hiberus University

QA & Testing

Junior Academy

160 horas

hiberus University

Módulo 2

Introducción a la Automatización de Pruebas

80 horas aproximadamente

2.2 Introducción automatizar pruebas para aplicaciones web

Jonatan Villén / Rubén García

Introducción a Selenium

- Introducción y Arquitectura Selenium WebDriver
- Introducción a XPath y CSS Selector
- Comandos WebDriver
- Switches Alerts and Windows
- Action Class
- JUnit en Selenium.

Buenas Prácticas

- Patrón Page Object Model
- Reusabilidad
- Sincronización
- Reporting

Introducción a Selenium

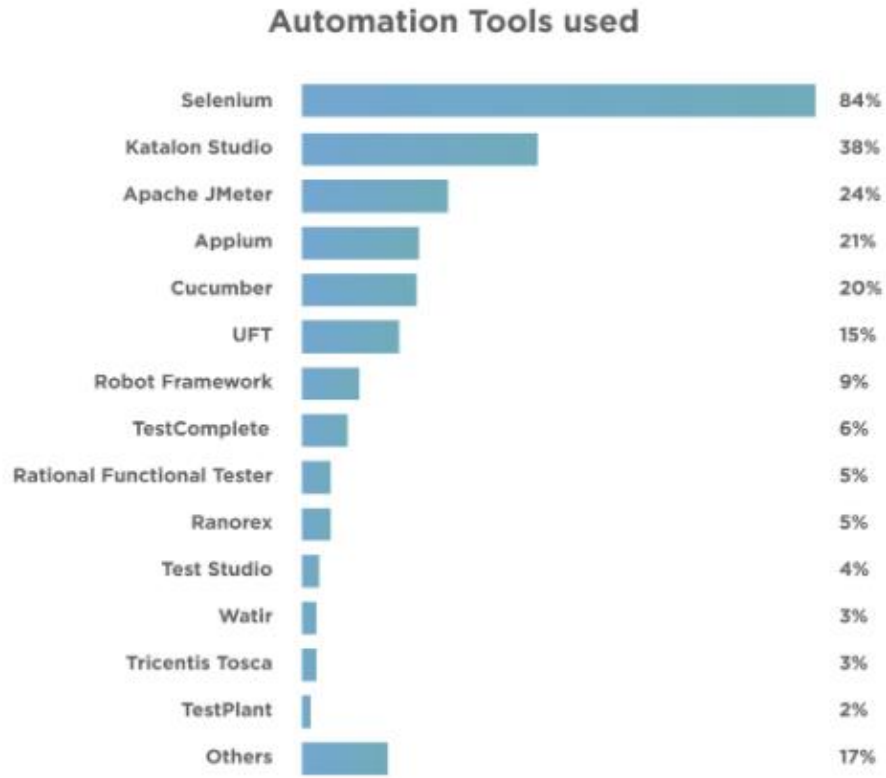
¿Qué es Selenium?

- Web de Selenium
- Grupo de herramientas de automatización de pruebas
- Código abierto
- Aprovecha el potencial de los navegadores web
- Automatiza flujo de trabajo de usuarios que interactúan con la aplicación web
- Lidera la lista de herramientas de automatización web y pruebas de automatización

Introducción a Selenium

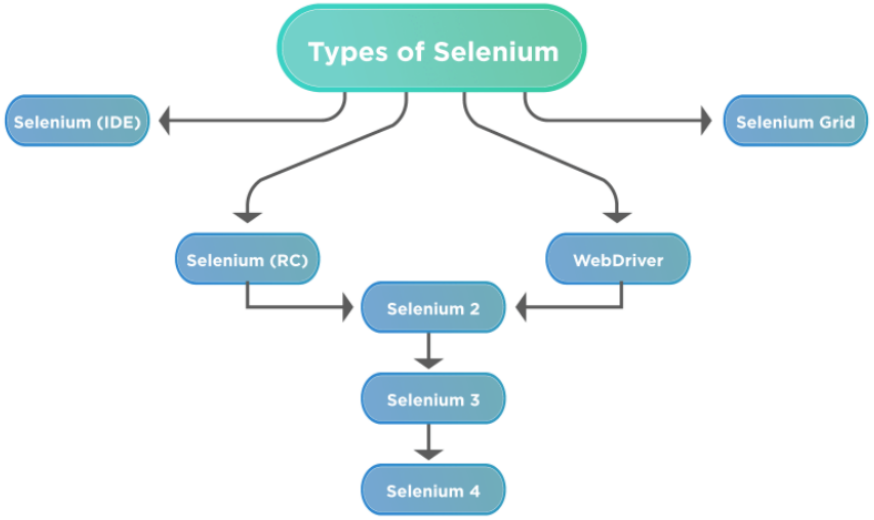
Herramienta líder

- Over 100 automation tools reported
- 84% have used Selenium
- Other popular tools
  - SoapUI
  - TestNG
  - Protractor
  - Postman

