EBook Gratis

APRENDIZAJE selenium-webdriver

Free unaffiliated eBook created from
Stack Overflow contributors.

#selenium-webdriver

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: selenium-webdriver

It is an unofficial and free selenium-webdriver ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official selenium-webdriver.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con Selenium-webdriver

Observaciones

Esta sección proporciona una descripción general de lo que es selenio-webdriver, y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema importante dentro de selenio-webdriver, y vincular a los temas relacionados. Dado que la Documentación para Selenium-webdriver es nueva, es posible que deba crear versiones iniciales de esos temas relacionados.

Versiones

Versión	Fecha de lanzamiento
0.0.1	2016-08-03

Examples

Instalación o configuración

Para comenzar a utilizar WebDriver, deberá obtener el controlador relevante del sitio de Selenium : Descargas de Selenium HQ . Desde aquí, debe descargar el controlador correspondiente a los navegadores y / o plataformas en los que está intentando ejecutar WebDriver, por ejemplo, si estaba probando en Chrome, el sitio de Selenium lo dirigirá a:

https://sites.google.com/a/chromium.org/chromedriver/

Para descargar `chromedriver.exe` .

Finalmente, antes de poder utilizar WebDriver, deberá descargar los enlaces de idioma relevantes. Por ejemplo, si usa C #, puede acceder a la descarga desde la página de descargas de Selenium HQ para obtener los archivos .dll necesarios o, alternativamente, descargarlos como paquetes en Visual Studio. a través del gestor de paquetes NuGet.

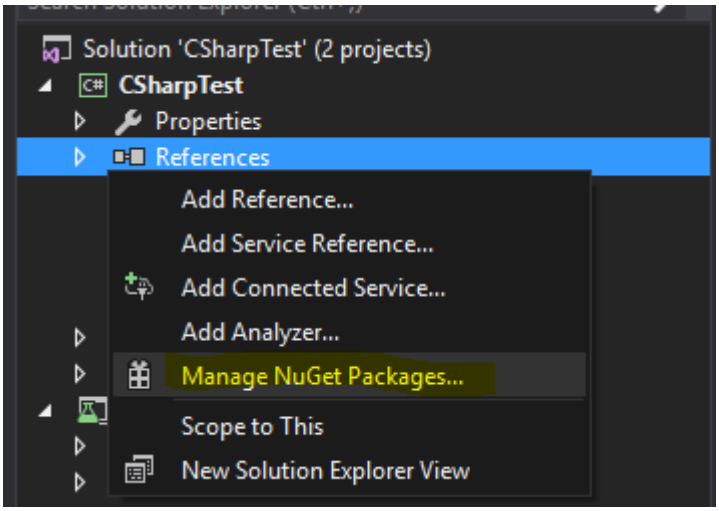
Los archivos requeridos ahora deben descargarse, para obtener información sobre cómo comenzar a usar WebDriver, consulte la documentación de `selenium-webdriver` .

Para Visual Studio [NuGet]

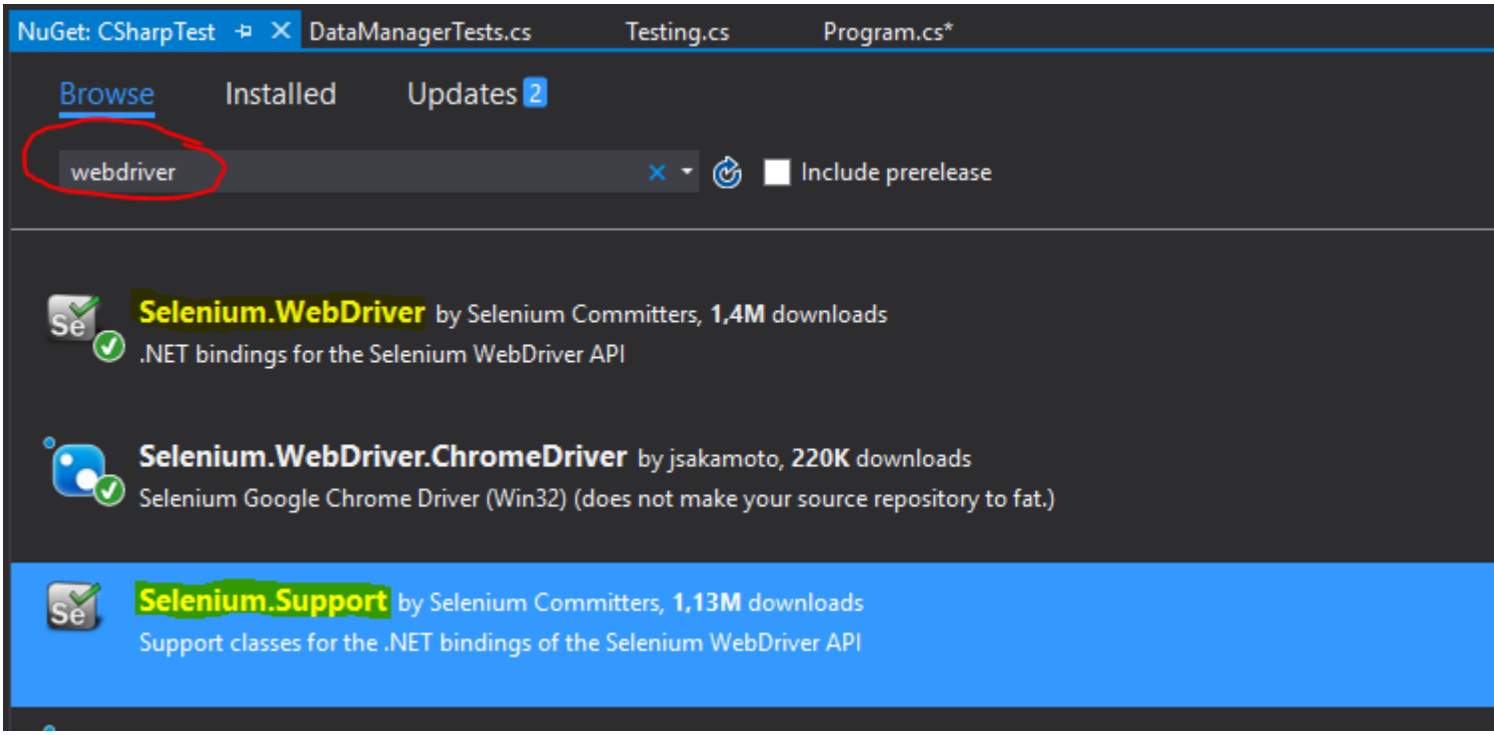
La forma más sencilla de instalar Selenium WebDriver es mediante el uso de un administrador de

paquetes NuGet.

En su proyecto, haga clic con el botón derecho en "Referencias", y haga clic en "Administrar paquetes de NuGet" como se muestra:



Luego, escriba en el cuadro de búsqueda " *webdriver* ". Entonces deberías ver algo como esto:



Instale " **Selenium.WebDriver** " y " **Selenium.Support** " (el paquete de Soporte incluye recursos adicionales, como Esperar) haciendo clic en el botón Instalar en el lado derecho.

Luego, puede instalar los WebDrivers que desee usar, como uno de estos:

- Selenium.WebDriver.ChromeDriver (Google Chrome)
- PhantomJS (sin cabeza)
-

¿Qué es Selenium WebDriver?

Selenium es un conjunto de herramientas diseñadas para automatizar los navegadores. Se usa comúnmente para pruebas de aplicaciones web en múltiples plataformas. Hay algunas herramientas disponibles bajo el paraguas de Selenium, como Selenium WebDriver (ex-Selenium RC), Selenium IDE y Selenium Grid.

WebDriver es una *interfaz de* control remoto que le permite manipular elementos DOM en páginas web, así como controlar el comportamiento de los agentes de usuario. Esta interfaz proporciona un protocolo de conexión independiente del idioma que se ha implementado para varias plataformas, tales como:

- GeckoDriver (Mozilla Firefox)
- ChromeDriver (Google Chrome)
- SafariDriver (Apple Safari)
- InternetExplorerDriver (MS InternetExplorer)
- MicrosoftWebDriver o EdgeDriver (MS Edge)
- OperaChromiumDriver (navegador Opera)

así como otras implementaciones:

- EventFiringWebDriver
- HtmlUnitDriver
- PhantomJSDriver
- RemoteWebDriver

Selenium WebDriver es una de las herramientas de Selenium que proporciona API orientadas a objetos en una variedad de idiomas para permitir un mayor control y la aplicación de prácticas de desarrollo de software estándar. Para simular con precisión la forma en que un usuario interactúa con una aplicación web, utiliza "Eventos nativos de nivel de SO" como opuestos a "Eventos de JavaScript sintetizados".

Enlaces a referir:

- http://www.seleniumhq.org/
- http://www.aosabook.org/en/selenium.html
- https://www.w3.org/TR/webdriver/

Instalación o configuración para Java

Para escribir pruebas utilizando Selenium Webdriver y Java como lenguaje de programación, deberá descargar los archivos JAR de Selenium Webdriver desde el sitio web de Selenium.

Hay varias formas de configurar un proyecto Java para el controlador web Selenium, una de las más fáciles de todas es usar Maven. Maven descarga los enlaces Java necesarios para el controlador de web Selenium, incluidas todas las dependencias. La otra forma es descargar los archivos JAR e importarlos a su proyecto.

Pasos para configurar el proyecto Selenium Webdriver usando Maven:

1. Instale Maven en el cuadro de Windows siguiendo este documento: https://maven.apache.org/install.html
2. Crear una carpeta con nombre `selenium-learing`
3. Cree un archivo en la carpeta anterior usando cualquier editor de texto con el nombre `pom.xml`
4. Copia el contenido de abajo a `pom.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
    <project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
            <modelVersion>4.0.0</modelVersion>
            <groupId>SeleniumLearning</groupId>
            <artifactId>SeleniumLearning</artifactId>
            <version>1.0</version>
            <dependencies>
                <dependency>
                    <groupId>org.seleniumhq.selenium</groupId>
                    <artifactId>selenium-learning</artifactId>
                    <version>3.0.0-beta1</version>
                </dependency>
            </dependencies>
    </project>
```

Nota : asegúrese de que la versión que especificó anteriormente sea la más reciente. Puede consultar la última versión desde aquí: http://docs.seleniumhq.org/download/maven.jsp

5. Usando la línea de comandos, ejecute el comando siguiente en el directorio del proyecto.

   ```
   mvn clean install
   ```

El comando anterior descargará todas las dependencias necesarias y luego se agregará al proyecto.

6. Escriba a continuación el comando para generar un proyecto de eclipse que puede importar al IDE de Eclipse.

   ```
   mvn eclipse:eclipse
   ```

7. Para importar el proyecto en el eclipse ide, puede seguir los siguientes pasos

   Abra Elipse -> Archivo -> Importar -> General -> Proyecto existente en el área de trabajo -> Siguiente -> Examinar -> Localice la carpeta que contiene pom.xml -> Ok -> Finalizar

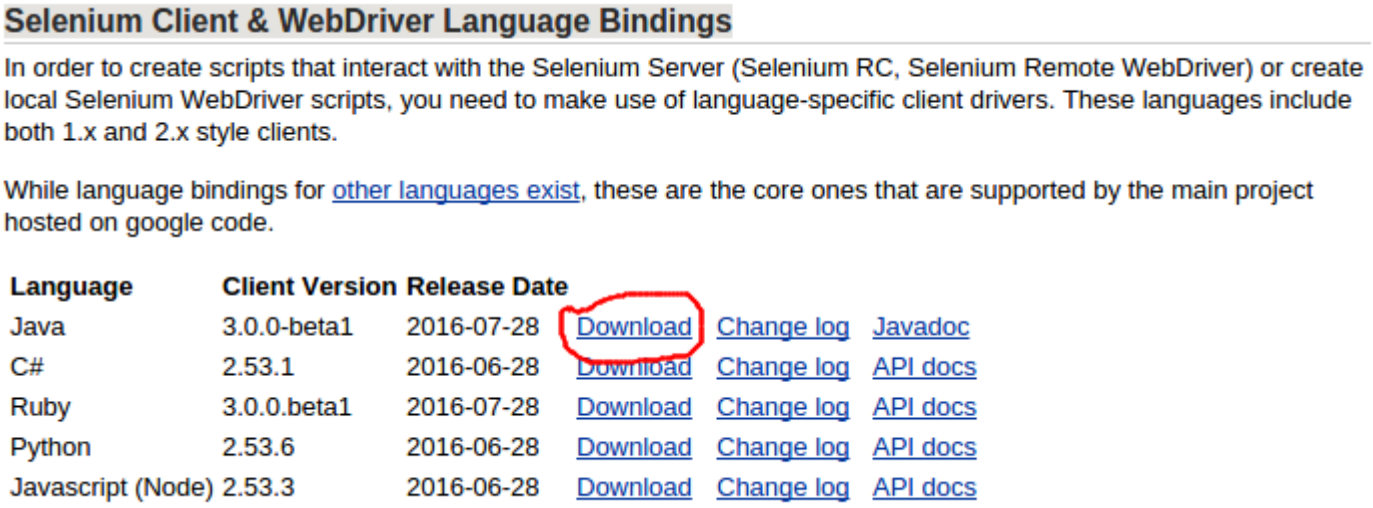
Instale el complemento m2eclipse haciendo clic derecho en su proyecto y seleccione Maven -> Habilitar administración de dependencias.

Pasos para configurar el proyecto Selenium Webdriver usando archivos Jar

1. Cree un nuevo proyecto en Eclipse siguiendo los pasos a continuación.

Abra Elipse -> Archivo -> Nuevo -> Proyecto Java -> Proporcione un nombre (aprendizaje de selenio) -> Finalizar

2. Descargue los archivos jar de http://www.seleniumhq.org/download/ . Debe descargar tanto **Selenium Standalone Server** como **Selenium Client y WebDriver Language Bindings** . Dado que este documento habla de Java, debe descargar solo el archivo jar de la sección de Java. Echa un vistazo en la captura de pantalla adjunta.



Selenium Client & WebDriver Language Bindings

In order to create scripts that interact with the Selenium Server (Selenium RC, Selenium Remote WebDriver) or create local Selenium WebDriver scripts, you need to make use of language-specific client drivers. These languages include both 1.x and 2.x style clients.

While language bindings for other languages exist, these are the core ones that are supported by the main project hosted on google code.

Language	Client Version	Release Date			
Java	3.0.0-beta1	2016-07-28	Download	Change log	Javadoc
C#	2.53.1	2016-06-28	Download	Change log	API docs
Ruby	3.0.0.beta1	2016-07-28	Download	Change log	API docs
Python	2.53.6	2016-06-28	Download	Change log	API docs
Javascript (Node)	2.53.3	2016-06-28	Download	Change log	API docs

Nota: Selenium Standalone Server solo es necesario si desea utilizar un servidor remoto para ejecutar las pruebas. Dado que este documento está arriba de la configuración del proyecto, es mejor tener todo en su lugar.

3. Los tarros se descargarán en un archivo zip, descomprímelos. Deberías poder ver `.jar` directamente.

4. En eclipse, haga clic con el botón derecho en el proyecto que creó en el paso 1 y siga los pasos a continuación.

   Propiedades -> Java Build Path -> Seleccione la pestaña Bibliotecas -> Haga clic en Agregar archivos externos -> Localice la carpeta del archivo descomprimido que descargó anteriormente -> Seleccione todos los archivos de la carpeta `lib` -> Haga clic en Aceptar -> Otra vez haga clic en Agregar archivos externos - > Localice la misma carpeta descomprimida -> Seleccione el archivo jar que está fuera de la carpeta lib (`client-combined-3.0.0-beta1-nodeps.jar`) -> Ok

Del mismo modo, agregue el `Selenium Standalone Server` siguiendo el paso anterior.

5. Ahora puedes comenzar a escribir código de selenio en tu proyecto.

PS : La documentación anterior se basa en la versión beta de selenium-3.0.0, por lo que los nombres de los archivos jar especificados pueden cambiar con la versión.

Lea Empezando con Selenium-webdriver en línea: https://riptutorial.com/es/selenium-webdriver/topic/878/empezando-con-selenium-webdriver

Capítulo 2: Acciones (Emulando gestos de usuario complejos)

Introducción

La clase de `Actions` nos permite emular con precisión cómo un usuario interactuaría con una página web / elementos. Usando una instancia de esta clase, puede describir una serie de acciones, como hacer clic, hacer doble clic, arrastrar, presionar teclas, etc. Una vez que se describen estas acciones, para poder llevar a cabo las acciones, debe realizar una llamada. (`.Build()`) y luego `.Build()` que se realicen (`.Perform()`). Así que debemos describir, construir, realizar. Los ejemplos a continuación se expandirán sobre esto.

Sintaxis

- dragAndDrop (fuente de WebElement, destino de WebElement)
- dragAndDropBy (fuente WebElement, int xOffset, int yOffset)
- realizar()

Parámetros

Parámetros	Detalles
fuente	Elemento para emular el botón hacia abajo en.
objetivo	Elemento para mover y soltar el mouse en.
xOffset	x coordinar para moverse a.
yOffset	y coordinar para moverse a.

Observaciones

Esta sección contiene información de la clase de Acciones de Selenium WebDriver. La clase Acciones le proporciona métodos convenientes para realizar gestos de usuarios complejos como arrastrar y soltar, mantener y hacer clic, etc.

Examples

Arrastrar y soltar

DO#

