificable por alguna razón, pero el elemento del campo de texto del correo electrónico sí lo es, podemos ubicar el elemento del campo de texto usando el hecho de que es un elemento de "entrada" "debajo" del elemento del correo electrónico.

```
By passwordLocator = RelativeLocator.with(By.tagName("input")).below(By.id("email"));
```

Left of (IZQUIERDA DE):

Si el botón de cancelación no es fácilmente identificable por alguna razón, pero el elemento del botón de envío sí lo es, podemos ubicar el elemento del botón de cancelación usando el hecho de que es un elemento de "botón" a la "izquierda" del elemento de envío.

```
By cancelLocator = RelativeLocator.with(By.tagName("button")).toLeftOf(By.id("submit"));
```

Right of(DERECHA DE):

Si el botón de envío no es fácilmente identificable por algún motivo, pero el elemento del botón de cancelación sí lo es, podemos ubicar el elemento del botón de envío utilizando el hecho de que es un elemento de "botón" "a la derecha" del elemento de cancelación.

```
By submitLocator = RelativeLocator.with(By.tagName("button")).toRightOf(By.id("cancel"));
```

LOCALIZADORES

Hay muchas formas de utilizar los localizadores en escenarios muy avanzados.

```
<ol id="vegetables">
 <li class="potatoes">…
 <li class="onions">…
 <li class="tomatoes"><span>Tomato is a Vegetable</span>…
</ol>
<ul id="fruits">
  <li class="bananas">…
  <li class="apples">…
  <li class="tomatoes"><span>Tomato is a Fruit</span>…
</ul>
```

Cuando se llama al método de búsqueda de elemento en la instancia del controlador, devuelve una referencia al primer elemento en el DOM que coincide con el localizador proporcionado.

```
WebElement vegetable = driver.findElement(By.className("tomatoes"));
```

hay dos elementos que tienen un nombre de clase de "tomates", por lo que este método devolverá el elemento en la lista de "verduras".

Una solución es ubicar un elemento con un atributo único que sea un ancestro del elemento deseado.

```
WebElement fruits = driver.findElement(By.id("fruits"));
WebElement fruit = fruits.findElement(By.id("tomatoes"));
```

Para mejorar ligeramente el rendimiento, podemos usar CSS o XPath para encontrar este elemento en un solo comando. Para este ejemplo, usaremos un Selector de CSS

```
WebElement fruit = driver.findElement(By.cssSelector("#fruits .tomatoes"));
```

Obtener referencias a todos los elementos que coincidan con un localizador, en lugar de solo el primero. Los métodos de búsqueda de elementos plurales devuelven una colección de referencias de elementos. Si no hay coincidencias, se devuelve una lista vacía.

```
List<WebElement> plants = driver.findElements(By.tagName("li"));
```

A menudo, obtiene una colección de elementos pero quiere trabajar con un elemento específico, lo que significa que necesita iterar sobre la colección e identificar el que desea.

```
List<WebElement> elements = driver.findElements(By.tagName("li"));

for (WebElement element : elements) {
    System.out.println("Paragraph text:" + element.getText());
}
```

Para rastrear (o) encontrar el elemento DOM que tiene el foco en el contexto de navegación actual.

```
driver.get("http://www.google.com");
driver.findElement(By.cssSelector("[name='q']")).sendKeys("webElement");

// Get attribute of current active element
String attr = driver.switchTo().activeElement().getAttribute("title");
System.out.println(attr);
```

LOCALIZADORES

Acciones de teclado, normalmente esta función es usada en formularios o paginas q no tenga en botón enviar o simplificar en una línea dos eventos diferentes escribir y aceptar

```
WebDriver driver = new FirefoxDriver();
try {
  // Navigate to Url
  driver.get("https://google.com");

  // Enter text "q" and perform keyboard action "Enter"
  driver.findElement(By.name("q")).sendKeys("q" + Keys.ENTER);
} finally {
  driver.quit();
```

Este método se utiliza para verificar si el elemento conectado está habilitado o deshabilitado en una página web. Devuelve un valor booleano, True si el elemento conectado está habilitado en el contexto de navegación actual; de lo contrario, devuelve false .

```
boolean value = driver.findElement(By.name("btnK")).isEnabled();
```

Este método determina si el Elemento al que se hace referencia está Seleccionado o no. Este método se usa ampliamente en casillas de verificación, botones de radio, elementos de entrada y elementos de opción.

Devuelve un valor booleano, True si el elemento al que se hace referencia está seleccionado en el contexto de navegación actual; de lo contrario, devuelve false.

```
boolean value = driver.findElement(By.cssSelector("input[type='checkbox']:first-of-type")).isSelected();
```

Este método Se utiliza para obtener el nombre de etiqueta del elemento al que se hace referencia que tiene el foco en el contexto de navegación actual.

```
String value = driver.findElement(By.cssSelector("h1")).getTagName();
```

Este método Recupera el texto representado del elemento especificado.

```
String text = driver.findElement(By.cssSelector("h1")).getText();
```

Este método las listas de selección tienen comportamientos especiales en comparación con otros elementos.

Los elementos seleccionados pueden requerir bastante código repetitivo para automatizar. Para reducir esto y hacer que sus pruebas sean más limpias, hay una clase Select en el paquete de soporte de Selenium. Para usarlo, necesitará la siguiente declaración

```
import org.openqa.selenium.support.ui.Select;
```

A continuación, puede crear un objeto Select mediante un WebElement que haga referencia a un <select>elemento.

```
WebElement selectElement = driver.findElement(By.id("selectElementID"));
Select selectObject = new Select(selectElement);
```

El objeto Seleccionar ahora le dará una serie de comandos que le permitirán interactuar con un <select>elemento. En primer lugar, hay diferentes formas de seleccionar una opción del <select>elemento.

```
<select>
 <option value=value1>Bread</option>
 <option value=value2 selected>Milk</option>
 <option value=value3>Cheese</option>
</select>
```

LOCALIZADORES

Hay tres formas de seleccionar la primera opción del elemento

```
// Seleccione una <opción> basada en el índice interno del elemento <select>
selectObject.selectByIndex(1);

// Seleccione una <opción> en función de su atributo de valor
selectObject.selectByValue("value1");

// Seleccione una <opción> según su texto
selectObject.selectByVisibleText("Bread");
```

Si desea anular la selección de algún elemento, ahora tiene cuatro opciones:

```
//Anule la selección de una <opción> basada en el índice interno del elemento <select>
selectObject.deselectByIndex(1);

// Anule la selección de una <opción> en función de su atributo de valor
selectObject.deselectByValue("value1");

// Anular la selección de una <opción> en función de su texto
selectObject.deselectByVisibleText("Bread");

// Deseleccionar todos los elementos <option> seleccionados
selectObject.deselectAll();
```

campus
Periferia IT

G R A C I A S

Periferia IT Group

Periferia IT Corp

CONTACT& BUSINESSIT

COOLEXPRESS