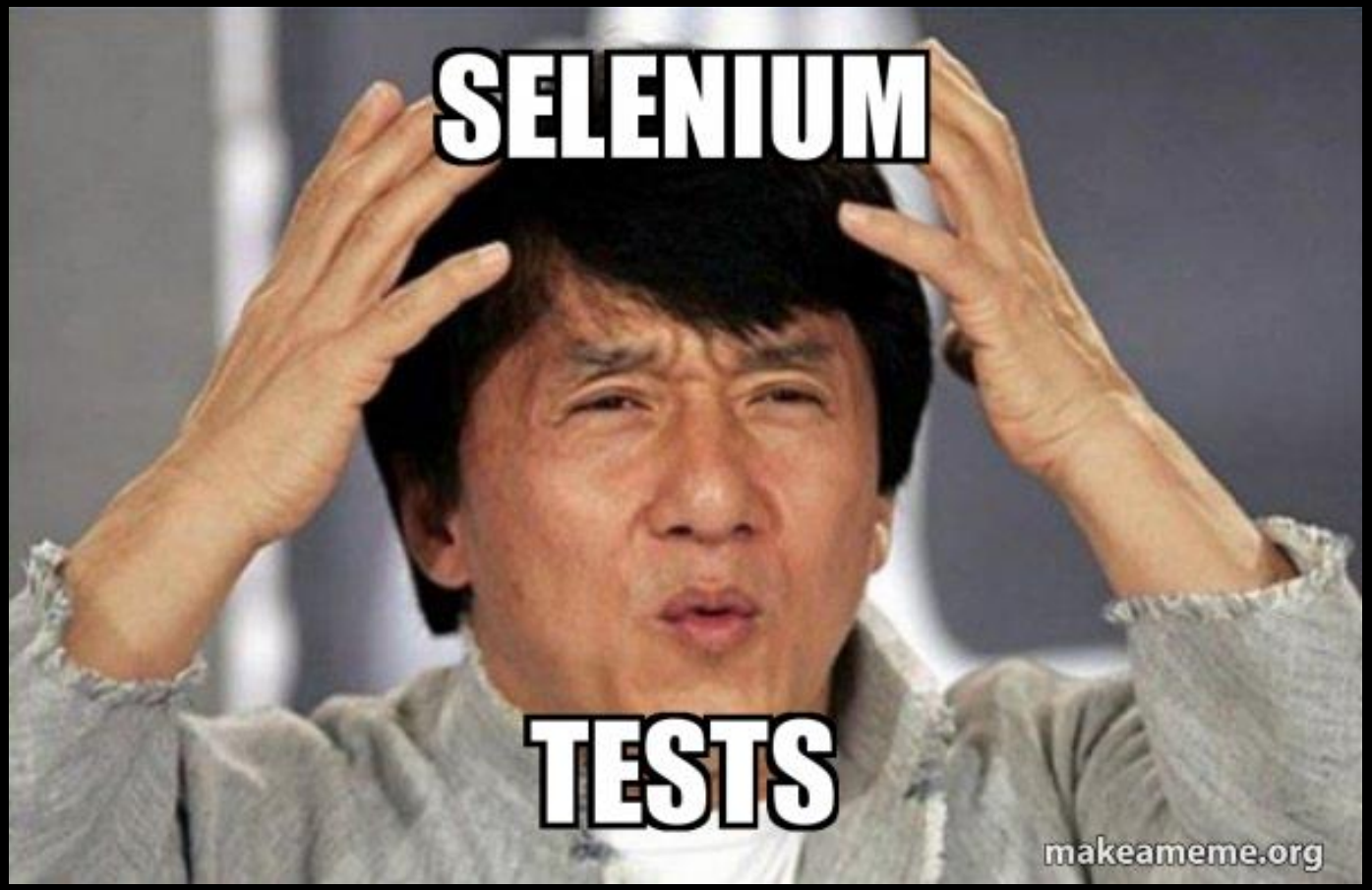


Introducción a Selenium

Componentes Suite



- No es solo herramienta de automatización
- **Conjunto de herramientas**
- Cada herramienta tiene capacidades específicas únicas
- Ayudan en el **diseño y desarrollo del marco** de automatización
- Utilizables **individualmente**
- **Combinables** entre sí
- Nos enfocamos en **Selenium 3**


SELENIUM
TESTS
makeameme.org

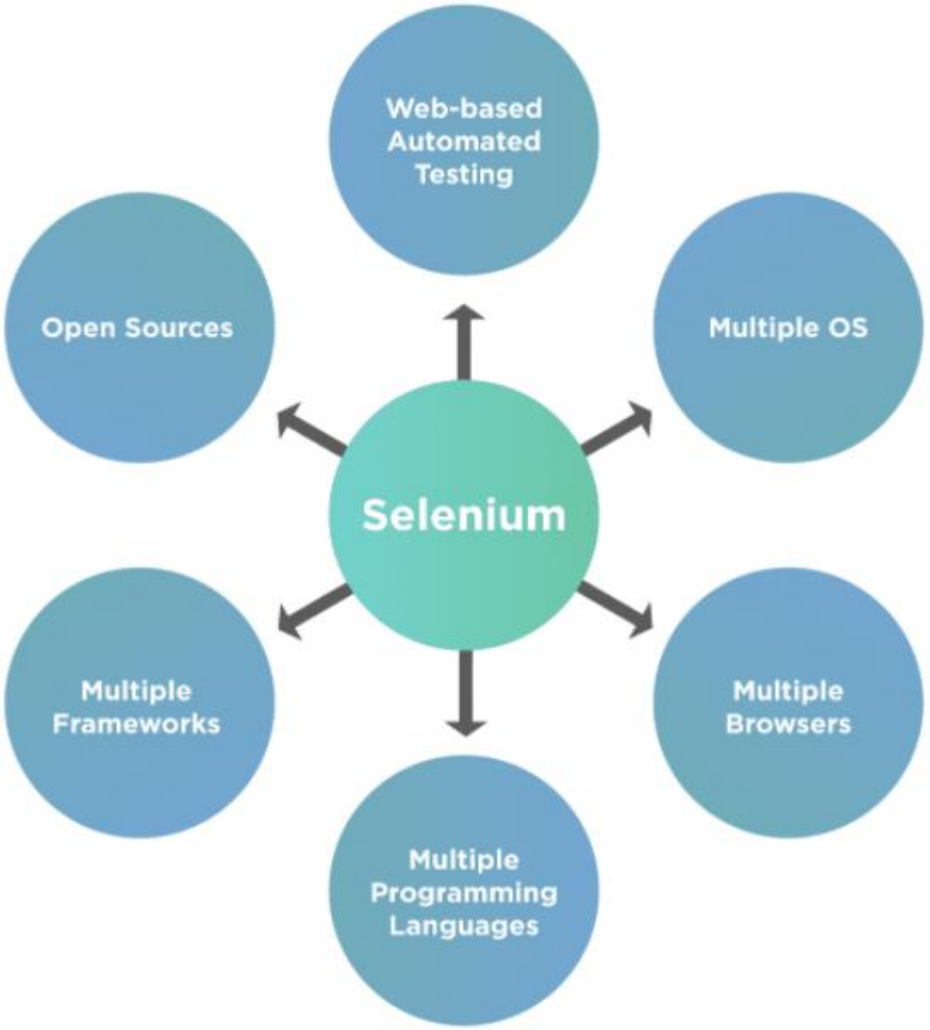
Introducción a Selenium

Selenium 3 Webdriver

- Componente más utilizado
- Permite escribir **código personalizado**
- **Lenguaje de programación de tu elección**
- Interactúa con el **navegador de tu elección**
- Utiliza controladores específicos del navegador
- Uiliza un protocolo llamado **JSONWireProtocol** para comunicar con navegador
- Selenium 2 implementación real fusión WebDriver y Selenium RC
- WebDriver y Selenium RC incorporadas en la versión 2 de WebDriver = Selenium 2
- Selenium 3 actualización sobre Selenium 2
- Selenium 3 estándar World Wide Web Consortium (W3C)
- Selenium RC pasó a un paquete heredado con mejoras y nuevas características
- Selenium 4 - nuevo y última versión de Selenium en fase beta

Introducción a Selenium

Características



Introducción a Selenium

Características

- **Código abierto**: descargar y usar de forma gratuita
- **Imitar las acciones del usuario**: clic, arrastrar y soltar la selección, casillas de verificación, pulsaciones de teclas, toques, desplazamiento
- **Fácil implementación**: se pueden desarrollar extensiones personalizadas
- **Soporte de idiomas**: Java, Python, JavaScript, C#, Ruby, Perl, Haskell, Go, entre otros.
- **Compatibilidad con navegadores**: funciona en todos los navegadores que existen y tiene soporte para Chrome, Firefox, Edge, Internet Explorer, Safari.
- **Compatibilidad con SO**: principales sistemas operativos como Linux, macOS y Windows.
- **Compatibilidad con Framework**: Maven, TestNG, PYTest, JUnit, Mocha, Jasmine, etc e integrable con herramientas de CI como Jenkins, Circle CI, Travis CI, Gitlab, etc.
- **Reutilización de código**: scripts compatibles con todos los navegadores. Mismo código ejecutable para varios navegadores usando los respectivos binarios de navegador y en máquinas separadas con una configuración de Grid.
- **Soporte de la comunidad**: muchos controles de calidad trabajando, fácil encontrar recursos, tutoriales y soporte en GitHub, StackOverflow, etc.