```
using OpenQA.Selenium;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Interactions;

namespace WebDriverActions
{
    class WebDriverTest
    {
        static void Main()
        {
            IWebDriver driver = new FirefoxDriver();

            driver.Navigate().GoToUrl("");
            IWebElement source = driver.FindElement(By.CssSelector(""));
            IWebElement target = driver.FindElement(By.CssSelector(""));
            Actions action = new Actions(driver);
            action.DragAndDrop(source, target).Perform();
        }
    }
}
```

Lo anterior encontrará un `IWebElement` , `source` , y lo arrastrará, y lo `IWebElement` en el segundo `target IWebElement` .

Java

Arrastre y suelte utilizando el elemento web de origen y destino.

Un método conveniente que realiza un clic y mantiene en la ubicación del elemento de origen, se mueve a la ubicación del elemento de destino, y luego libera el mouse.

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.interactions.Actions;

/**
 * Drag and Drop test using source and target webelement
 */
public class DragAndDropClass {
    public static void main(String[] args) {
        WebDriver driver = new FirefoxDriver();
        driver.get("");
        WebElement source = driver.findElement(By.cssSelector(""));
        WebElement target = driver.findElement(By.cssSelector(""));
        Actions action = new Actions(driver);
        action.build();
        action.dragAndDrop(source, target).perform();
    }
}
```

Arrastre un elemento y suéltelo en un desplazamiento dado.

Un método conveniente que realiza un clic y mantiene en la ubicación del elemento fuente, se

mueve en un desplazamiento dado (xey, ambos enteros), luego libera el mouse.

```
WebElement source = driver.findElement(By.cssSelector(""));
Actions action = new Actions(driver);
action.build()
action.dragAndDropBy(source, x, y).perform(); // x and y are integers value
```

Mover al elemento

DO#

Supongamos que desea probar que cuando se desplaza sobre un elemento, se muestra una lista desplegable. Es posible que desee verificar el contenido de esta lista, o tal vez seleccionar una opción de la lista.

Primero cree una acción, para desplazarse sobre el elemento *(por ejemplo, mi elemento tiene el texto de enlace "Admin")* :

```
Actions mouseHover = new Actions(driver);
mouseHover.MoveToElement(driver.FindElement(By.LinkText("Admin"))).Perform();
```

En el ejemplo anterior:

- Has creado la acción `mouseHover`
- Usted le ha dicho al `driver` que se mueva a un elemento específico
- Desde aquí puede realizar otras `Actions` con el objeto `mouseHover` o continuar con las pruebas con su objeto `driver`

Este enfoque es de particular uso cuando el hacer clic en un elemento realiza una función diferente a la del cursor sobre él.

Un ejemplo completo:

```
Actions mouseHover = new Actions(driver);
mouseHover.MoveToElement(driver.FindElement(By.LinkText("Admin"))).Perform();

Assert.IsTrue(driver.FindElement(By.LinkText("Edit Record")).Displayed);
Assert.IsTrue(driver.FindElement(By.LinkText("Delete Record")).Displayed);
```

Lea Acciones (Emulando gestos de usuario complejos) en línea: https://riptutorial.com/es/selenium-webdriver/topic/4849/acciones--emulando-gestos-de-usuario-complejos-

Capítulo 3: Configuración / Obtención del tamaño de la ventana del navegador

Introducción

Configurar o obtener el tamaño de la ventana de cualquier navegador durante la automatización

Sintaxis

- driver.manage (). window (). maximiza ();
- driver.manage (). window (). setSize (*DimensionObject*);
- driver.manage (). window (). getSize ()

Examples

JAVA

Establecer al tamaño máximo de la ventana del navegador:

```
//Initialize Browser
System.setProperty("webdriver.gecko.driver", "E:\\path\\to\\geckodriver.exe");
WebDriver driver = new FirefoxDriver();
driver.get("https://www.google.com/");

//Set Browser window size
driver.manage().window().maximize();
```

Establecer el tamaño específico de la ventana:

```
//Initialize Browser
System.setProperty("webdriver.gecko.driver", "E:\\path\\to\\geckodriver.exe");
WebDriver driver = new FirefoxDriver();
driver.get("https://www.google.com/");

//Initialize Dimension class object and set Browser window size
org.openqa.selenium.Dimension d = new org.openqa.selenium.Dimension(400, 500);
driver.manage().window().setSize(d);
```

Obtener el tamaño de la ventana del navegador:

```
//Initialize Browser
System.setProperty("webdriver.gecko.driver", "E:\\path\\to\\geckodriver.exe");
WebDriver driver = new FirefoxDriver();
driver.get("https://www.google.com/");

//Get Browser window size and print on console
System.out.println(driver.manage().window().getSize());
```

Lea Configuración / Obtención del tamaño de la ventana del navegador en línea: https://riptutorial.com/es/selenium-webdriver/topic/10093/configuracion---obtencion-del-tamano-de-la-ventana-del-navegador

Capítulo 4: Configuración de cuadrícula de selenio

Introducción

Selenium Grid es un marco para ejecutar pruebas distribuidas en una variedad de dispositivos de prueba. Se utiliza para probar aplicaciones web. Es posible escribir pruebas en diferentes lenguajes de programación populares, incluyendo C #, Groovy, Java, Perl, PHP, Python y Ruby. Las pruebas se pueden ejecutar en un rango de navegadores web en plataformas como Windows, Linux y OS X.

Es un software de código abierto, publicado bajo la licencia Apache 2.0: los desarrolladores web pueden descargarlo y usarlo sin costo alguno.

Sintaxis

- Para ejecutar el archivo jar, la siguiente es la sintaxis de cada archivo jar.
- `java -jar <jar-file-full-name>.jar -<your parameters if any>`

Parámetros

Parámetros	Detalles
papel	Es lo que le dice al selenio que era `hub` o `node`
Puerto	Esto es para especificar qué puerto debe escuchar el `hub` o `node` .
cubo	Este parámetro se usa en el `node` para especificar la url del hub
nombre del navegador	Se ha utilizado en el `node` para especificar el nombre del navegador como Firefox, Chrome, Internet Explorer.
maxInstances	Es donde se especifica la instancia del navegador, por ejemplo. 5 significa que habrá 5 instancias del navegador cuyo usuario especificado estará presente.
nodeConfig	Un archivo de configuración de Json para el nodo. Puede especificar el rol, puerto, etc. aquí
hubConfig	Un archivo de configuración de Json para el nodo. Puede especificar el rol, el puerto, las instancias máximas, etc. aquí

Examples

Código Java para Selenium Grid

```
String hubUrl = "http://localhost:4444/wd/hub"
DesiredCapabilities capability = DesiredCapabilities.firefox(); //or which browser you want
RemoteWebDriver driver = new RemoteWebDriver(hubUrl, capability);
```

Creando un nodo y nodo de Selenium Grid

Creando un hub

Una configuración rápida para una configuración de hub y nodo en la red de selenio. Para más información ver: Grid 2 docs.

Requerimientos

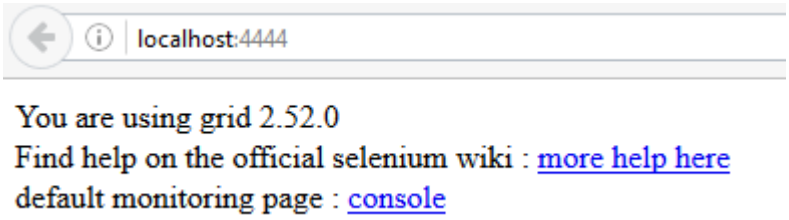
Para configurar un hub de rejilla necesita el flujo:

- Selenium-server-standalone-.jar

Creando el hub

Para crear un hub necesitas ejecutar el servidor selenium.

1. Descargar Selenium-server-standalone-.jar
2. Abra su terminal y navegue a la carpeta donde está Selenium-server-standalone-.jar
3. Ejecute el siguiente comando:
    1. Para la configuración predeterminada, `java -jar selenium-server-standalone-<Version>.jar -role hub`
    2. Para la configuración de Json `java -jar selenium-server-standalone-<Version>.jar -role hub -hubConfig hubConfig.json`
4. Abra http: // localhost: 4444 / verá un mensaje que sigue



Al hacer clic en la `console` -> `View config` para ver la Configuración de los detalles del concentrador.

Creando un nodo

Requerimientos

Para configurar un hub de rejilla necesita el flujo:

- Selenium-server-standalone-.jar
- Webdrivers
    - Driver cromo
    - Controlador de FireFox
    - Controlador Microsoft Edge
- Navegadores
    - Cromo
    - FireFox
    - Microsoft Edge (Windows 10)

Creando el nodo

Ahora para crear nodos para el hub

1. Descargar Selenium-server-standalone-.jar
2. Descarga los navegadores que quieras probar en
3. Descargue los controladores para los navegadores que desea probar en
4. Abra el nuevo terminal y navegue a la ubicación del archivo jar del servidor Selenium
5. Ejecute el siguiente comando:
    1. para la configuración predeterminada `java -jar selenium-server-standalone-<VERSION NUMBER>.jar -role node`
    2. Para la configuración de Json `java -jar selenium-server-standalone-<Version>.jar -role node -nodeConfig nodeConfig.json`
6. Ahora vaya a http: // localhost: 4444 / grid / console para ver los detalles del nodo

Configuragtion via Json

Una configuración de ejemplo para un hub:

```
java -jar selenium-server-standalone-<version>.jar -role hub -hubConfig hubConfig.json
```

```
{
    "_comment" : "Configuration for Hub - hubConfig.json",
    "host": ip,
    "maxSessions": 5,
    "port": 4444,
    "cleanupCycle": 5000,
    "timeout": 300000,
    "newSessionWaitTimeout": -1,
    "servlets": [],
    "prioritizer": null,
    "capabilityMatcher": "org.openqa.grid.internal.utils.DefaultCapabilityMatcher",
    "throwOnCapabilityNotPresent": true,
    "nodePolling": 180000,
    "platform": "WINDOWS"
}
```

Una configuración de ejemplo para un nodo

```
java -jar selenium-server-standalone-<version>.jar -role node -nodeConfig nodeConfig.json
```

```
{
  "capabilities":
  [
    {
      "browserName": "opera",
      "platform": "WINDOWS",
      "maxInstances": 5,
      "seleniumProtocol": "WebDriver",
      "webdriver.opera.driver": "C:/Selenium/drivers/operadriver.exe",
      "binary":"C:/Program Files/Opera/44.0.2510.1159/opera.exe"
    },
    {
      "browserName": "chrome",
      "platform": "WINDOWS",
      "maxInstances": 5,
      "seleniumProtocol": "WebDriver",
      "webdriver.chrome.driver": "C:/Selenium/drivers/chromedriver.exe",
      "binary":"C:/Program Files/Google/Chrome/Application/chrome.exe"
    },
    {
      "browserName": "firefox",
      "platform": "WINDOWS",
      "maxInstances": 5,
      "seleniumProtocol": "WebDriver",
      "webdriver.gecko.driver": "C:/Selenium/drivers/geckodriver.exe",
      "binary":"C:/Program Files/Mozilla Firefox/firefox.exe"
    }
  ],
  "proxy": "org.openqa.grid.selenium.proxy.DefaultRemoteProxy",
  "maxSession": 5,
  "port": 5555,
  "register": true,
  "registerCycle": 5000,
  "hub": "http://localhost:4444",
  "nodeStatusCheckTimeout": 5000,
  "nodePolling": 5000,
  "role": "node",
  "unregisterIfStillDownAfter": 60000,
  "downPollingLimit": 2,
  "debug": false,
  "servlets" : [],
  "withoutServlets": [],
  "custom": {}
}
```

Lea Configuración de cuadrícula de selenio en línea: https://riptutorial.com/es/selenium-webdriver/topic/2504/configuracion-de-cuadricula-de-selenio

Capítulo 5: De desplazamiento

Introducción

Este tema proporcionará varios enfoques sobre cómo realizar el desplazamiento con `selenium`

Examples

Desplazamiento usando Python

1. Desplazarse al elemento de destino (botón "NAVEGAR PLANTILLAS" en la parte inferior de la página) con `Actions`

```
from selenium import webdriver
from selenium.webdriver.common.action_chains import ActionChains

driver = webdriver.Chrome()
driver.get('http://www.w3schools.com/')
target = driver.find_element_by_link_text('BROWSE TEMPLATES')
actions = ActionChains(driver)
actions.move_to_element(target)
actions.perform()
```

2. Desplácese hasta el elemento de destino (botón "BROWSE TEMPLATES" en la parte inferior de la página) con `JavaScript`

```
from selenium import webdriver

driver = webdriver.Chrome()
driver.get('http://www.w3schools.com/')
target = driver.find_element_by_link_text('BROWSE TEMPLATES')
driver.execute_script('arguments[0].scrollIntoView(true);', target)
```

3. Desplácese hasta el elemento de destino (botón "BROWSE TEMPLATES" en la parte inferior de la página) con el método incorporado

```
from selenium import webdriver

    driver = webdriver.Chrome()
    driver.get('http://www.w3schools.com/')
    target = driver.find_element_by_link_text('BROWSE TEMPLATES')
    target.location_once_scrolled_into_view
```

Tenga en cuenta que `location_once_scrolled_into_view` también devuelve las coordenadas `x` , `y` del elemento después de desplazarse

4. Desplazarse hasta la parte inferior de la página con las `Keys`

```
from selenium import webdriver
```

```
from selenium.webdriver.common.keys import Keys

driver = webdriver.Chrome()
driver.get('http://www.w3schools.com/')
driver.find_element_by_tag_name('body').send_keys(Keys.END) # Use send_keys(Keys.HOME) to
scroll up to the top of page
```

Tenga en cuenta que `send_keys(Keys.DOWN)` / `send_keys(Keys.UP)` y `send_keys(Keys.PAGE_DOWN)` / `send_keys(Keys.PAGE_UP)` también podrían usarse para desplazarse

Desplazamiento diferente usando java de diferentes maneras.

A continuación, la solución para dar también se puede usar en otros lenguajes de programación compatibles con algunos cambios de sintaxis

---

1. Para **desplazarse hacia abajo en la** página / sección / división en la página web mientras haya una barra de desplazamiento personalizada (no es el desplazamiento del navegador). Haga clic aquí para ver la demo y verifique que la barra de desplazamiento tenga su elemento independiente.

   En el siguiente código dado pase el elemento de la barra de desplazamiento y requiera puntos de desplazamiento.

```
    public static boolean scroll_Page(WebElement webelement, int scrollPoints)
    {
    try
    {
        System.out.println("---------------- Started - scroll_Page ----------------");
        driver = ExecutionSetup.getDriver();
        dragger = new Actions(driver);

        // drag downwards
        int numberOfPixelsToDragTheScrollbarDown = 10;
        for (int i = 10; i < scrollPoints; i = i + numberOfPixelsToDragTheScrollbarDown)
        {
            dragger.moveToElement(webelement).clickAndHold().moveByOffset(0,
numberOfPixelsToDragTheScrollbarDown).release(webelement).build().perform();
        }
        Thread.sleep(500);
        System.out.println("---------------- Ending - scroll_Page ----------------");
        return true;
    }
    catch (Exception e)
    {
        System.out.println("---------------- scroll is unsucessfully done in scroll_Page -----
-----------");
        e.printStackTrace();
        return false;
    }
  }
```

---

2. Para hacerlo **de desplazamiento ascendente** página / sección / división en la página web mientras que hay barra de desplazamiento a medida (no el navegador de desplazamiento).

Haga clic aquí para ver la demo y verifique que la barra de desplazamiento tenga su elemento independiente.

En el siguiente código dado pase el elemento de la barra de desplazamiento y requiera puntos de desplazamiento.

```
public static boolean scroll_Page_Up(WebElement webelement, int scrollPoints)
{
    try
    {
        System.out.println("---------------- Started - scroll_Page_Up ----------------");
        driver = ExecutionSetup.getDriver();
        dragger = new Actions(driver);
        // drag upwards
        int numberOfPixelsToDragTheScrollbarUp = -10;
        for (int i = scrollPoints; i > 10; i = i + numberOfPixelsToDragTheScrollbarUp)
        {
            dragger.moveToElement(webelement).clickAndHold().moveByOffset(0,
numberOfPixelsToDragTheScrollbarUp).release(webelement).build().perform();
        }
        System.out.println("---------------- Ending - scroll_Page_Up ----------------");
        return true;
    }
    catch (Exception e)
    {
        System.out.println("---------------- scroll is unsucessfully done in scroll_Page_Up---
-------------");
        e.printStackTrace();
        return false;
    }
}
```

---

3. Para desplazarse hacia abajo cuando el **navegador múltiple se desplaza** (navegador integrado) y desea desplazarse hacia abajo con la **tecla de avance de página** . Haga clic aquí para la demostración

   En el código dado a continuación, pase su elemento de área de desplazamiento como `<div>` y requiera la tecla de página abajo.

```
 public static boolean pageDown_New(WebElement webeScrollArea, int iLoopCount)
 {
    try
    {
        System.out.println("---------------- Started - pageDown_New ----------------");
        driver = ExecutionSetup.getDriver();
        dragger = new Actions(driver);

        for (int i = 0; i <= iLoopCount; i++)
        {

dragger.moveToElement(webeScrollArea).click().sendKeys(Keys.PAGE_DOWN).build().perform();
        }
        System.out.println"---------------- Ending - pageDown_New ----------------");
        return true;
    }
    catch (Exception e)
```

```
    {
        System.out.println("---------------- Not able to do page down ----------------");
        return false;
    }
}
```

---

4. Para desplazarse hacia arriba cuando el **navegador múltiple se desplaza** (navegador integrado) y desea desplazarse hacia arriba con la **tecla Page UP** . Haga clic aquí para la demostración

   En el código dado a continuación, pase su elemento de área de desplazamiento como `<div>` y requiera una página arriba.

```
public static boolean pageUp_New(WebElement webeScrollArea, int iLoopCount)
{
    try
    {
        System.out.println("---------------- Started - pageUp_New ----------------");
        driver = ExecutionSetup.getDriver();
        dragger = new Actions(driver);

        for (int i = 0; i <= iLoopCount; i++)
        {

dragger.moveToElement(webeScrollArea).click().sendKeys(Keys.PAGE_UP).build().perform();
        }
        System.out.println("---------------- Ending - pageUp_New ----------------");
        return true;
    }
    catch (Exception e)
    {
        System.out.println("---------------- Not able to do page up ----------------");
        return false;
    }
}
```

---

5. Para desplazarse hacia abajo cuando el **navegador múltiple se desplaza** (navegador integrado) y desea desplazarse hacia abajo con la **tecla de flecha solo hacia abajo** . Haga clic aquí para la demostración

   En el código dado a continuación, pase su elemento de área de desplazamiento como `<div>` y requiera la tecla hacia abajo.

```
public static boolean scrollDown_Keys(WebElement webeScrollArea, int iLoopCount)
{
    try
    {
        System.out.println("---------------- Started - scrollDown_Keys ----------------");
        driver = ExecutionSetup.getDriver();
        dragger = new Actions(driver);

        for (int i = 0; i <= iLoopCount; i++)
        {
```

```
dragger.moveToElement(webeScrollArea).click().sendKeys(Keys.DOWN).build().perform();
        }
        System.out.println("---------------- Ending - scrollDown_Keys ----------------");
        return true;
    }
    catch (Exception e)
    {
        System.out.println("---------------- Not able to do scroll down with keys-------------
---");
        return false;
    }
}
```

---

6. Para desplazarse hacia arriba cuando el **navegador múltiple se desplaza** (navegador integrado) y desea desplazarse hacia arriba con la **tecla de flecha hacia arriba solamente** . Haga clic aquí para la demostración

   En el código dado a continuación, pase su elemento de área de desplazamiento como `<div>` y requiera una tecla arriba.

```
public static boolean scrollUp_Keys(WebElement webeScrollArea, int iLoopCount)
{
    try
    {
        System.out.println("---------------- Started - scrollUp_Keys ----------------");
        driver = ExecutionSetup.getDriver();
        dragger = new Actions(driver);

        for (int i = 0; i <= iLoopCount; i++)
        {
            dragger.moveToElement(webeScrollArea).click().sendKeys(Keys.UP).build().perform();
        }
        System.out.println("---------------- Ending - scrollUp_Keys ----------------");
        return true;
    }
    catch (Exception e)
    {
        System.out.println("---------------- Not able to do scroll up with keys---------------
-");
        return false;
    }
}
```

---

7. Para desplazarse hacia arriba / abajo cuando el **navegador se desplaza** (navegador integrado) y desea desplazarse hacia arriba / abajo con **solo un punto fijo** . Haga clic aquí para la demostración

   En el siguiente código dado pase su punto de desplazamiento. Positivo significa abajo y negativo significa desplazarse hacia arriba.

```
public static boolean scroll_without_WebE(int scrollPoint)
{
    JavascriptExecutor jse;
    try
```

```
    {
        System.out.println("---------------- Started - scroll_without_WebE ----------------");

        driver = ExecutionSetup.getDriver();
        jse = (JavascriptExecutor) driver;
        jse.executeScript("window.scrollBy(0," + scrollPoint + ")", "");

        System.out.println("---------------- Ending - scroll_without_WebE ----------------");
        return true;
    }
    catch (Exception e)
    {
        System.out.println("---------------- scroll is unsucessful in scroll_without_WebE ----
------------");
        e.printStackTrace();
        return false;
    }
}
```

---

8. Para desplazarse hacia arriba / abajo cuando el **navegador navega** (navegador incorporado) y desea desplazarse hacia arriba / abajo para **hacer elemento en el área visible o desplazamiento dinámico** . Haga clic aquí para la demostración

   En el siguiente código dado pase su elemento.

```
public static boolean scroll_to_WebE(WebElement webe)
{
    try
    {
        System.out.println("---------------- Started - scroll_to_WebE ----------------");

        driver = ExecutionSetup.getDriver();
        ((JavascriptExecutor) driver).executeScript("arguments[0].scrollIntoView();", webe);

        System.out.println("---------------- Ending - scroll_to_WebE ----------------");
        return true;
    }
    catch (Exception e)
    {
        System.out.println("---------------- scroll is unsucessful in scroll_to_WebE ---------
-------");
        e.printStackTrace();
        return false;
    }
}
```

Nota: Por favor verifique su caso y utilice métodos. Si falta algún caso, hágamelo saber.

Lea De desplazamiento en línea: https://riptutorial.com/es/selenium-webdriver/topic/9063/de-desplazamiento

Capítulo 6: Ejecución de Javascript en la página.

Sintaxis

- object ExecuteAsyncScript (secuencia de comandos de cadena, objeto params [] args);
- objeto ExecuteScript (secuencia de comandos de cadena, objeto params [] args);

Examples

DO#

Para ejecutar JavaScript en una instancia de `IWebDriver` , debe convertir `IWebDriver` a una nueva interfaz, `IJavaScriptExecutor`

```
IWebDriver driver;
IJavaScriptExecutor jsDriver = driver as IJavaScriptExecutor;
```

Ahora puede acceder a todos los métodos disponibles en la instancia de `IJavaScriptExecutor` que le permiten ejecutar Javascript, por ejemplo:

```
jsDriver.ExecuteScript("alert('running javascript');");
```

Pitón

Para ejecutar Javascript en python, use `execute_script("javascript script here")` . execute_script se llama en una instancia de webdriver, y puede ser cualquier javascript válido.

```
from selenium import webdriver
driver = webdriver.Chrome()
driver.execute_script("alert('running javascript');")
```

Java

Para ejecutar Javascript en Java, cree un nuevo controlador web que admita Javascript. Para usar la función `executeScript()` , el controlador se debe convertir a un `JavascriptExecutor` , o una nueva variable se puede establecer en el valor del controlador fundido: `((JavascriptExecutor)driver)` controlador `((JavascriptExecutor)driver) . driver.executeScript()` una cadena que es Javascript válido.

```
WebDriver driver = new ChromeDriver();
JavascriptExecutor JavascriptExecutor = ((JavascriptExecutor)driver);
JavascriptExecutor.executeScript("alert('running javascript');");
```

Rubí

```
require "selenium-webdriver"

driver = Selenium::WebDriver.for :chrome
driver.execute_script("alert('running javascript');")
```

Lea Ejecución de Javascript en la página. en línea: https://riptutorial.com/es/selenium-webdriver/topic/6986/ejecucion-de-javascript-en-la-pagina-

Capítulo 7: Espere

Examples

Tipos de Espera en Selenium WebDriver

Al ejecutar cualquier aplicación web, es necesario tener en cuenta el tiempo de carga. Si su código intenta acceder a cualquier elemento que aún no esté cargado, WebDriver lanzará una excepción y su secuencia de comandos se detendrá.

Hay tres tipos de esperas:

- **Esperas implícitas**
- **Esperas explícitas**
- **Espera fluida**

Las esperas implícitas se usan para establecer el tiempo de espera en todo el programa, mientras que las esperas explícitas se usan solo en partes específicas.

Espera implícita

Una espera implícita es decirle a WebDriver que sondee el DOM durante un cierto período de tiempo cuando intenta encontrar un elemento o elementos si no están disponibles de inmediato. Las esperas implícitas son básicamente su forma de decirle a WebDriver la latencia que desea ver si el elemento web especificado no está presente que WebDriver está buscando. La configuración predeterminada es 0. Una vez establecida, la espera implícita se establece para la vida de la instancia del objeto WebDriver. La espera implícita se declara en la parte de creación de instancias del código mediante el siguiente fragmento de código.

Ejemplo en **Java** :

```
driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);
// You need to import the following class - import java.util.concurrent.TimeUnit;
```

Ejemplo en **C #** :

```
driver.Manage().Timeouts().ImplicitlyWait(TimeSpan.FromSeconds(15));
```

Entonces, en este caso, le está diciendo a WebDriver que debe esperar 15 segundos en los casos en que el elemento especificado no esté disponible en la interfaz de usuario (DOM).

Espera explícita

Puede encontrar instancias en las que algún elemento demore más en cargarse. Establecer la espera implícita para tales casos no tiene sentido ya que el navegador esperará innecesariamente el mismo tiempo para cada elemento, lo que aumenta el tiempo de automatización. La espera explícita ayuda aquí al pasar por alto la espera implícita para algunos elementos específicos.