Introducción a Selenium

Compatibilidad


¿QUÉ SOMOS?
¡NAVEGADORES!
¡NAVEGADORES!
¡NAVEGADORES!
¿QUÉ QUEREMOS?
¡MAYOR VELOCIDAD!
¡MAYOR VELOCIDAD!
¡MAYOR VELOCIDAD!
¿Y CUÁNDO LO QUEREMOS?
¡AHORA MISMO!
¡AHORA MISMO!
¡AHORA MISMO!
¡NAVEGADORES!

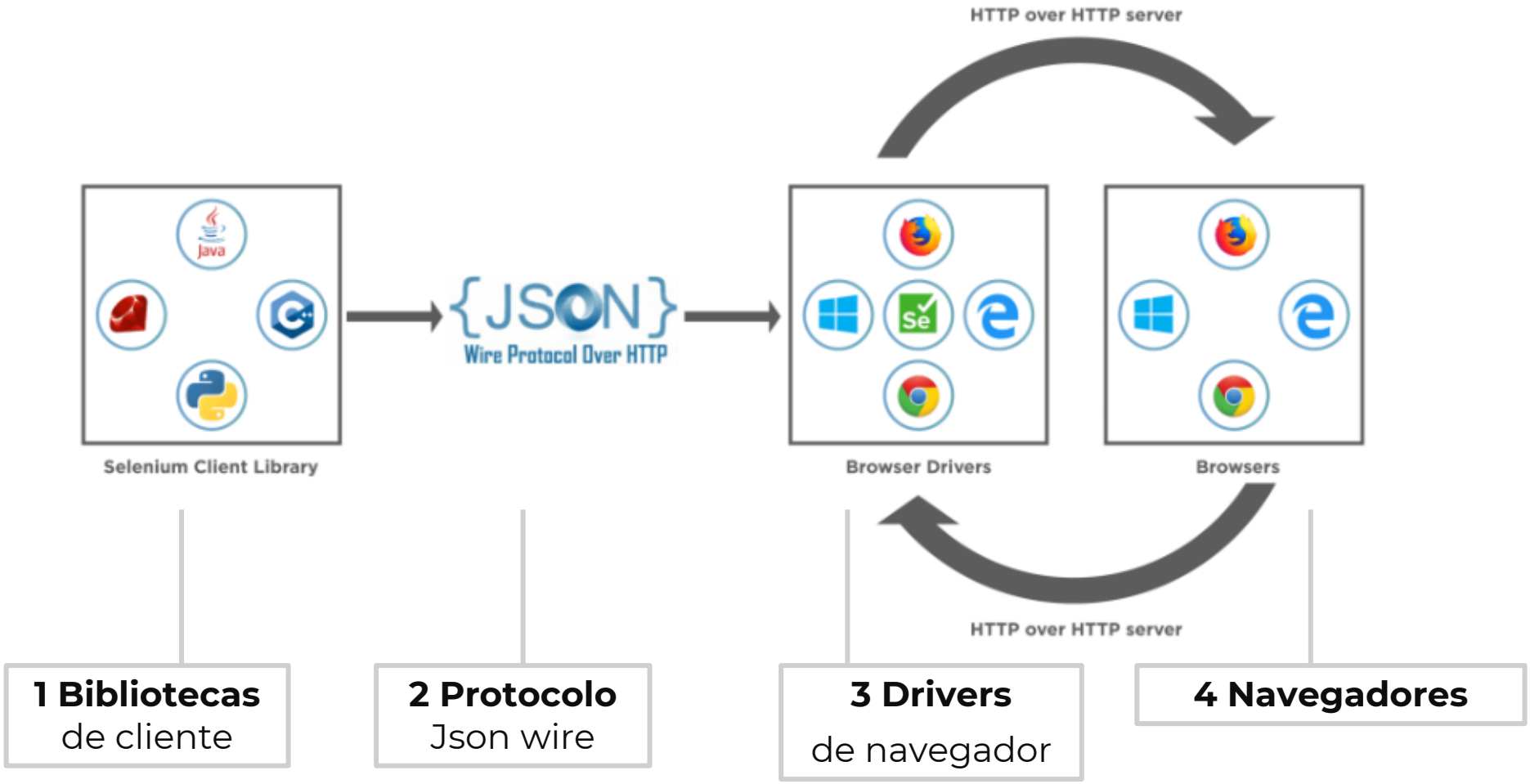
Introducción a Selenium

Limitaciones

- **Sin soporte para automatizar aplicaciones nativas** basadas en escritorio
- **Sin soporte para aserciones y validaciones,** necesita emparejarse con marco de prueba como JUnit, TestNG, PyTest, etc. para las afirmaciones
- **No es compatible con el escaneo de imágenes y códigos** como lectura de códigos de barras, CAPTCHAs
- **Sin soporte para pruebas de API**
- **Sin soporte para pruebas de rendimiento**
- **Sin informes incorporados,** necesario combinarlo con un marco como JUnit, TestNG

Introducción a Selenium

Arquitectura



Introducción a Selenium

Arquitectura y comandos Selenium WebDriver

- **Bibliotecas de cliente de Selenium WebDriver / Enlaces de idioma**: los evaluadores de software seleccionan idiomas con los que están cómodos. Hay enlaces disponibles para Java, C#, Python, Ruby, PHP, etc. Cualquiera con conocimiento básico sobre lenguajes puede obtener enlaces específicos. Selenium Achitecture brinda flexibilidad a los evaluadores para realizar la automatización en su zona de confort.

- **Protocolo Json Wire**: facilita toda la comunicación que se produce en Selenium entre navegador y código. Este es el corazón de Selenium. JSON Wire Protocol proporciona un medio para la transferencia de datos mediante una API RESTful (Representational State Transfer) que proporciona un mecanismo de transporte y define un servicio web RESTful mediante JSON sobre HTTP.

- **Drivers de navegador**: cada navegador tiene su propia implementación del estándar W3C. Existen binarios específicos para el navegador y ocultan la lógica de implementación del usuario final. El protocolo JSONWire establece una conexión entre los binarios del navegador y las librerías del cliente.

- **Navegadores**: solo podrá ejecutar pruebas en los navegadores si están instalados localmente en la máquina local o en las del servidor. La instalación del navegador es necesaria.

Introducción a Selenium

Cómo usarlo en 5 pasos

hiberus University

1. Crea una **instancia** de WebDriver específica para el navegador
2. Navega a la **página web** deseada que necesita ser automatizada
3. Localiza un **elemento HTML** en la página web
4. Realiza una **acción** en un elemento HTML
5. Ejecuta las **pruebas**
6. Registra los **resultados** de las pruebas utilizando un framework de pruebas

Introducción a Selenium

¿Qué son los localizadores o locators?