Las esperas explícitas son esperas inteligentes que se limitan a un elemento web en particular. Usando esperas explícitas, básicamente le está diciendo a WebDriver al máximo que debe esperar X unidades de tiempo antes de que se rinda.

Las esperas explícitas se realizan utilizando las clases WebDriverWait y ExpectedConditions. En el siguiente ejemplo, esperaremos hasta 10 segundos para que un elemento cuyo id es un nombre de usuario se haga visible antes de continuar con el siguiente comando. Aquí están los pasos.

Ejemplo en **Java** :

```
//Import these two packages:
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

//Declare a WebDriverWait variable. In this example, we will use myWaitVar as the name of the
variable.
WebDriverWait myWaitVar = new WebDriverWait(driver, 30);

//Use myWaitVar with ExpectedConditions on portions where you need the explicit wait to occur.
In this case, we will use explicit wait on the username input before we type the text tutorial
onto it.
myWaitVar.until(ExpectedConditions.visibilityOfElementLocated(By.id(“username”)));
driver.findElement(By.id(“username”)).sendKeys(“tutorial”);
```

La clase ExpectedConditions tiene algunas condiciones comunes predefinidas para esperar un elemento. Haga clic aquí para ver la lista de estas condiciones en el enlace de Java.

Ejemplo en **C #** :

```
using OpenQA.Selenium;
using OpenQA.Selenium.Support.UI;
using OpenQA.Selenium.PhantomJS;

// You can use any other WebDriver you want, such as ChromeDriver.
using (var driver = new PhantomJSDriver())
{
    driver.Navigate().GoToUrl("http://somedomain/url_that_delays_loading");

    // We aren't going to use it more than once, so no need to declare this a variable.
    new WebDriverWait(driver, TimeSpan.FromSeconds(10))
        .Until(ExpectedConditions.ElementIsVisible(By.Id("element-id")));

    // After the element is detected by the previous Wait,
    // it will display the element's text
    Console.WriteLine(driver.FindElement(By.Id("element-id")).Text);
}
```

En este ejemplo, el sistema esperará 10 segundos hasta que el elemento esté visible. Si el elemento no será visible después del tiempo de espera, el WebDriver lanzará una `WebDriverTimeoutException` .

Tenga en cuenta que: si el elemento es visible antes del tiempo de espera de 10 segundos, el sistema procederá inmediatamente a continuar con el proceso.

Espera fluida

A diferencia de la espera implícita y explícita, la espera fluida utiliza dos parámetros. Tiempo de espera y frecuencia de sondeo. Digamos que tenemos un valor de tiempo de espera de 30 segundos y la frecuencia de sondeo de 2 segundos. WebDriver comprobará el elemento después de cada 2 segundos hasta el valor de tiempo de espera (30 segundos). Después de que se excede el valor de tiempo de espera sin ningún resultado, se lanza una excepción. A continuación se muestra un código de ejemplo que muestra la implementación de espera fluida.

Ejemplo en **Java** :

```
Wait wait = new FluentWait(driver).withTimeout(30, SECONDS).pollingEvery(2,
SECONDS).ignoring(NoSuchElementException.class);


WebElement testElement = wait.until(new Function() {
    public WebElement apply(WebDriver driver) {
        return driver.findElement(By.id("testId"));
    }
});
```

Otra ventaja de usar la espera fluida es que podemos ignorar tipos específicos de excepciones (por ejemplo, NoSuchElementExceptions) mientras esperamos. Debido a todas estas disposiciones, la espera fluida es útil en las aplicaciones AJAX, así como en escenarios cuando el tiempo de carga del elemento fluctúa con frecuencia. El uso estratégico de la espera fluida mejora significativamente los esfuerzos de automatización.

Diferentes tipos de condiciones de espera explícitas.

En espera explícita, esperas que ocurra una condición. Por ejemplo, desea esperar hasta que se pueda hacer clic en un elemento.

Aquí hay una demostración de algunos problemas comunes.

Tenga en cuenta: En todos estos ejemplos, puede usar cualquier `By` como un localizador, como nombre de `classname` , `xpath` , `link text` , `tag name` o `cssSelector`

Espera hasta que el elemento sea visible

Por ejemplo, si su sitio web tarda en cargarse, puede esperar hasta que la página finalice la carga y su elemento sea visible para el WebDriver.

DO#

```
WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));
wait.Until(ExpectedConditions.ElementIsVisible(By.Id("element-id")));
```

Java

```
WebDriverWait wait = new WebDriverWait(driver, 10);
wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("element-id")));
```

Espera hasta que el elemento ya no sea visible.

Igual que antes, pero invertido.

DO#

```
WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));
wait.Until(ExpectedConditions.InvisibilityOfElementLocated(By.Id("element-id")));
```

Java

```
WebDriverWait wait = new WebDriverWait(driver, 10);
wait.until(ExpectedConditions.invisibilityOfElementLocated(By.id("element-id")));
```

Espera hasta que el texto esté presente en el elemento especificado.

DO#

```
IWebElement element = driver.FindElement(By.Id("element-id"));

WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));
wait.Until(ExpectedConditions.TextToBePresentInElement(element, "text"));
```

Java

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

WebDriverWait wait = new WebDriverWait(driver, 10);
wait.until(ExpectedConditions.textToBePresentInElement(element, "text"));
```

Si vas al enlace dado arriba, verás todas las condiciones de espera allí.

La diferencia entre el uso de estas condiciones de espera se encuentra en su parámetro de entrada.

Eso significa que necesita pasar el elemento Web si su parámetro de entrada es WebElement, debe pasar el localizador de elementos si toma el localizador Por como su parámetro de entrada.

Elija sabiamente qué tipo de condición de espera desea utilizar.

Esperando solicitudes de Ajax para completar

DO#

```
using OpenQA.Selenium
using OpenQA.Selenium.Chrome;
using System.Threading;

namespace WebDriver Tests
{
    class WebDriverWaits
    {
        static void Main()
        {
            IWebDriver driver = new ChromeDriver(@"C:\WebDriver");

            driver.Navigate().GoToUrl("page with ajax requests");
            CheckPageIsLoaded(driver);

            // Now the page is fully loaded, you can continue with further tests.
        }

        private void CheckPageIsLoaded(IWebDriver driver)
        {
            while (true)
            {
                bool ajaxIsComplete = (bool)(driver as
IJavaScriptExecutor).ExecuteScript("return jQuery.active == 0");
                if (ajaxIsComplete)
                    return;
                Thread.Sleep(100);
            }
        }
    }
}
```

Este ejemplo es útil para páginas donde se realizan solicitudes ajax, aquí usamos el `IJavaScriptExecutor` para ejecutar nuestro propio código JavaScript. Ya que está dentro de un `while` de bucle seguirá funcionando hasta que `ajaxIsComplete == true` y así se ejecuta la instrucción de retorno.

Verificamos que todas las solicitudes de ajax estén completas al confirmar que `jQuery.active` es igual a `0` . Esto funciona porque cada vez que se realiza una nueva solicitud de ajax `jQuery.active` se incrementa y cada vez que una solicitud complementa se decrementa, de esto podemos deducir que cuando `jQuery.active == 0` todas las solicitudes de ajax deben completarse.

Espera fluida

La espera fluida es una superclase de espera **explícita** (`WebDriverWait`) que es más configurable ya que puede aceptar un argumento para la función de espera. **Transmitiré la** espera **implícita** , ya que es una buena práctica evitarla.

Uso (Java):

```
Wait wait = new FluentWait<>(this.driver)
        .withTimeout(driverTimeoutSeconds, TimeUnit.SECONDS)
        .pollingEvery(500, TimeUnit.MILLISECONDS)
        .ignoring(StaleElementReferenceException.class)
        .ignoring(NoSuchElementException.class)
        .ignoring(ElementNotVisibleException.class);

WebElement foo = wait.until(ExpectedConditions.presenceOfElementLocated(By.yourBy));

// or use your own predicate:
WebElement foo = wait.until(new Function() {
  public WebElement apply(WebDriver driver) {
    return element.getText().length() > 0;
  }
});
```

Cuando usa la espera explícita con sus valores predeterminados, es simplemente un `FluentWait<WebDriver>` con valores predeterminados de: `DEFAULT_SLEEP_TIMEOUT = 500;` y haciendo `NotFoundException` omiso de `NotFoundException` .

Espera fluida

Cada instancia de FluentWait define la cantidad máxima de tiempo para esperar una condición, así como la frecuencia con la que se verifica la condición. Además, el usuario puede configurar la espera para ignorar tipos específicos de excepciones mientras espera, como NoSuchElementExceptions al buscar un elemento en la página. Está asociado con el controlador.

```
Wait<WebDriver> wait = new FluentWait<WebDriver>(driver)
.withTimeout(30, SECONDS) //actuall wait for the element to pe present
.pollingEvery(5, SECONDS) //selenium will keep looking for the element after every 5seconds
.ignoring(NoSuchElementException.class); //while ignoring this condition
wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.id("username"));
```

Lea Espere en línea: https://riptutorial.com/es/selenium-webdriver/topic/4435/espere

Capítulo 8: Excepciones en Selenium-WebDriver

Introducción

Hay una serie de excepciones que pueden ser lanzadas al usar un webdriver. Los ejemplos a continuación están destinados a dar una idea de lo que significan.

Examples

Excepciones de Python

Documentación de excepciones de selenio

ElementNotInteractableException: se lanza cuando un elemento está presente en el DOM pero las interacciones con ese elemento afectarán a otro elemento debido al orden de la pintura

- **ElementNotSelectableException: se** lanza cuando se intenta seleccionar un elemento no seleccionable. Ejemplos de elementos no seleccionables:
  - guión
- **ElementNotVisibleException: se** lanza cuando un elemento está presente en el DOM, pero no es visible y, por lo tanto, no se puede interactuar con él. Se encuentra con más frecuencia al intentar hacer clic o leer el texto de un elemento que está oculto a la vista.
- **ErrorInResponseException: se lanza** cuando se produce un error en el servidor. Esto puede suceder cuando se comunica con la extensión de Firefox o el servidor del controlador remoto.
- **ImeActivationFailedException: se ha producido un error** al activar un motor IME.
- **ImeNotAvailableException: se lanza** cuando el soporte de IME no está disponible. Esta excepción se produce para cada llamada de método relacionada con IME si el soporte de IME no está disponible en la máquina.
- **InvalidArgumentException:** los argumentos pasados a un comando no son válidos o tienen un formato incorrecto.
- **InvalidCookieDomainException: se** lanza al intentar agregar una cookie en un dominio diferente al de la URL actual.
- **InvalidElementStateException: se lanza** cuando una acción da como resultado un estado no válido para un elemento. Subclases:
  - ElementNotInteractableException
  - ElementNotSelectableException
  - ElementNotVisibleException
- **InvalidSelectorException: se lanza** cuando el selector que se usa para encontrar un elemento no devuelve un elemento web. Actualmente, esto solo sucede cuando el selector es una expresión xpath y es sintácticamente no válido (es decir, no es una expresión xpath) o la expresión no selecciona WebElements (por ejemplo, "count (// input)").

- **InvalidSwitchToTargetException: se lanza** cuando no existe el objetivo de marco o ventana que se va a cambiar.
- **MoveTargetOutOfBoundsException: se lanza** cuando el objetivo proporcionado al método move () de ActionsChains no es válido, es decir, está fuera del documento.
- **NoAlertPresentException: se lanza** cuando se cambia a una alerta sin presentar. Esto puede deberse a una operación en la clase Alert () cuando todavía no hay una alerta en la pantalla.
- **NoSuchAttributeException: se lanza** cuando no se puede encontrar el atributo del elemento. Es posible que desee verificar si el atributo existe en el navegador en particular con el que está probando. Algunos navegadores pueden tener diferentes nombres de propiedad para la misma propiedad. (IE8's .innerText vs. Firefox .textContent)
- **NoSuchElementException: se lanza** cuando no se puede encontrar el elemento. Si encuentra esta excepción, puede querer verificar lo siguiente:
    - Compruebe su selector utilizado en su find_by ...
    - Es posible que el elemento aún no esté en la pantalla en el momento de la operación de búsqueda, (la página web todavía se está cargando) vea selenium.webdriver.support.wait.WebDriverWait () para saber cómo escribir un contenedor de espera para esperar a que aparezca un elemento.
- **NoSuchFrameException: se lanza** cuando el objetivo de cuadro que se va a cambiar no existe.
- **NoSuchWindowException: se lanza** cuando no existe el objetivo de ventana que se va a cambiar. Para encontrar el conjunto actual de manejadores de ventana activos, puede obtener una lista de los manejadores de ventana activos de la siguiente manera:
  `print driver.window_handles`
- **RemoteDriverServerException:**
- **StaleElementReferenceException: se lanza** cuando una referencia a un elemento ahora está "obsoleta". Stale significa que el elemento ya no aparece en el DOM de la página. Las posibles causas de StaleElementReferenceException incluyen, entre otras:
    - Ya no está en la misma página, o la página puede haberse actualizado desde que se localizó el elemento.
    - Es posible que el elemento se haya eliminado y se haya vuelto a agregar a la pantalla, ya que estaba ubicado. Como un elemento que se está reubicando. Esto puede suceder normalmente con un marco javascript cuando los valores se actualizan y el nodo se reconstruye.
    - El elemento puede haber estado dentro de un iframe u otro contexto que se actualizó.
- **TimeoutException: Se lanza** cuando un comando no se completa con el tiempo suficiente.
- **UnableToSetCookieException: se** lanza cuando un controlador no puede establecer una cookie.
- **UnexpectedAlertPresentException: se** lanza cuando aparece una alerta inesperada. Generalmente se genera cuando un modo esperado está bloqueando el formulario webdriver ejecutando más comandos.
- **UnexpectedTagNameException: se** lanza cuando una clase de soporte no obtiene un elemento web esperado.
- **WebDriverException:** excepción del controlador web base. Todas las excepciones de webdriver usan la excepción WebDriverException o InvalidStateException como clase principal.

Lea Excepciones en Selenium-WebDriver en línea: https://riptutorial.com/es/selenium-webdriver/topic/10546/excepciones-en-selenium-webdriver

Capítulo 9: Informes HTML

Introducción

Este tema cubre la creación de informes HTML para pruebas de selenio. Hay varios tipos de complementos disponibles para informes y los más utilizados son Allure, ExtentReports y ReportNG.

Examples

ExtentReports

Este ejemplo cubre la implementación de ExtentReports en Selenium usando TestNG, Java y Maven.

Los ExtentReports están disponibles en dos versiones, comunitaria y comercial. Para la facilidad y el propósito de demostración, usaremos la versión de la comunidad.

1. Dependencia

Agregue la dependencia en su archivo pom.xml de Maven para los informes de extensión.

```
<dependency>
     <groupId>com.aventstack</groupId>
     <artifactId>extentreports</artifactId>
     <version>3.0.6</version>
 </dependency>
```

2. Configurar complementos

Configure el plugin maven surefire como se muestra a continuación en pom.xml

```
 <build>
 <defaultGoal>clean test</defaultGoal>
 <plugins>
     <plugin>
         <groupId>org.apache.maven.plugins</groupId>
         <artifactId>maven-compiler-plugin</artifactId>
         <version>3.6.1</version>
         <configuration>
             <source>${jdk.level}</source>
             <target>${jdk.level}</target>
         </configuration>
     </plugin>
     <plugin>
         <groupId>org.apache.maven.plugins</groupId>
         <artifactId>maven-surefire-plugin</artifactId>
         <version>2.19.1</version>
         <configuration>
             <suiteXmlFiles>
```

```
                    <suiteXmlFile>testng.xml</suiteXmlFile>
                </suiteXmlFiles>
            </configuration>
        </plugin>
    </plugins>
</build>
```

3. Prueba de muestra con ExtentReports

Ahora, crea una prueba con el nombre test.java

```
public class TestBase {
    WebDriver driver;

    ExtentReports extent;
    ExtentTest logger;
    ExtentHtmlReporter htmlReporter;
    String htmlReportPath = "C:\\Screenshots/MyOwnReport.html"; //Path for the HTML report to
be saved

    @BeforeTest
    public void setup(){
        htmlReporter = new ExtentHtmlReporter(htmlReportPath);
        extent = new ExtentReports();
        extent.attachReporter(htmlReporter);

        System.setProperty("webdriver.chrome.driver", "pathto/chromedriver.exe");
        driver = new ChromeDriver();

    }

    @Test
    public void test1(){
        driver.get("http://www.google.com/");
        logger.log(Status.INFO, "Opened site google.com");
        assertEquals(driver.getTitle()), "Google");
        logger.log(Status.PASS, "Google site loaded");
    }

    @AfterMethod
    public void getResult(ITestResult result) throws Exception {
        if (result.getStatus() == ITestResult.FAILURE)
        {
            logger.log(Status.FAIL, MarkupHelper.createLabel(result.getName() + " Test case
FAILED due to below issues:", ExtentColor.RED));
            logger.fail(result.getThrowable());
        }
        else if (result.getStatus() == ITestResult.SUCCESS)
        {
            logger.log(Status.PASS, MarkupHelper.createLabel(result.getName() + " Test Case
PASSED", ExtentColor.GREEN));
        }
        else if (result.getStatus() == ITestResult.SKIP)
        {
            logger.log(Status.SKIP, MarkupHelper.createLabel(result.getName() + " Test Case
SKIPPED", ExtentColor.BLUE));
        }
     }

    @AfterTest
```

```
    public void testend() throws Exception {
        extent.flush();
    }

    @AfterClass
    public void tearDown() throws Exception {
        driver.close();
    }
```

Informes de encanto

Este ejemplo cubre la implementación de Allure Reports en Selenium utilizando TestNG, Java y Maven.

Configuración de Maven

Repositorio

Agregue el siguiente código para configurar el repositorio jcenter

```
<repository>
            <id>jcenter</id>
            <name>bintray</name>
            <url>http://jcenter.bintray.com</url>
</repository>
```

Dependencia

Agregue las siguientes dependencias a su pom.xml

```
<dependency>
        <groupId>org.aspectj</groupId>
        <artifactId>aspectjweaver</artifactId>
        <version>${aspectj.version}</version>
    </dependency>
    <dependency>
        <groupId>ru.yandex.qatools.allure</groupId>
        <artifactId>allure-testng-adaptor</artifactId>
        <version>1.5.4</version>
    </dependency>
```

Configuración del complemento Surefire

```
<plugin>
        <groupId> org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>2.20</version>
        <configuration>
            <argLine>-
```

```
javaagent:${settings.localRepository}/org/aspectj/aspectjweaver/${aspectj.version}/aspectjweaver-
${aspectj.version}.jar
            </argLine>
            <properties>
                <property>
                    <name>listener</name>
                    <value>ru.yandex.qatools.allure.testng.AllureTestListener</value>
                </property>
            </properties>
            <suiteXmlFiles>testng.xml</suiteXmlFiles>
            <testFailureIgnore>false</testFailureIgnore>
        </configuration>
    </plugin>
```

Prueba de muestra para Allure Report

Crear una prueba de muestra con el nombre test.java

```
public class test{
    WebDriver driver;
    WebDriverWait wait;

    @BeforeMethod
    public void setup() {
        System.setProperty("webdriver.chrome.driver", "path to/chromedriver.exe");
        driver = new ChromeDriver();
        driver.get("https://www.google.com/");
        wait = new WebDriverWait(driver,50);
    }

    @Title("Title check")
    @Description("Checking the title of the loaded page.")
    @Test
    public void searchTest(){
        String title = driver.getTitle();
        LogUtil.log("Title Fetched: "+title);
        assertEquals(title,"Google");
        LogUtil.log("Test Passed. Expected: Google | Actual: "+title);
        System.out.println("Page Loaded");
    }

    @AfterMethod
    public void teardown(){
        driver.close();
    }
}
```

En la clase anterior hemos utilizado la clase LogUtiil. Esto se hace simplemente para registrar los **pasos** en nuestra prueba. A continuación se muestra el código para el mismo

LogUtil.java

```
public final class LogUtil {

    private LogUtil() {
    }
```

```
    @Step("{0}")
    public static void log(final String message){
        //intentionally empty
    }
}
```

aquí

@Title ("") agregará el título a su prueba en Allure Report

@Description ("") agregará la descripción a su prueba

@Paso ("") agregará un paso en el informe de atractivo para la prueba

Durante la ejecución, se generará un archivo xml en la carpeta "target / allure-results /"

Informe final con Jenkins

Si se está ejecutando en Jenkins con el complemento Allure Report instalado, Jenkins procesará automáticamente el informe en su trabajo.

Informe final sin Jenkins

Para aquellos que no tienen un Jenkins, use la siguiente línea de comandos para crear el informe html. Allure CLI es una aplicación Java, por lo que está disponible para todas las plataformas. Tiene que instalar manualmente Java 1.7+ antes de usar Allure CLI.

Debian

Para los repositorios basados en Debian proporcionamos un PPA para que la instalación sea sencilla: Instale Allure CLI para debian

```
$ sudo apt-add-repository ppa:yandex-qatools/allure-framework
$ sudo apt-get update
$ sudo apt-get install allure-commandline
```

Las distribuciones soportadas son: Trusty y Precise. Después de la instalación, tendrá un comando de encanto disponible.

Mac OS

Puede instalar Allure CLI a través de Homebrew.

```
$ brew tap qatools/formulas
$ brew install allure-commandline
```

Después de la instalación, tendrá un comando de encanto disponible.

Windows y otros Unix

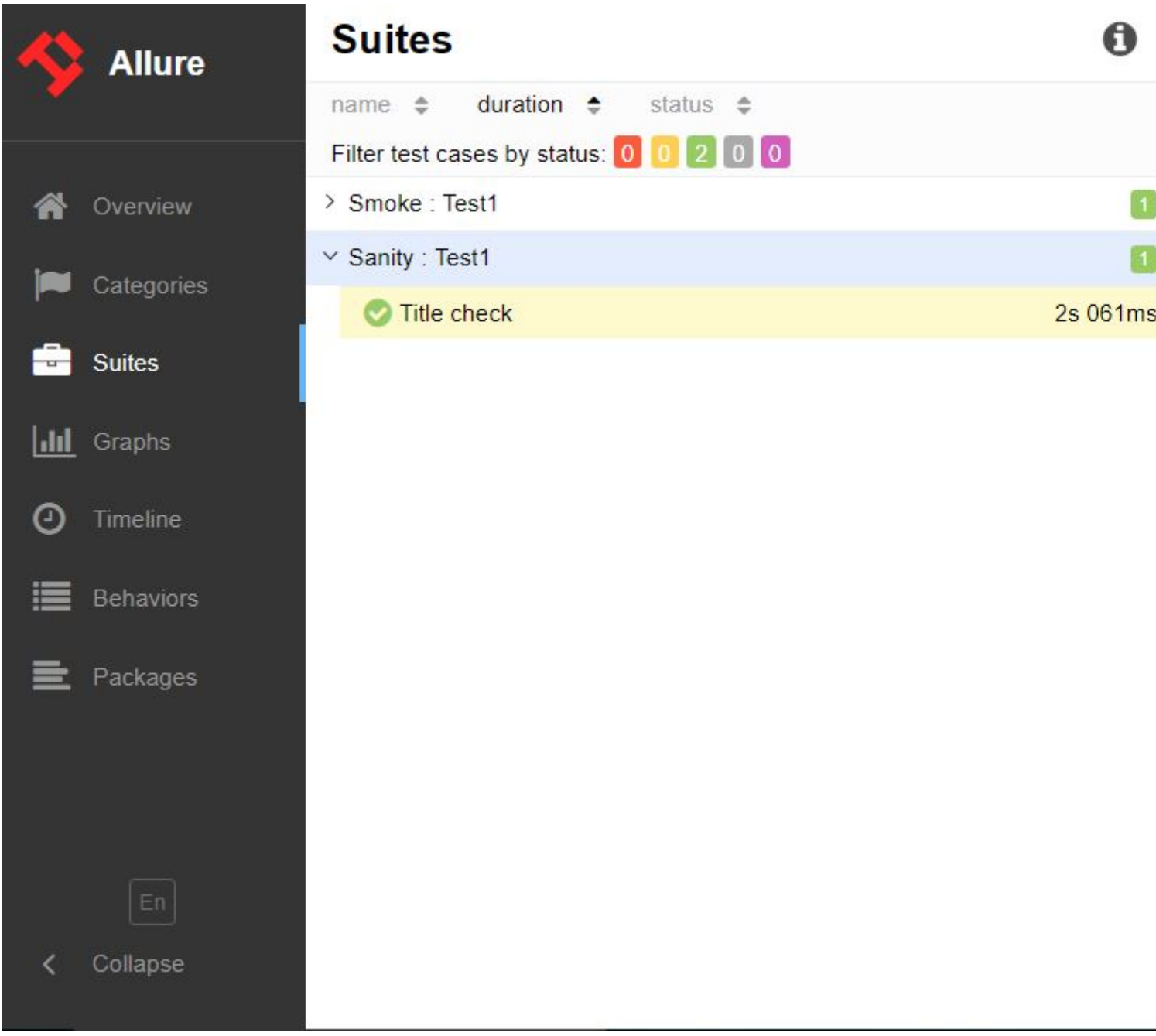
1. Descargue la versión más reciente como archivo comprimido zip desde

https://github.com/allure-framework/allure-core/releases/latest .

2. Descomprima el archivo en el directorio allure-commandline. Navegue al directorio bin.
3. Utilice allure.bat para Windows y allure para otras plataformas Unix.

En la línea de comandos / terminal ahora simplemente ingrese la siguiente sintaxis y el informe se generará en la carpeta de informes de encanto

```
$ allure generate directory-with-results/
```



Lea Informes HTML en línea: https://riptutorial.com/es/selenium-webdriver/topic/10721/informes-html

Capítulo 10: Interacción con elemento web

Examples

DO#

Borrar el contenido del elemento (generalmente cuadro de texto)

```
interactionWebElement.Clear();
```

Ingreso de datos al elemento (generalmente cuadro de texto)

```
interactionWebElement.SendKeys("Text");
```

Almacenando el valor del elemento.

```
string valueinTextBox = interactionWebElement.GetAttribute("value");
```

Almacenando texto de elemento.

```
string textOfElement = interactionWebElement.Text;
```

Haciendo clic en un elemento

```
interactionWebElement.Click();
```

Presentar un formulario

```
interactionWebElement.Submit();
```

Identificando la visibilidad de un elemento en la página.

```
bool isDisplayed=interactionWebElement.Displayed;
```

Identificando el estado de un elemento en la página.

```
bool isEnabled = interactionWebElement.Enabled;

bool isSelected=interactionWebElement.Selected;
```

Localizando elemento hijo de la interacción.

```
IWebElement childElement = interactionWebElement.FindElement(By.Id("childElementId"));
```

Localización de elementos hijo de la interacción.

```
Ilist<IWebElement> childElements =
interactionWebElement.FindElements(By.TagName("childElementsTagName"));
```

Java

Borrar el contenido de un elemento web: (nota: al simular las acciones de los usuarios en las pruebas, es mejor enviar retroceso, consulte la siguiente acción)

```
interactionWebElement.clear();
```

Ingresando datos - simulando el envío de pulsaciones:

```
interactionWebElement.sendKeys("Text");
interactionWebElement.sendKeys(Keys.CONTROL + "c"); // copy to clipboard.
```

Obteniendo el valor del atributo de un elemento:

```
interactionWebElement.getAttribute("value");
interactionWebElement.getAttribute("style");
```

Obteniendo el texto del elemento:

```
String elementsText = interactionWebElement.getText();
```

Seleccionando de desplegable:

```
Select dropDown = new Select(webElement);
dropDown.selectByValue(value);
```

Autoexplicativo:

```
interactionWebElement.click();
interactionWebElement.submit(); //for forms
interactionWebElement.isDisplayed();
interactionWebElement.isEnabled(); // for exampale - is clickable.
interactionWebElement.isSelected(); // for radio buttons.
```

Acciones usando `org.openqa.selenium.interactions.Actions` :

Arrastrar y soltar:

```
Action dragAndDrop = builder.clickAndHold(someElement)
   .moveToElement(otherElement)
   .release(otherElement)
   .build();

dragAndDrop.perform();
```

Seleccione múltiple:

```
Action selectMultiple = builder.keyDown(Keys.CONTROL)
   .click(someElement)
   .click(someOtherElement)
   .keyUp(Keys.CONTROL);

dragAndDrop.perform();
```

Auto explicativo (usando el constructor):

```
builder.doubleClick(webElement).perform();
builder.moveToElement(webElement).perform(); //hovering
```

Vea aquí para más ejemplos de acciones avanzadas y una lista completa.

Utilizando Javascript:

```
// Scroll to view element:
((JavascriptExecutor) driver).executeJavaScript("arguments[0].scrollIntoView(true);",
webElement);
```

Lea Interacción con elemento web en línea: https://riptutorial.com/es/selenium-webdriver/topic/4280/interaccion-con-elemento-web

Capítulo 11: Interactuando con la (s) ventana (s) del navegador

Examples

Gestionando la ventana activa

DO#

Maximizando la ventana

```
driver.Manage().Window.Maximize();
```

Esto es bastante sencillo, asegura que nuestra ventana activa actual se maximiza.

Posición de la ventana

```
driver.Manage().Window.Position = new System.Drawing.Point(1, 1);
```

Aquí esencialmente movemos la ventana actualmente activa a una nueva posición. En el objeto `Point` proporcionamos coordenadas `x` e `y` ; Estos se usan luego como compensaciones desde la esquina superior izquierda de la pantalla para determinar dónde se debe colocar la ventana. Tenga en cuenta que también puede almacenar la posición de la ventana en una variable:

```
System.Drawing.Point windowPosition = driver.Manage().Window.Position;
```

Tamaño de la ventana

La configuración y obtención del tamaño de la ventana utiliza la misma sintaxis que la posición:

```
driver.Manage().Window.Size = new System.Drawing.Size(100, 200);
System.Drawing.Size windowSize = driver.Manage().Window.Size;
```

URL de la ventana

Podemos obtener la URL actual de la ventana activa:

```
string url = driver.Url;
```

También podemos establecer la URL para la ventana activa, lo que hará que el controlador navegue al nuevo valor:

```
driver.Url = "http://stackoverflow.com/";
```

Manijas de ventana

Podemos obtener el manejador para la ventana actual:

```
string handle = driver.CurrentWindowHandle;
```

Y podemos obtener las asas para todas las ventanas abiertas:

```
IList<String> handles = driver.WindowHandles;
```

Pitón

Maximizando la ventana

```
driver.maximize_window()
```

Obtener la posición de la ventana

```
driver.get_window_position() # returns {'y', 'x'} coordinates
```

Establecer posición de la ventana

```
driver.set_window_position(x, y) # pass 'x' and 'y' coordinates as arguments
```

Obtener el tamaño de la ventana.