- Son la forma de identificar un elemento HTML en una página web
- Conocidos como "Selenium Locators“
- Concepto de "Selenium Locators":
    - ¿Qué son los localizadores en Selenium?
    - ¿Cómo localizar un elemento web en el DOM?
    - ¿Qué localizadores son compatibles con Selenium?

Introducción a Selenium

¿Qué son los localizadores?

Algunas de las diferentes etiquetas de los WebElements:

- <**title**>: Representa el titulo de un documento HTML.
- <**body**>: Define una parte de la sección principal (cuerpo) en un documento HTML.
- <**table**>: Se utiliza para definir una tabla en un documento HTML.
- <**th**>: Se utiliza para crear la cabecera de un grupo de celdas en una tabla HTML.
- <**tr**>: Define una fila de celdas en una tabla.
- <**td**>: Se utiliza para crear una celda de datos en una tabla HTML.
- <**span**>: Se utiliza para agrupar y aplicar estilos a los elementos en línea.

Introducción a Selenium

¿Qué son los localizadores?

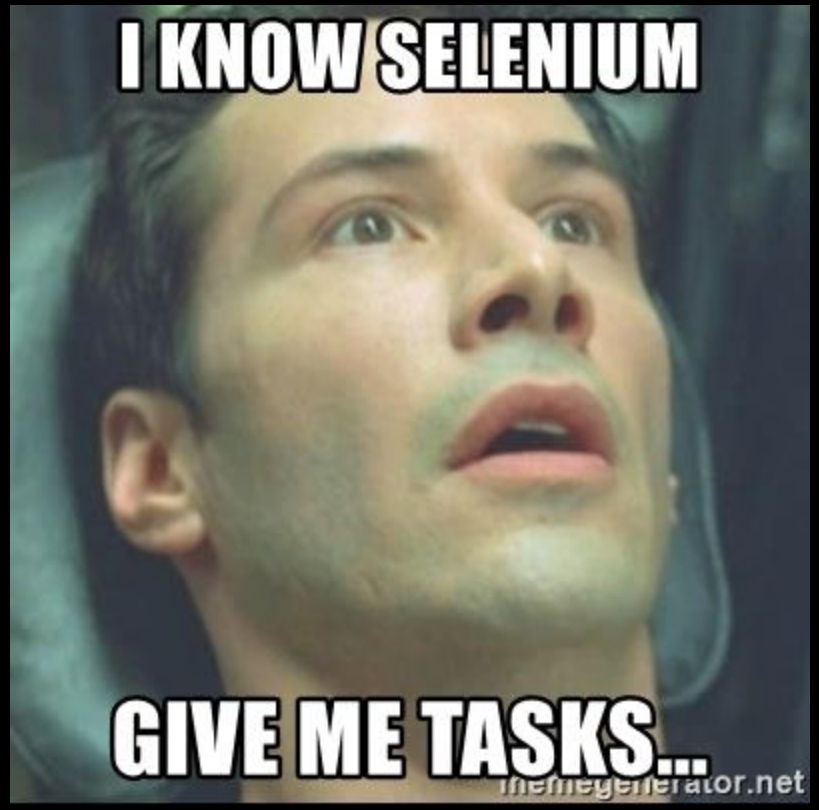
Algunas de las diferentes etiquetas de los WebElements:

- <**div**>: Definir una parte de la separación.
- <**input**>: Define la información que se obtiene en la entrada seleccionada.
- <**button**>: Especifica un botón que tenga la acción de press/push.
- <**a**>: Especifica un enlace (Hipervínculo).
- <**li**>: Define un elemento de lista ya sea lista ordenada o lista desordenada.
- <**select**>: Se utiliza para crear una lista desplegable.
- <**ul**>: Define una lista desordenada de elementos.

Introducción a Selenium

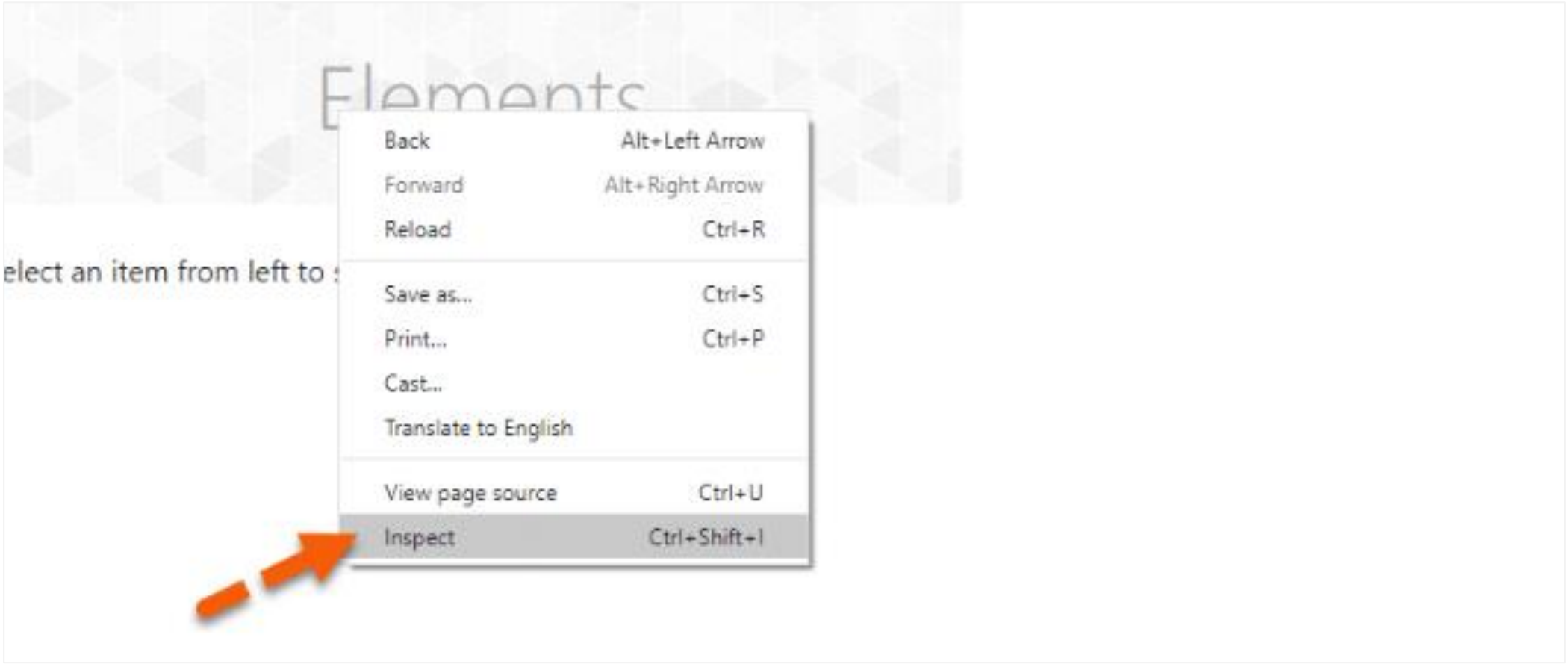
Cómo localizar un elemento web en DOM

- Acceder al DOM haciendo click derecho en la página Web y luego seleccionando **Inspeccionar**
- Se abrirá la **consola de Herramientas** para desarrolladores
- De forma predeterminada, abrirá la pestaña " **Elements** “con la estructura DOM completa de la página
- Pasar el puntero sobre las **etiquetas HTML en el DOM**
- Resaltará los **elementos correspondientes** que representa en la página web
- Para encontrar un elemento web en el DOM hacer click en la **flecha “Icono del mouse”**
- Seleccionar el **elemento web**
- Se resaltará el **elemento HTML** correspondiente


I KNOW SELENIUM
GIVE ME TASKS...