```
driver.get_window_size() # returns {'width', 'height'} values
```

Establecer tamaño de la ventana

```
driver.set_window_size(width, height) # pass 'width' and 'height' values as arguments
```

Título de la página actual

```
driver.title
```

URL actual

```
driver.current_url
```

Manijas de ventana

```
driver.current_window_handle
```

Lista de ventanas abiertas actualmente

```
driver.window_handles
```

Cerrar la ventana del navegador actual

Cambia a la nueva pestaña abierta. Cerrar las ventanas actuales (en este caso la nueva pestaña). Volver a la primera ventana.

TRANSPORTADOR:

```
browser.getAllWindowHandles().then(function (handles) {
    browser.driver.switchTo().window(handles[1]);
    browser.driver.close();
    browser.driver.switchTo().window(handles[0]);
});
```

JAVA Selenio:

```
        Set<String> handlesSet = driver.getWindowHandles();
        List<String> handlesList = new ArrayList<String>(handlesSet);
        driver.switchTo().window(handlesList.get(1));
        driver.close();
        driver.switchTo().window(handlesList.get(0));
```

Manejar múltiples ventanas

Pitón

Escenario más utilizado:

1. *abrir página en nueva ventana*
2. *cambiar a eso*
3. *hacer algo*
4. *cierralo*
5. *volver a la ventana principal*

```
# Open "Google" page in parent window
driver.get("https://google.com")

driver.title # 'Google'

# Get parent window
parent_window = driver.current_window_handle

# Open "Bing" page in child window
driver.execute_script("window.open('https://bing.com')")

# Get list of all windows currently opened (parent + child)
all_windows = driver.window_handles

# Get child window
child_window = [window for window in all_windows if window != parent_window][0]
```

```
# Switch to child window
driver.switch_to.window(child_window)

driver.title # 'Bing'

# Close child window
driver.close()

# Switch back to parent window
driver.switch_to.window(parent_window)

driver.title # 'Google'
```

Lea Interactuando con la (s) ventana (s) del navegador en línea: https://riptutorial.com/es/selenium-webdriver/topic/5181/interactuando-con-la--s--ventana--s--del-navegador

Capítulo 12: Localización de elementos web

Sintaxis

- ByChained (params By [] bys)

Observaciones

Los artículos se encuentran en Selenium mediante el uso de *localizadores* y la clase `By` . Para realizar un robusto proyecto de automatización con Selenium, se deben usar localizadores para elementos web de forma inteligente. Los localizadores deben ser **descriptivos, únicos y es poco probable que cambien,** por lo que no obtendrá falsos positivos en las pruebas, por ejemplo. La prioridad es utilizar:

1. **ID** : ya que es único y obtendrá exactamente el elemento que desea.
2. **Nombre de clase** : es descriptivo y puede ser único en un contexto dado.
3. **CSS** (mejor rendimiento que xpath): para selectores más complicados.
4. **XPATH** : donde no se puede usar CSS (XPATH Axis), por ejemplo, `div::parent` .

El resto de los localizadores son propensos a cambios o renderización, y preferiblemente deben evitarse.

Regla de oro: si su código no puede localizar un elemento en particular, una razón podría ser que su código no ha esperado a que se descarguen todos los elementos de DOM. Considere pedirle a su programa que "espere" por un corto período de tiempo (intente de 3 a 5 segundos y luego aumente lentamente según sea necesario) antes de buscar dicho elemento. Aquí hay un ejemplo en Python, tomado de esta pregunta :

```
from selenium import webdriver
import time

browser = webdriver.Firefox()
browser.get("https://app.website.com")

reports_element = browser.find_element_by_xpath("//button[contains(text(), 'Reports')]")

# Element not found! Try giving time for the browser to download all DOM elements:
time.sleep(10)

reports_element = browser.find_element_by_xpath("//button[contains(text(), 'Reports')]")
# This returns correct value!
```

Examples

Localización de elementos de página usando WebDriver

Para interactuar con WebElements en una página web, primero necesitamos identificar la

ubicación del elemento.

Por es la palabra clave disponible en selenio.

Puedes localizar los elementos por ...

1. **Por** *identificación*
2. **Por** *nombre de clase*
3. **Por nombre de** *etiqueta*
4. **Por** *nombre*
5. **Por** *enlace de texto*
6. **Por** *texto de enlace parcial*
7. **Por** *CSS Selector*
8. **Por** *XPath*
9. **Usando** *JavaScript*

Considere el siguiente ejemplo de script

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

En el código anterior, el nombre de usuario y la contraseña se configuran utilizando los ID. Ahora vas a identificar los elementos con id.

```
driver.findElement(By.id(username));

driver.findElement(By.id(password));
```

Como Selenio admite 7 idiomas diferentes, este documento le da una idea para ubicar los elementos en todos los idiomas.

Por identificación

Ejemplo de cómo encontrar un elemento usando ID:

```
<div id="coolestWidgetEvah">...</div>

Java       -  WebElement element = driver.findElement(By.id("coolestWidgetEvah"));
C#         -  IWebElement element = driver.FindElement(By.Id("coolestWidgetEvah"));
Python     -  element = driver.find_element_by_id("coolestWidgetEvah")
Ruby       -  element = driver.find_element(:id, "coolestWidgetEvah")
JavaScript/Protractor -  var elm = element(by.id("coolestWidgetEvah"));
```

Por nombre de clase

Ejemplo de cómo encontrar un elemento usando el nombre de clase:

```
<div class="cheese"><span>Cheddar</span></div>

Java       -  WebElement element = driver.findElement(By.className("cheese"));
C#         -  IWebElement element = driver.FindElement(By.ClassName("cheese"));
Python     -  element = driver.find_element_by_class_name("cheese")
Ruby       -  cheeses = driver.find_elements(:class, "cheese")
JavaScript/Protractor -  var elm = element(by.className("cheese"));
```

Por nombre de etiqueta

Ejemplo de cómo encontrar un elemento usando el nombre de etiqueta:

```
<iframe src="..."></iframe>

Java       -  WebElement element = driver.findElement(By.tagName("iframe"));
C#         -  IWebElement element = driver.FindElement(By.TagName("iframe"));
Python     -  element = driver.find_element_by_tag_name("iframe")
Ruby       -  frame = driver.find_element(:tag_name, "iframe")
JavaScript/Protractor -  var elm = element(by.tagName("iframe"));
```

Por nombre

Ejemplo de cómo encontrar un elemento usando nombre:

```
<input name="cheese" type="text"/>

Java       -  WebElement element = driver.findElement(By.name("cheese"));
C#         -  IWebElement element = driver.FindElement(By.Name("cheese"));
Python     -  element = driver.find_element_by_name("cheese")
Ruby       -  cheese = driver.find_element(:name, "cheese")
JavaScript/Protractor -  var elm = element(by.name("cheese"));
```

Por enlace de texto

Ejemplo de cómo encontrar un elemento usando el texto del enlace:

```
<a href="http://www.google.com/search?q=cheese">cheese</a>>

Java       -  WebElement element = driver.findElement(By.linkText("cheese"));
C#         -  IWebElement element = driver.FindElement(By.LinkText("cheese"));
Python     -  element = driver.find_element_by_link_text("cheese")
Ruby       -  cheese = driver.find_element(:link, "cheese")
JavaScript/Protractor -  var elm = element(by.linkText("cheese"));
```

Por texto de enlace parcial

Ejemplo de cómo encontrar un elemento utilizando texto de enlace parcial:

```
<a href="http://www.google.com/search?q=cheese">search for cheese</a>>

Java       -  WebElement element = driver.findElement(By.partialLinkText("cheese"));
C#         -  IWebElement element = driver.FindElement(By.PartialLinkText("cheese"));
Python     -  element = driver.find_element_by_partial_link_text("cheese")
Ruby       -  cheese = driver.find_element(:partial_link_text, "cheese")
JavaScript/Protractor -  var elm = element(by.partialLinkText("cheese"));
```

Por los selectores de CSS

Ejemplo de cómo encontrar un elemento utilizando los selectores de CSS:

```
<div id="food" class="dairy">milk</span>

Java       -  WebElement element = driver.findElement(By.cssSelector("#food.dairy")); //# is
used to indicate id and . is used for classname.
C#         -  IWebElement element = driver.FindElement(By.CssSelector("#food.dairy"));
Python     -  element = driver.find_element_by_css_selector("#food.dairy")
Ruby       -  cheese = driver.find_element(:css, "#food span.dairy.aged")
JavaScript/Protractor -  var elm = element(by.css("#food.dairy"));
```

Aquí hay un artículo sobre la creación de selectores de CSS:
http://www.w3schools.com/cssref/css_selectors.asp

Por XPath

Ejemplo de cómo encontrar un elemento usando XPath:

```
<input type="text" name="example" />

Java       -  WebElement element = driver.findElement(By.xpath("//input"));
C#         -  IWebElement element = driver.FindElement(By.XPath("//input"));
Python     -  element = driver.find_element_by_xpath("//input")
Ruby       -  inputs = driver.find_elements(:xpath, "//input")
JavaScript/Protractor -  var elm = element(by.xpath("//input"));
```

Aquí hay un artículo sobre XPath: http://www.w3schools.com/xsl/xpath_intro.asp

---

Usando JavaScript

Puede ejecutar un javascript arbitrario para encontrar un elemento y siempre que devuelva un elemento DOM, se convertirá automáticamente en un objeto WebElement.

Ejemplo simple en una página que ha cargado jQuery:

```
Java       -  WebElement element = (WebElement)
((JavascriptExecutor)driver).executeScript("return $('.cheese')[0]");

C#         -  IWebElement element = (IWebElement)
((IJavaScriptExecutor)driver).ExecuteScript("return $('.cheese')[0]");

Python     -  element = driver.execute_script("return $('.cheese')[0]");
Ruby       -  element = driver.execute_script("return $('.cheese')[0]")
JavaScript/Protractor -
```

Tenga en cuenta: este método no funcionará si su WebDriver en particular no es compatible con JavaScript, como SimpleBrowser .

Seleccionando por criterios múltiples [C #]

También es posible utilizar selectores juntos. Esto se hace usando el objeto `OpenQA.Selenium.Support.PageObjects.ByChained` :

```
element  = driver.FindElement(new ByChained(By.TagName("input"), By.ClassName("class"));
```

Cualquier número de `By` s se puede encadenar y se utiliza como una selección de tipo AND (es decir, todos los `By` s coinciden)

Seleccionar elementos antes de que la página se detenga.

Al llamar a `driver.Navigate().GoToUrl(url);` , la ejecución del código se detiene hasta que la página está completamente cargada. Esto a veces es innecesario cuando solo desea extraer datos.

Nota: los ejemplos de código a continuación podrían considerarse hacks. No hay una manera "oficial" de hacer esto.

Crear un nuevo hilo

Cree e inicie un hilo para cargar una página web, luego use Esperar .

DO#

```
using (var driver = new ChromeDriver())
{
    new Thread(() =>
    {
        driver.Navigate().GoToUrl("http://stackoverflow.com");
    }).Start();

    new WebDriverWait(driver, TimeSpan.FromSeconds(10))
      .Until(ExpectedConditions.ElementIsVisible(By.XPath("//div[@class='summary']/h3/a")));
}
```

Usar tiempos de espera

Usando un WebDriverTimeout, puede cargar una página, y después de un cierto período de tiempo, lanzará una excepción, lo que hará que la página deje de cargarse. En el bloque catch, puedes usar Wait .

DO#

```
using (var driver = new ChromeDriver())
{
    driver.Manage().Timeouts().SetPageLoadTimeout(TimeSpan.FromSeconds(5));

    try
    {
        driver.Navigate().GoToUrl("http://stackoverflow.com");
    }
    catch (WebDriverTimeoutException)
    {
        new WebDriverWait(driver, TimeSpan.FromSeconds(10))
          .Until(ExpectedConditions.ElementIsVisible
            (By.XPath("//div[@class='summary']/h3/a")));
    }
}
```

El problema : cuando configura el tiempo de espera demasiado corto, la página dejará de cargarse independientemente de si su elemento deseado está presente. Cuando configura el tiempo de espera demasiado tiempo, negará el beneficio de rendimiento.

Lea Localización de elementos web en línea: https://riptutorial.com/es/selenium-webdriver/topic/3991/localizacion-de-elementos-web

Capítulo 13: Manejar una alerta

Examples

Selenio con Java

Aquí es cómo manejar una alerta emergente en Java con Selenium:

Hay 3 tipos de ventanas emergentes.

1. **Alerta simple** : alerta ("Esta es una alerta simple");
2. **Alerta de confirmación** : var popuResult = confirm ("Confirmar ventana emergente con el botón Aceptar y Cancelar");
3. **Alerta rápida** : var person = prompt ("¿Te gusta stackoverflow?", "Sí / No");

Es hasta el usuario qué tipo de ventana emergente debe manejarse en su caso de prueba.

O puedes

1. aceptar () aceptar la alerta
2. despedir () para descartar la alerta
3. getText () Para obtener el texto de la alerta.
4. sendKeys () para escribir un texto a la alerta

Para simple alerta:

```
Alert simpleAlert = driver.switchTo().alert();
String alertText = simpleAlert.getText();
System.out.println("Alert text is " + alertText);
simpleAlert.accept();
```

Para la alerta de confirmación:

```
Alert confirmationAlert = driver.switchTo().alert();
String alertText = confirmationAlert.getText();
System.out.println("Alert text is " + alertText);
confirmationAlert.dismiss();
```

Para alerta rápida:

```
Alert promptAlert  = driver.switchTo().alert();
String alertText = promptAlert .getText();
System.out.println("Alert text is " + alertText);
//Send some text to the alert
promptAlert .sendKeys("Accepting the alert");
Thread.sleep(4000); //This sleep is not necessary, just for demonstration
promptAlert .accept();
```

de acuerdo a sus necesidades.

Otra forma en que puedes hacer esto es envolver tu código dentro de un try-catch:

```
try{
   // Your logic here.
} catch(UnhandledAlertException e){
  Alert alert = driver.switchTo().alert();
  alert.accept();
}
// Continue.
```

DO#

Aquí se explica cómo cerrar una alerta emergente en C # con Selenium:

```
IAlert alert = driver.SwitchTo().Alert();
// Prints text and closes alert
System.out.println(alert.Text);
alert.Accept();
or
alert.Dismiss();
```

de acuerdo a sus necesidades.

Otra forma en que puedes hacer esto es envolver tu código dentro de un try-catch:

```
try{
   // Your logic here.
} catch(UnhandledAlertException e){
  var alert = driver.SwitchTo().Alert();
  alert.Accept();
}
// Continue.
```

Pitón

Hay varias formas de cambiar a alerta emergente en `Python` :

1. *En desuso*

```
alert = driver.switch_to_alert()
```

2. *Utilizando* `switch_to` :

```
alert = driver.switch_to.alert
```

3. *Usando* `ExplicitWait` :

```
 from selenium.webdriver.common.by import By
 from selenium.webdriver.support.ui import WebDriverWait
```

```
from selenium.webdriver.support import expected_conditions as EC

alert = WebDriverWait(driver, TIMEOUT_IN_SECONDS).until(EC.alert_is_present())
```

4. *Al declarar la instancia de la clase de* `Alert` :

```
from selenium.webdriver.common.alert import Alert

alert = Alert(driver)
```

Para rellenar el campo de entrada en una `prompt()` emergente activada por la `prompt() JavaScript prompt()` :

```
alert.send_keys('Some text to send')
```

Para confirmar el diálogo emergente *:

```
alert.accept()
```

Despedir:

```
alert.dismiss()
```

Para obtener texto de la ventana emergente:

```
alert.text
```

* *PS* `alert.dismiss()` *podría usarse para confirmar ventanas emergentes activadas por* `JavaScript alert()` *así como* `alert.confirm()`

Lea Manejar una alerta en línea: https://riptutorial.com/es/selenium-webdriver/topic/6048/manejar-una-alerta

Capítulo 14: Manejo de errores en la automatización usando Selenium

Examples

Pitón

`WebDriverException` es una excepción base de `Selenium-WebDriver` que se puede usar para capturar todas las demás excepciones de `Selenium-WebDriver`

Para poder capturar una excepción, primero se debe importar:

```
from selenium.common.exceptions import WebDriverException as WDE
```

y entonces:

```
try:
    element = driver.find_element_by_id('ID')
except WDE:
    print("Not able to find element")
```

De la misma manera puedes importar otras excepciones más específicas:

```
from selenium.common.exceptions import ElementNotVisibleException
from selenium.common.exceptions import NoAlertPresentException
...
```

Si desea extraer solo el mensaje de excepción:

```
from selenium.common.exceptions import UnexpectedAlertPresentException

try:
    driver.find_element_by_tag_name('a').click()
except UnexpectedAlertPresentException as e:
    print(e.__dict__["msg"])
```

Lea Manejo de errores en la automatización usando Selenium en línea: https://riptutorial.com/es/selenium-webdriver/topic/9548/manejo-de-errores-en-la-automatizacion-usando-selenium

Capítulo 15: Marcos de conmutación

Sintaxis

- **Java**
- driver.switchTo (). frame (nombre de la cadena);
- driver.switchTo (). frame (String id);
- driver.switchTo (). frame (int index);
- driver.switchTo (). frame (WebElement frameElement);
- driver.switchTo (). defaultContent ();
- **DO#**
- driver.SwitchTo (). Frame (int frameIndex);
- driver.SwitchTo (). Frame (IWebElement frameElement);
- driver.SwitchTo (). Frame (string frameName);
- driver.SwitchTo (). DefaultContent ();
- **Pitón**
- driver.switch_to_frame (nameOrId)
- driver.switch_to.frame (nameOrId)
- driver.switch_to_frame (índice)
- driver.switch_to.frame (índice)
- driver.switch_to_frame (frameElement)
- driver.switch_to.frame (frameElement)
- driver.switch_to_default_content ()
- driver.switch_to.default_content ()
- **JavaScript**
- driver.switchTo (). frame (nameOrId)
- driver.switchTo (). frame (índice)
- driver.switchTo (). defaultContent ()

Parámetros

parámetro	detalles
nameOrId	Seleccione un cuadro por su nombre de identificación.
índice	Seleccione un cuadro por su índice de base cero.
marco elemento	Seleccione un marco utilizando su WebElement previamente ubicado

Examples

Para cambiar a un marco utilizando Java

Para una instancia, si el código fuente html de una vista o elemento html está envuelto por un iframe como este:

```
<iframe src="../images/eightball.gif" name="imgboxName" id="imgboxId">
   <p>iframes example</p>
   <a href="../images/redball.gif" target="imgbox">Red Ball</a>
</iframe><br />
```

... luego, para realizar cualquier acción en los elementos web del iframe, primero debe cambiar el enfoque al iframe, utilizando cualquiera de los siguientes métodos:

Uso de Id. De cuadro (solo debe usarse si conoce el id del iframe).

```
driver.switchTo().frame("imgboxId"); //imgboxId - Id of the frame
```

Usando el nombre del marco (debe usarse solo si conoce el nombre del iframe).

```
driver.switchTo().frame("imgboxName"); //imgboxName - Name of the frame
```

Uso del índice de cuadros (debe usarse solo si no tiene el ID o el nombre del iframe), donde el índice define la posición del iframe entre todos los cuadros.

```
driver.switchTo().frame(0); //0 - Index of the frame
```

Nota: Si tiene tres cuadros en la página, entonces el primer cuadro estará en el índice 0, el segundo en el índice 1 y el tercero en el índice 2.

Usar el elemento web ubicado anteriormente (debe usarse solo si ya ha localizado el marco y lo ha devuelto como un elemento `WebElement`).

```
driver.switchTo().frame(frameElement); //frameElement - webelement that is the frame
```

Entonces, para hacer clic en el ancla de la `Red Ball` :

```
driver.switchTo().frame("imgboxId");
driver.findElement(By.linkText("Red Ball")).Click();
```

Salir de un frame utilizando Java.

Para cambiar el enfoque al documento principal o al primer fotograma de la página. Tienes que usar la siguiente sintaxis.

```
driver.switchTo().defaultContent();
```

Cambia a un marco usando C

1. Cambie a un cuadro por índice.

Aquí estamos cambiando al índice 1. El índice se refiere al orden de los marcos en la página. Esto debería usarse como último recurso, ya que la identificación del marco o los nombres son mucho más confiables.

```
driver.SwitchTo().Frame(1);
```

2. Cambiar a un cuadro por nombre

```
driver.SwitchTo().Frame("Name_Of_Frame");
```

3. Cambie a un marco por Título, ID u otros al pasar IWebElement

Si desea cambiar a un cuadro por ID o título, debe pasar un elemento web como parámetro:

```
driver.SwitchTo().Frame(driver.FindElement(By.Id("ID_OF_FRAME")));
driver.SwitchTo().Frame(driver.FindElement(By.CssSelector("iframe[title='Title_of_Frame']")));
```

También tenga en cuenta que si su cuadro tarda unos segundos en aparecer, es posible que deba esperar :

```
new WebDriverWait(driver, TimeSpan.FromSeconds(10))
    .Until(ExpectedConditions.ElementIsVisible(By.Id("Id_Of_Frame")));
```

Sal de un cuadro:

```
driver.SwitchTo().DefaultContent()
```

Salir de un cuadro usando C

Para cambiar el enfoque al documento principal o al primer fotograma de la página. Tienes que usar la siguiente sintaxis.

webDriver.SwitchTo (). DefaultContent ();

Cambiar entre marcos secundarios de un marco principal.

Considera que tienes un marco padre (Frame-Parent). y 2 marcos secundarios (Frame_Son, Frame_Daughter). Veamos varias condiciones y cómo manejarlas.

1. De padre a hijo o hija:

```
driver.switchTo().frame("Frame_Son");
driver.switchTo().frame("Frame_Daughter");
```

2. Desde el hijo al padre: si el marco principal es el predeterminado, cambie al marco predeterminado, o bien, del marco predeterminado al marco principal. Pero no puedes cambiar directamente de hijo a padre.

```
driver.switchTo().defaultContent();
driver.switchTo().frame("Frame_Parent");
```

3. De hijo a hija: si tu hermana comete un error, no le grites, simplemente comunícate con tu padre. De manera similar, le das control al marco principal y luego al marco secundario.

```
driver.switchTo().defaultContent();
driver.switchTo().frame("Frame_Parent");
driver.switchTo().frame("Frame_Daughter");
```

Espera a que se carguen tus marcos

En algunos casos, es posible que su marco no se muestre de inmediato y probablemente tenga que esperar hasta que se cargue para cambiar. O de lo contrario tendrá NoSuchFrameException.

Así que siempre es una buena opción esperar antes de cambiar. La siguiente es una forma ideal de esperar hasta que se carga un marco.

```
try{
        new WebDriverWait(driver, 300).ignoring(StaleElementReferenceException.class).
                        ignoring(WebDriverException.class).

until(ExpectedConditions.visibilityOf((driver.findElement(By.id("cpmInteractionDivFrame"))));}
    catch{
```

// lanza la excepción solo si su marco no es visible en su tiempo de espera 300 segundos}

Lea Marcos de conmutación en línea: https://riptutorial.com/es/selenium-webdriver/topic/4589/marcos-de-conmutacion

Capítulo 16: Modelo de objetos de página

Introducción

Un papel importante en la automatización de sitios web y aplicaciones web implica identificar elementos en la pantalla e interactuar con ellos. Los artículos se encuentran en Selenium mediante el uso de localizadores y la clase `By` . Estos localizadores e interacciones se colocan dentro de los objetos de página como una mejor práctica para evitar duplicar el código y facilitar el mantenimiento. Encapsula `WebElements` y se supone que contiene información de comportamiento y devolución en la página (o parte de una página en una aplicación web).

Observaciones

El modelo de objetos de página es un patrón en el que escribimos clases orientadas a objetos que sirven como interfaz para una vista particular de la página web. Usamos los métodos de esa clase de página para realizar la acción requerida. Hace unos años, estábamos manipulando el código HTML de la página web en las clases de prueba directamente, lo cual era muy difícil de mantener junto con los cambios frágiles en la interfaz de usuario.

Sin embargo, tener su código organizado de una forma de Patrón de Objeto de Página proporciona una API específica para la aplicación, lo que le permite manipular los elementos de la página sin excavar alrededor del HTML. La Rue of Thumb básica dice, su objeto de página debe tener todo lo que un humano puede hacer en esa página web. Por ejemplo, para acceder al campo de texto en una página web, debe utilizar un método para obtener el texto y devolver la cadena después de realizar todas las modificaciones.

Algunos puntos importantes que debe tener en cuenta al diseñar los objetos de página:

1. Por lo general, el objeto de página no debe compilarse solo para las páginas, pero usted prefiere compilarlo para elementos significativos de la página. Por ejemplo, una página con varias pestañas para mostrar diferentes gráficos de sus académicos debe tener el mismo número de páginas que el recuento de pestañas.

2. La navegación de una vista a otra debe devolver la instancia de clases de página.

3. Los métodos de utilidad que deben estar allí solo para una vista o página web específica deben pertenecer solo a esa clase de página.

4. Los métodos de aserción no deben ser atendidos por clases de página, puede tener métodos para devolver booleanos pero no verificarlos allí. Por ejemplo, para verificar el nombre completo del usuario, puede tener un método para obtener un valor booleano:

```
public boolean hasDisplayedUserFullName (String userFullName) {
    return driver.findElement(By.xpath("xpathExpressionUsingFullName")).isDisplayed();
}
```

5. Si su página web está basada en iframe, prefiera tener clases de página para iframes también.

Ventajas del patrón de objeto de página:

1. Separación limpia entre el código de prueba y el código de página
2. En caso de cualquier cambio en la interfaz de usuario de la página web, no es necesario cambiar el código en varios lugares. Cambiar solo en clases de pagina.
3. No hay localizadores de elementos dispersos.
4. Hace que el código sea más fácil de entender
5. Facil mantenimiento

Examples

Introducción (Utilizando Java)

Un ejemplo para realizar una prueba de inicio de sesión basada en el patrón de objeto de página:

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;

/**
 * Class which models the view of Sign-In page
 */
public class SignInPage {

    @FindBy(id="username")
    private usernameInput;

    @FindBy(id="password")
    private passwordInput;

    @FindBy(id="signin")
    private signInButton;

    private WebDriver driver;

    public SignInPage(WebDriver driver) {
        this.driver = driver;
    }

    /**
     * Method to perform login
     */
    public HomePage performLogin(String username, String password) {
        usernameInput.sendKeys(username);
        passwordInput.sendKeys(password);
        signInButton.click();
        return PageFactory.initElements(driver, HomePage.class);
    }
}
```

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;
/**
 * Class which models the view of home page
 */
public class HomePage {
    @FindBy(id="logout")
    private logoutLink;

    private WebDriver driver;

    public HomePage(WebDriver driver) {
        this.driver = driver;
    }

    /**
     * Method to log out
     */
    public SignInPage logout() {
        logoutLink.click();
        wait.ForPageToLoad();
        return PageFactory.initElements(driver, SignInPage.class);
    }
}