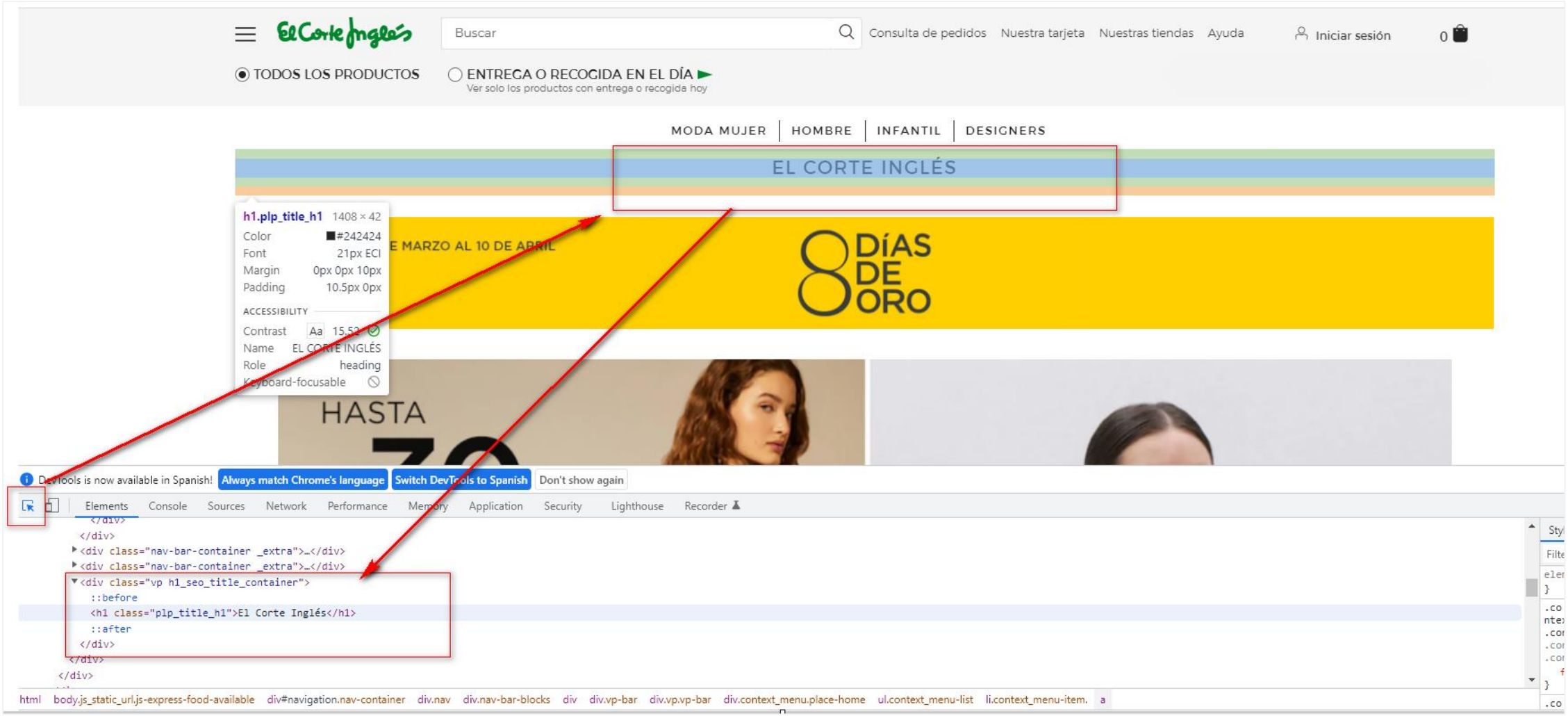
Introducción a Selenium

Cómo localizar un elemento web en DOM



Introducción a Selenium

Cómo localizar un elemento web en DOM



Introducción a Selenium

¿Qué localizadores son compatibles en Selenium?

- **className**: un operador ClassName utiliza un atributo de clase para identificar un objeto.
- **cssSelector**: para crear reglas de estilo para web e identificar cualquier elemento web.
- **id**: similar a la clase, también podemos identificar elementos usando el atributo 'id'.
- **linkText**: el texto utilizado en los hipervínculos también puede ubicar el elemento
- **name**: el atributo de nombre también puede identificar un elemento
- **parcialLinkText:** Parte del texto en el enlace también puede identificar un elemento
- **tagName**: también podemos usar una etiqueta para ubicar elementos
- **xpath**: lenguaje para consultar el XML e identificar de manera única el elemento web en cualquier página.

Introducción a Selenium

XPath

Ruta XML para navegar a través de la estructura HTML de la página

Sintaxis o lenguaje para encontrar cualquier elemento en una página web

Utiliza una **expresión de ruta XML**

Para **documentos HTML y XML** usando la estructura HTML DOM

Contiene la ruta del elemento situado en la página web

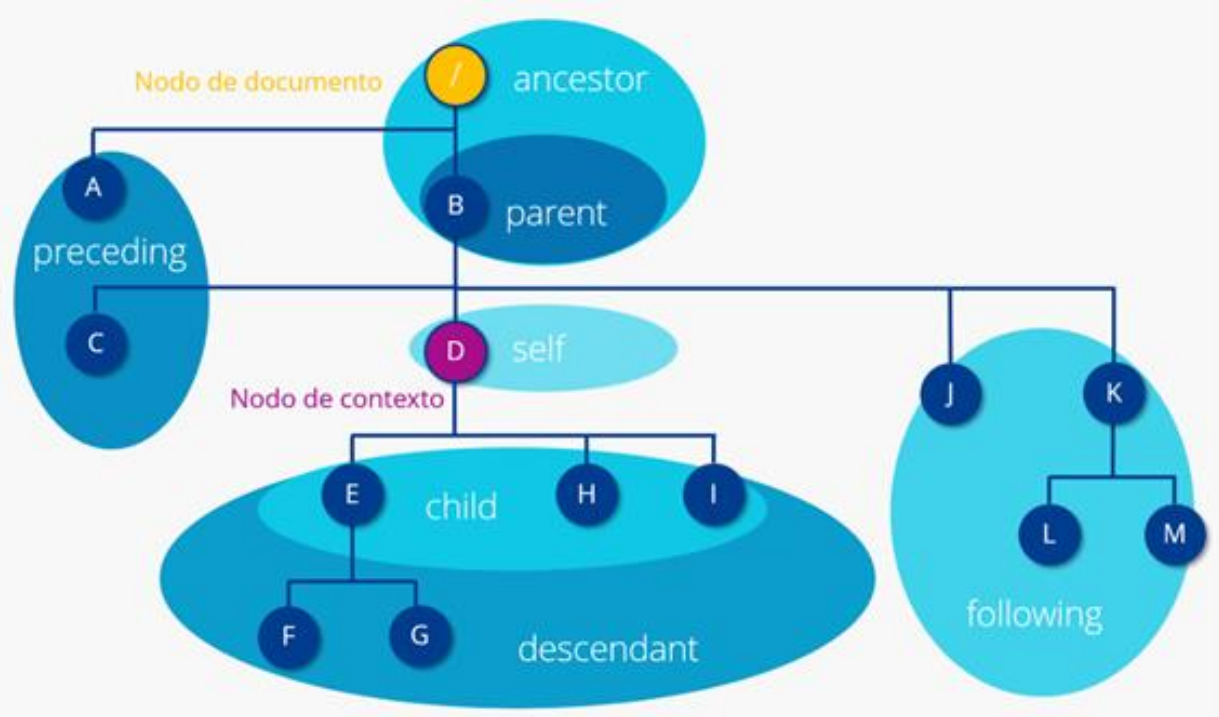
La sintaxis XPath estándar es:

- **//** : Selecciona el nodo actual
- **Tagname**: Tagname del nodo en particular
- **@:** Seleccionar atributo
- : Nombre del atributo del nodo
- **Valor**: Valor del atributo

Introducción a Selenium

Escribir Xpath en Selenium

- XPath básico
- Contains ()
- OR & AND
- Función text() de XPath
- Métodos de Ejes en XPath:



Introducción a Selenium

Ejercicios Xpath en Selenium

- Ejercicio 0. XPath básico:

  Si la entrada HTML DOM es:

```
<html>
    <body>
        <a>link</a>
        <div class='container' id='divone'>
            <p class='common' id='enclosedone'>Element One</p>
            <p class='common' id='enclosedtwo'>Element Two</p>
        </div>
    </body>
</html>
```

Introducción a Selenium

CSS Selector en Selenium

- Las páginas Web actuales están construidas con CSS
- CSS definen **comportamiento estético** como tamaño y fuentes de toda la página
- CSS Selectors > Combinación de **elemento y valor de selector** que identifica elemento web
- Esta combinación se conoce como **patrón de selector**
- El uso de **selectores CSS** para ubicar elementos, tiene algunos beneficios:
  - Es más rápido a la hora de navegar automáticamente
  - Las rutas a los elementos son más legibles

Introducción a Selenium

Comandos básicos CSS Selector

- Atributo ID
- Atributo Class
- Localización de elementos con más de un atributo
- Localizador elementos Child en CSS
- Otros selectores CSS

Introducción a Selenium

Comandos del navegador Selenium Webdriver

- Comando **Get** - ¿Cómo abrir una página web en Selenium?
- Comando **getTitle** - ¿Cómo obtener el título de la página web en Selenium?
- Comando **getCurrentUrl** - ¿Cómo leer la URL de la página web en Selenium?
- Comando **getPageSource** - ¿Cómo leer el código fuente de la página web en Selenium?
- Comando **close** - ¿Cómo cerrar el navegador en Selenium?
- Comando **quit** - ¿Cómo cerrar todas las ventanas del navegador en Selenium?

Introducción a Selenium

Comandos del navegador Selenium Webdriver

Comando Get:

Sirve para cargar la pagina web por la que se desea navegar

```
driver.get("https://www.google.com");

//Or can be written as

String URL = "https://www.google.com";
driver.get(URL);
```

Introducción a Selenium

Comandos del navegador Selenium Webdriver

Comando GetTitle:

Sirve para obtener el titulo de la pagina web.

```
driver.getTitle();

//O puede utilizarse como

String Title = driver.getTitle();
```

Introducción a Selenium

Comandos del navegador Selenium Webdriver

Comando GetCurrentUrl:

Sirve para obtener la URL actual de la pagina web.

```
driver.getCurrentUrl();

//O puede escribirse como

String CurrentUrl = driver.getCurrentUrl();
```

Introducción a Selenium

Comandos del navegador Selenium Webdriver

Comando GetPageSource:

Sirve para obtener el código fuente de la pagina web.

```
driver.getPageSource();

//O puede escribirse como
String PageSource = driver.getPageSource();
```

Introducción a Selenium

Comandos del navegador Selenium Webdriver

Comando Close:

Sirve para cerrar la ventana actual que Selenium Webdriver esta controlando en ese momento.

```
driver.close();
```

Introducción a Selenium

Comandos del navegador Selenium Webdriver

Comando Quit:

Sirve para cerrar todas las ventanas abiertas por Selenium Webdriver.

```
driver.quit();
```

Introducción a Selenium

Comandos de navegación de Selenium Webdriver

- Comando **Navigate** - ¿Cómo navegar a la URL o como abrir una página web en Selenium?
- Comando **Forward** - ¿Cómo navegar hacía adelante en el navegador con Selenium?
- Comando **Back** - ¿Cómo navegar hacía atrás en el navegador con Selenium?
- Comando **Refresh** - ¿Cómo actualizar el navegador con Selenium?

Introducción a Selenium

Comandos de navegación Selenium Webdriver

Comando Navigate:

Sirve para cargar una nueva pagina web en la ventana actual del navegador.

```
driver.navigate().to("https://www.hiberus.com");
```

Introducción a Selenium

Comandos de navegación Selenium Webdriver

Comando Forward:

Sirve para simular el hacer click en el botón de navegar hacia Adelante..

```
driver.navigate().forward();
```

Introducción a Selenium

Comandos de navegación Selenium Webdriver

Comando Back:

Sirve para simular el hacer click en el botón de navegar hacia Atrás.

```
driver.navigate().back();
```

Introducción a Selenium

Comandos de navegación Selenium Webdriver

Comando Refresh:

Sirve para actualizar la pagina actual.

```
driver.navigate().refresh();
```

Introducción a Selenium

¿Qué es un WebElement?

- Es un objeto que representa un **elemento HTML**
- Los elementos HTML se escriben con **etiquetas**
- Sintaxis: etiqueta de **inicio y finalización, y contenido** en el medio:

  <tagname> contenido </tagname>.
- El elemento HTML es todo, desde la etiqueta inicial hasta la etiqueta final:

   Hola
- Los elementos HTML se pueden **anidar** - los elementos pueden contener elementos
- Todos los documentos HTML constan de **elementos HTML anidados**

```
<html>
        <body>
                <h1> My First Heading </h1>
                <p> My first paragraph. </p>
        </body>
</html>
```

Introducción a Selenium

Acciones de WebElement

Sintaxis de un objeto **WebElement:**

```
WebElement element = driver.findElement(By.xpath(‘//input[@id=“username”]’));
```

Comandos de un objeto **WebElement:**

- Comando **Clear**
- Comando **SendKeys**
- Comando **Click**
- Comando **IsDisplayed**
- Comando **IsEnabled**
- Comando **IsSelected**
- Comando **Submit**
- Comando **GetText**
- Comando **GetAttribute**

Introducción a Selenium

Acciones WebElement

Comando Clear:

Sirve para borrar el valor de una entrada de texto.

```
WebElement element = driver.findElement(By.id("UserName"));
element.clear();

//O puede escribirse como

driver.findElement(By.id("UserName")).clear();
```

Introducción a Selenium

Acciones WebElement

Comando SendKeys:

Sirve para escribir un valor en una entrada de texto.

```
WebElement element = driver.findElement(By.id("UserName"));
element.sendKeys("Manolo");

//O puede escribirse como

driver.findElement(By.id("UserName")).sendKeys("Manolo");
```

Introducción a Selenium

Acciones WebElement

Comando Click:

Sirve para hacer click en cualquier elemento.

```
WebElement element = driver.findElement(By.linkText("Clientes"));
element.click();

//Or can be written as

driver.findElement(By.linkText("Clientes")).click();
```

Introducción a Selenium

Acciones WebElement

Comando IsDisplayed:

Sirve para determinar si un elemento se esta mostrando actualmente o no.

```
WebElement element = driver.findElement(By.id("UserName"));
boolean status = element.isDisplayed();

//Or can be written as

boolean staus = driver.findElement(By.id("UserName")).isDisplayed();
```

Introducción a Selenium

Acciones WebElement

Comando IsEnabled:

Sirve para determinar si un elemento esta habilitado o no.

```
WebElement element = driver.findElement(By.id("UserName"));
boolean status = element.isEnabled();

//O puede escribirse como

boolean staus = driver.findElement(By.id("UserName")).isEnabled();

//O se puede utilizar como
WebElement element = driver.findElement(By.id("userName"));
boolean status = element.isEnabled();
// Comprueba si el campo de texto está habilitado, si es así introduzca el valor
if(status){
    element.sendKeys("Manolo");
}
```

Introducción a Selenium

Acciones WebElement

Comando IsSelected:

Sirve para determinar si un elemento esta seleccionado o no.

```
WebElement element = driver.findElement(By.id("Sex-Male"));
boolean status = element.isSelected();

//Or can be written as

boolean staus = driver.findElement(By.id("Sex-Male")).isSelected();
```

Introducción a Selenium

Acciones WebElement

Comando GetText:

Sirve para devolver un texto interno del elemento

```
WebElement element = driver.findElement(By.xpath("anyLink"));
String linkText = element.getText();
```

Introducción a Selenium

Acciones WebElement

Comando GetAttribute:

Sirve para obtener el valor del atributo del elemento

```
WebElement element = driver.findElement(By.id("SubmitButton"));
String attValue = element.getAttribute("id"); //This will return "SubmitButton"
```

Introducción a Selenium

Encontrar WebElements en Selenium WebDriver

Una web consta de numerosos WebElements:

- Cuadros de texto
- Botones
- Listas...

Para poder usar los comandos de WebElement, antes debemos encontrar los elementos aplicando:

- Métodos para encontrar elementos.
- Utilización de estrategias de localización de los elementos.

Introducción a Selenium

Encontrar WebElements en Selenium WebDriver

Métodos para encontrar elementos:

- FindElement

```
WebElement elementName = driver.findElement(By.LocatorStrategy("LocatorValue"));
```

- FindElements

```
List<WebElement> elementNameList = driver.findElements(By.LocatorsStrategy("LocatorValue"));
```

Métodos para usar estrategias de localización de elementos:

- By ID
- By NAME
- By CLASS NAME
- By CSS SELECTOR
- By XPATH
- By LINKTEXT

Introducción a Selenium

Encontrar WebElements en Selenium Webdriver

Estrategia de localización ID:

Sirve para encontrar elementos por el atributo ID. El valor de búsqueda dado debe de coincidir con el atributo ID.

WebElement HTML:

```
<button id="submit">ACEPTAR</button>
```

Sintaxis Selenium WebDriver:

```
WebElement button = driver.findElement(By.id("submit"));
```

Introducción a Selenium

Encontrar WebElements en Selenium Webdriver

Estrategia de localización NAME:

Sirve para encontrar o ubicar elementos basados en el atributo NAME.

WebElement HTML:

```
<input id="user-name" name="username" >
```

Sintaxis Selenium WebDriver:

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

Introducción a Selenium

Encontrar WebElements en Selenium Webdriver

Estrategia de localización CLASSNAME:

Sirve para encontrar elementos que coincidan con el valor del *class* especificado. Hay que tener en cuenta que las clases con valores compuestos no están permitidas como nombres de estrategia.

WebElement HTML:

<input *id*="user-name" *class*="text" >

~~<input *id*="user-name" *class*="text custom" >~~

Sintaxis Selenium WebDriver:

WebElement input = driver.findElement(By.className("text"));

~~WebElement input = driver.findElement(By.className("text custom"));~~

Introducción a Selenium

Encontrar WebElements en Selenium Webdriver

Estrategia de localización CSS SELECTOR:

Sirve para encontrar elementos con el selector de CSS.

WebElement HTML:

```
<input id="user-name" name="username" >
```

Sintaxis Selenium WebDriver:

```
WebElement input = driver.findElement(By.cssSelector("input#user-name"));
```

Introducción a Selenium

Encontrar WebElements en Selenium Webdriver

Estrategia de localización XPATH:

Sirve para hacer coincidir la expresión XPath con el valor de búsqueda y de este modo ubicar el elemento.

WebElement HTML:

<input *id*="user-name" *name*="username" >

Sintaxis Selenium WebDriver:

```
WebElement input = driver.findElement(By.xpath("//input[@id='user-name']"));
```

Introducción a Selenium

Encontrar WebElements en Selenium Webdriver

Estrategia de localización LINKTEXT:

Sirve para hacer encontrar elementos de enlaces mediante su valor de texto.

WebElement HTML:

```
<a id="simpleLink" href="http://www.ejemplo.com" >Link Ejemplo</a>
```

Sintaxis Selenium WebDriver:

```
WebElement input = driver.findElement(By.linkText("Link Ejemplo"));
```

Introducción a Selenium

Encontrar WebElements en Selenium Webdriver

Estrategia de localización LINKTEXT:

Sirve para hacer encontrar elementos de enlaces mediante su valor de texto.

WebElement HTML:

```
<a id="simpleLink" href="http://www.ejemplo.com" >Link Ejemplo</a>
```

Sintaxis Selenium WebDriver:

```
WebElement input = driver.findElement(By.linkText("Link Ejemplo"));
```

Introducción a Selenium

Anotaciones JUnit

@BeforeClass:

Sirve para inicializar cualquier objeto que se use en la ejecución de las pruebas.

```
1  @BeforeClass
2      public static void SetUpClass()
3      {
4              //Initialization code goes here
5              System.out.println("This is @BeforeClass annotation");
6      }
```

Introducción a Selenium

Anotaciones JUnit

@Before:

Sirve para inicializar cualquier objeto durante el tiempo que se esta usando el método de la prueba.

```
1  @Before
2      public void SetUp()
3      {
4              // Setting up the test environment
5              System.out.println("This is @Before annotation");
6      }
```

Introducción a Selenium

Anotaciones JUnit

@Test:

Sirve para decirle a JUnit que el método public void() que tiene, se puede ejecutar como un caso de prueba.

```
1       @Test
2       public void Addition()
3       {
4               c= a+b;
5               assertEquals(15,c);
6               System.out.println("This is first @Test annotation method= " +c);
7       }
8
9       @Test
10      public void Multiplication()
11      {
12              c=a*b;
13              assertEquals(50,c);
14              System.out.println("This is second @Test annotation method= " +c);
15      }
```

Introducción a Selenium

Anotaciones JUnit

@After:

Sirve para liberar la inicialización que se realiza en la anotación Before. Se ejecutara después de cada método de prueba (@Test).

```
1       @After
2       public void TearDown()
3       {
4               // Cleaning up the test environment
5               c= null;
6               System.out.println("This is @After annotation");
7       }
```

Introducción a Selenium

Anotaciones JUnit

@AfterClass:

Sirve para liberar la inicialización que se realiza en la anotación BeforeClass. Se ejecutara después de que todos los métodos de prueba, hayan terminado de ejecutarse.

```
1     @AfterClass
2     public static void TearDownClass()
3     {
4             //Release your resources here
5             System.out.println("This is @AfterClass annotation");
6     }
```

Introducción a Selenium

Anotaciones JUnit

@Ignore:

Sirve para decirle a JUnit que este método de prueba, no se ejecutara.

```
1     @Ignore
2     public void IgnoreMessage()
3     {
4        String info = "JUnit Annotation Blog" ;
5        assertEquals(info,"JUnit Annotation Blog");
6        System.out.println("This is @Ignore annotation");
7     }
```

Introducción a Selenium

Assertions en JUnit

assertArrayEquals:

Sirve para comparar 2 arrays y afirmar distintas propiedades de este.

```
1  @Test
2  public void testAssertArrayEquals() {
3   String[] nombresEsperados = { "java", "junit", "jboss" };
4   String[] nombresActuales = { "java", "junit", "jboss" };
5
6   assertArrayEquals("Fallo - No son los mismos arrays",
7     nombresEsperados, nombresActuales);
8  }
```

Introducción a Selenium

Assertions en JUnit

assertEquals:

Sirve para comparar 2 tipos de datos u objetos y afirmar que son iguales.

```
1  @Test
2  public void testSumar() {
3   assertEquals("El resultado esta mal: ", (1 + 1), 2);
4  }
```

Introducción a Selenium

Assertions en JUnit

assertFalse:

Sirve para afirmar que un tipo de dato u objeto es falso.

```
1  public boolean validarMax(int maximo){
2   boolean max = false;
3
4   for (int i = 0; i < maximo; i++) {
5    if(i == 3){
6     max = true;
7     break;
8    }
9   }
10  return max;
11 }
```

```
1  @Test
2  public void testValidarMax() {
3   assertFalse("Esta variable no es false: ", validarMax(3));
4  }
```

Introducción a Selenium

Assertions en JUnit

assertNotNull:

Sirve para afirmar que un tipo de dato u objeto no es nulo.

```
@Test
 public void testAssertNotNull() {
  List<string> nombres = new ArrayList<string>();
  assertNotNull("La lista 'nombres' esta null:", nombres);

  for (int i = 0; i < 3; i++) {
   if (i == 2) {
    nombres = null;
   }
  }
  assertNotNull("La lista 'nombres' esta null:", nombres);
 }
</string></string>
```

Introducción a Selenium

Assertions en JUnit

assertNotNull:

Sirve para afirmar que un tipo de dato u objeto no es nulo.

```
1   @Test
2    public void testAssertNotNull() {
3     List<string> nombres = new ArrayList<string>();
4     assertNotNull("La lista 'nombres' esta null:", nombres);
5
6     for (int i = 0; i < 3; i++) {
7      if (i == 2) {
8       nombres = null;
9      }
10    }
11    assertNotNull("La lista 'nombres' esta null:", nombres);
12   }
13
```

Introducción a Selenium

Assertions en JUnit

assertNotSame:

Sirve para comparar dos tipos de datos y afirmar que son distintos.

```
1  public boolean validarMax(int maximo){
2   boolean max = false;
3
4   for (int i = 0; i < maximo; i++) {
5    if(i == 3){
6     max = true;
7     break;
8    }
9   }
10  return max;
11 }
```

```
1  @Test
2  public void testAssertNotSame() {
3   boolean esMaximo = validarMax(4);// El retorno es true
4   boolean noEsMaximo = validarMax(3);// EL retorno es falso
5
6   assertNotSame("Fallo - No son iguales los dos objetos: ", esMaximo,
7     noEsMaximo);
8  }
```

Introducción a Selenium

Assertions en JUnit

assertNull:

Sirve para afirmar que un tipo de dato u objeto es nulo.

```
1  @Test
2  public void testAssertNull() {
3   String nombre = null;
4
5   List<string> nombres = new ArrayList<string>();
6   nombres.add("Java");
7   nombres.add("JUnit");
8   nombres.add("JBoss");
9
10  for(String n: nombres){
11   if(n.equals("JUnit")){
12    nombre = "Tutorial JUnit Assertions"; //Asignamos un valor a la variable
13    break;
14   }
15  }
16  assertNull("No esta nulo el objeto", nombre);
17 }
18
```

Introducción a Selenium

Assertions en JUnit

assertTrue:

Sirve para afirmar que un tipo de dato u objeto es verdadero.

```
public boolean validarMax(int maximo){
 boolean max = false;

 for (int i = 0; i < maximo; i++) {
  if(i == 3){
   max = true;
   break;
  }
 }
 return max;
}
```

```
@Test
public void testAssertTrue() {
 assertTrue("Esta variable no es false: ", validarMax(4));
}
```

Introducción a Selenium

Suite en JUnit

Suite:

Consiste en una agrupación de varias clases con pruebas para que se ejecuten de forma conjunta.

Definición de una Suite:

```
1
2  import org.junit.runner.RunWith;
3  import org.junit.runners.Suite;
4
5  @RunWith(Suite.class)
6  @Suite.SuiteClasses({
7          UserServiceTest.class,
8          LifecycleTest.class
9  })
10 public class TestingSuite {
11 //clase vacia
12
13 }
14
```

hiberus University

Muchas gracias

¿Preguntas?

hiberus University

QA & Testing

Junior Academy

160 horas