/**
 * Login test class
 */
public class LoginTest {
    public void testLogin() {
        SignInPage signInPage = new SignInPage(driver);
        HomePage homePage = signInPage.login(username, password);
        signInPage = homePage.logout();
    }
}
```

DO#

Los objetos de página deben contener comportamiento, información de retorno para las aserciones y posiblemente un método para el método de estado de preparación de página en la inicialización. Selenium soporta objetos de página usando anotaciones. En C # es como sigue:

```
using OpenQA.Selenium;
using OpenQA.Selenium.Support.PageObjects;
using OpenQA.Selenium.Support.UI;
using System;
using System.Collections.Generic;

public class WikipediaHomePage
{
    private IWebDriver driver;
    private int timeout = 10;
    private By pageLoadedElement = By.ClassName("central-featured-logo");

    [FindsBy(How = How.Id, Using = "searchInput")]
    [CacheLookup]
```

```
    private IWebElement searchInput;

    [FindsBy(How = How.CssSelector, Using = ".pure-button.pure-button-primary-progressive")]
    [CacheLookup]
    private IWebElement searchButton;

    public ResultsPage Search(string query)
    {
        searchInput.SendKeys(query);
        searchButton.Click();
    }

    public WikipediaHomePage VerifyPageLoaded()
    {
        new WebDriverWait(driver, TimeSpan.FromSeconds(timeout)).Until<bool>((drv) =>  return
drv.ExpectedConditions.ElementExists(pageLoadedElement));

        return this;
    }
}
```

notas:

- `CacheLookup` guarda el elemento en el caché y guarda devolver un nuevo elemento cada llamada. Esto mejora el rendimiento pero no es bueno para elementos que cambian dinámicamente.
- `searchButton` tiene 2 nombres de clase y no ID, por eso no puedo usar el nombre o la identificación de la clase.
- Verifiqué que mis localizadores me devolverán el elemento que deseo usando las Herramientas de desarrollador (para Chrome), en otros navegadores puedes usar FireBug o similar.
- `Search()` método `Search()` devuelve otro objeto de página (`ResultsPage`) cuando el clic de búsqueda lo redirecciona a otra página.

Página de Buenas Prácticas Modelo de Objeto

- Cree archivos separados para el encabezado y el pie de página (ya que son comunes para todas las páginas y no tiene sentido hacer que formen parte de una sola página)
- Mantenga los elementos comunes (como Buscar / Atrás / Siguiente, etc.) en un archivo separado (la idea es eliminar cualquier tipo de duplicación y mantener la segregación lógica)
- Para Driver, es una buena idea crear una clase de Driver separada y mantener el Driver como estático para que se pueda acceder a él en todas las páginas. (Tengo todas mis páginas web extendidas DriverClass)
- Las funciones utilizadas en PageObjects se dividen en la parte más pequeña posible teniendo en cuenta la frecuencia y la forma en que se llamarán (la forma en que lo hizo para iniciar sesión, aunque el inicio de sesión se puede dividir en enterUsername y enterPassword, pero aún así se mantiene ya que la función de inicio de sesión es más lógica porque en la mayoría de los casos, la función de inicio de sesión se llamará en lugar de llamadas separadas para ingresar las funciones de nombre de usuario y contraseña.
- El uso de PageObjects en sí mismo separa el script de prueba de los elementLocators
- Tener funciones de utilidad en carpetas utils separadas (como DateUtil, excelUtils, etc.)

- Tener configuraciones en una carpeta conf separada (como establecer el entorno en el que se deben ejecutar las pruebas, configurar las carpetas de salida y entrada)
- Incorporar screenCapture en caso de fallo.
- Tenga una variable de espera estática en DriverClass con un tiempo de espera implícito como lo ha hecho Siempre intente tener esperas condicionales en lugar de esperas estáticas como: wait.until (ExpectedConditions). Esto asegura que la espera no ralentiza la ejecución innecesariamente.

Lea Modelo de objetos de página en línea: https://riptutorial.com/es/selenium-webdriver/topic/4853/modelo-de-objetos-de-pagina

Capítulo 17: Navega entre múltiples cuadros

Introducción

En las páginas web que contienen una cantidad de cuadros, Selenen considera que el marco es una ventana por lo que el acceso al contenido presente en el marco necesita cambiar al marco. Muchas veces necesitamos una estructura web donde tenemos marcos con marcos para navegar dentro de las ventanas de marcos. Selenium proporciona el método swithTo ().

Examples

Ejemplo de marco

```
<iframe "id="iframe_Login1">

    <iframe "id="iframe_Login2">

        <iframe "id="iframe_Login3">

        </iframe>

    </iframe>

</iframe>
```

Para cambiar al marco en selenio, utilice los métodos swithTo () y frame ().

driver.switchTo (). frame (iframe_Login1); driver.switchTo (). frame (iframe_Login2); driver.switchTo (). frame (iframe_Login3);

Para volver a cambiar, podemos usar parentFrame () y defaultContest ();

parentFrame (): cambia el foco al contexto padre. Si el contexto actual es el contexto de navegación de nivel superior, el contexto permanece sin cambios.

```
driver.switchTo().parentFrame();
```

defaultContent (): selecciona el primer fotograma de la página o el documento principal cuando una página contiene iframes.

```
 driver.switchTo().defaultContent();
```

Lea Navega entre múltiples cuadros en línea: https://riptutorial.com/es/selenium-webdriver/topic/9803/navega-entre-multiples-cuadros

Capítulo 18: Navegación

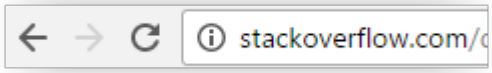
Sintaxis

- **DO#**
- void Back ()
- void Forward ()
- void GotToUrl (url de cadena)
- void Actualizar ()
- **Pitón**
- driver.back ()
- driver.forward ()
- driver.get ("URL")
- driver.refresh ()
- **Java**
- driver.navigate (). back ();
- driver.navigate (). forward ();
- driver.navigate (). to ("URL");
- driver.navigate (). refresh ();

Examples

Navegar () [C #]

Es posible navegar directamente por el navegador, como usar los comandos estándar de la barra de herramientas disponibles en todos los navegadores:



Puede crear un objeto de navegación llamando a `Navigate()` en el controlador:

```
IWebDriver driver
INavigation navigation = driver.Navigate();
```

Un objeto de navegación le permite realizar numerosas acciones que navegan por el navegador en la web:

```
//like pressing the back button
navigation.Back();
//like pressing the forward button on a browser
navigation.Forward();
//navigate to a new url in the current window
navigation.GoToUrl("www.stackoverflow.com");
//Like pressing the reload button
navigation.Refresh();
```

Navegar () [Java]

Para navegar a cualquier url:

```
driver.navigate().to("http://www.example.com");
```

Para mover hacia atrás:

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

Para avanzar hacia adelante:

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

Para actualizar la página:

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

Métodos del navegador en WebDriver

WebDriver, la interfaz principal que se utiliza para las pruebas, que representa un navegador web idealizado. Los métodos en esta clase se dividen en tres categorías:

- Control del propio navegador.
- Selección de elementos web
- Ayudas de depuración

Los métodos clave son get (String), que se usa para cargar una nueva página web, y los diversos métodos similares a findElement (By), que se usa para encontrar WebElements. En este post vamos a aprender los métodos de control del navegador. obtener

```
void get(java.lang.String url)
```

Cargue una nueva página web en la ventana actual del navegador. Esto se hace usando una operación HTTP GET, y el método se bloqueará hasta que se complete la carga. es mejor esperar hasta que termine este tiempo de espera, ya que si la página subyacente cambia mientras su prueba está ejecutando los resultados de las llamadas futuras en esta interfaz, será contra la página recién cargada. **Uso**

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

 //setting timeout for page load
 driver.manage().timeouts().pageLoadTimeout(20, TimeUnit.SECONDS);

 //Call Url in get method
 driver.get("https://www.google.com");
 //or
 driver.get("https://seleniumhq.org");
```

getCurrentUrl

```
java.lang.String getCurrentUrl()
```

Obtenga una cadena que representa la URL actual que el navegador está mirando. Devuelve la URL de la página actualmente cargada en el navegador.

Uso

```
//Getting current url loaded in browser & comparing with expected url
 String pageURL = driver.getCurrentUrl();
 Assert.assertEquals(pageURL, "https://www.google.com");
```

GetTitle

```
java.lang.String getTitle()
```

Devuelve el título de la página actual, con espacios en blanco iniciales y finales eliminados, o nulo si aún no está configurado.

Uso

```
//Getting current page title loaded in browser & comparing with expected title
 String pageTitle = driver.getTitle();
 Assert.assertEquals(pageTitle, "Google");

getPageSource

java.lang.String getPageSource()
```

Obtener la fuente de la última página cargada. Si la página se ha modificado después de cargarla (por ejemplo, por Javascript) no hay garantía de que el texto devuelto sea el de la página modificada.

Uso

```
//get the current page source
 String pageSource = driver.getPageSource();
```

cerrar

```
void close()
```

Cierre la ventana actual, cerrando el navegador si es la última ventana abierta actualmente. Si hay más de una ventana abierta con esa instancia de controlador, este método cerrará la ventana que tiene el foco actual en ella.

Uso

```
//Close the current window
     driver.close();
```

dejar

```
void quit()
```

Sale de este controlador, cerrando cada ventana asociada. Después de llamar a este método, no podemos usar ningún otro método utilizando la misma instancia de controlador.

Uso

```
//Quit the current driver session / close all windows associated with driver
     driver.quit();
```

Todos estos son métodos muy útiles disponibles en Selenium 2.0 para controlar el navegador según sea necesario.

Lea Navegación en línea: https://riptutorial.com/es/selenium-webdriver/topic/7272/navegacion

Capítulo 19: Navegadores sin cabeza

Examples

PhantomJS [C #]

`PhantomJS` es un navegador web sin cabeza con todas las funciones **con** el soporte de JavaScript.

Antes de comenzar, deberá descargar un controlador PhantomJS y asegurarse de poner esto al principio de su código:

```
using OpenQA.Selenium;
using OpenQA.Selenium.PhantomJS;
```

Genial, ahora en la inicialización:

```
var driver = new PhantomJSDriver();
```

Esto simplemente creará una nueva instancia de la clase PhantomJSDriver. Luego puede usarlo de la misma manera que todos los WebDriver, como:

```
using (var driver = new PhantomJSDriver())
{
    driver.Navigate().GoToUrl("http://stackoverflow.com/");

    var questions = driver.FindElements(By.ClassName("question-hyperlink"));

    foreach (var question in questions)
    {
        // This will display every question header on StackOverflow homepage.
        Console.WriteLine(question.Text);
    }
}
```

Esto funciona bien. Sin embargo, el problema que probablemente encontró es que, cuando trabaja con la interfaz de usuario, `PhantomJS` abre una nueva ventana de consola, que en la mayoría de los casos no es realmente necesaria. Afortunadamente, podemos ocultar la ventana e incluso mejorar ligeramente el rendimiento utilizando `PhantomJSOptions` y `PhantomJSDriverService` . Ejemplo de trabajo completo a continuación:

```
// Options are used for setting "browser capabilities", such as setting a User-Agent
// property as shown below:
var options = new PhantomJSOptions();
options.AddAdditionalCapability("phantomjs.page.settings.userAgent",
"Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:25.0) Gecko/20100101 Firefox/25.0");

// Services are used for setting up the WebDriver to your likings, such as
// hiding the console window and restricting image loading as shown below:
var service = PhantomJSDriverService.CreateDefaultService();
service.HideCommandPromptWindow = true;
```

```
service.LoadImages = false;

// The same code as in the example above:
using (var driver = new PhantomJSDriver(service, options))
{
    driver.Navigate().GoToUrl("http://stackoverflow.com/");

    var questions = driver.FindElements(By.ClassName("question-hyperlink"));

    foreach (var question in questions)
    {
        // This will display every question header on StackOverflow homepage.
        Console.WriteLine(question.Text);
    }
}
```

`PhantomJSDriverService` profesional: haga clic en una clase (por ejemplo, `PhantomJSDriverService`) y presione F12 para ver exactamente lo que contienen junto con una breve descripción de lo que hacen.

SimpleBrowser [C #]

`SimpleBrowser` es un WebDriver ligero **sin** soporte de JavaScript.

Es considerablemente más rápido que un `PhantomJS` mencionado `PhantomJS` , sin embargo, cuando se trata de funcionalidad, se limita a tareas simples sin funciones sofisticadas.

En primer lugar, deberá descargar el paquete SimpleBrowser.WebDriver y luego colocar este código al principio:

```
using OpenQA.Selenium;
using SimpleBrowser.WebDriver;
```

Ahora, aquí hay un breve ejemplo de cómo usarlo:

```
using (var driver = new SimpleBrowserDriver())
{
    driver.Navigate().GoToUrl("http://stackoverflow.com/");

    var questions = driver.FindElements(By.ClassName("question-hyperlink"));

    foreach (var question in questions)
    {
        // This will display every question header on StackOverflow homepage.
        Console.WriteLine(question.Text);
    }
}
```

Navegador sin cabeza en Java

HTMLUnitDriver

HTMLUnitDriver es la implementación más liviana de un navegador sin cabeza (sin GUI) para

Webdriver basado en HtmlUnit. Modela documentos HTML y proporciona una API que le permite invocar páginas, completar formularios, hacer clic en enlaces, etc., tal como lo hace en su navegador normal. Es compatible con JavaScript y funciona con las bibliotecas AJAX. Se utiliza para probar y recuperar datos del sitio web.

Ejemplo: uso de HTMLUnitDriver para obtener una lista de preguntas de `http://stackoverflow.com/` .

```
import java.util.List;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.htmlunit.HtmlUnitDriver;

class testHeadlessDriver{
            private void getQuestions() {
                    WebDriver driver = new HtmlUnitDriver();
                    driver.get("http://stackoverflow.com/");
                    driver.manage().timeouts().implicitlyWait(60, TimeUnit.SECONDS);
                    List<WebElement> questions = driver.findElements(By.className("question-
hyperlink"));
                    questions.forEach((question) -> {
                        System.out.println(question.getText());
                    });
                   driver.close();
                }
    }
```

Es igual que cualquier otro navegador (Mozilla Firefox, Google Chrome, IE), pero no tiene GUI, la ejecución no está visible en la pantalla.

Lea Navegadores sin cabeza en línea: https://riptutorial.com/es/selenium-webdriver/topic/3931/navegadores-sin-cabeza

Capítulo 20: Oyentes

Examples

JUIT

Si está utilizando JUnit para ejecutar, puede extender la clase `TestWatcher` :

```
public class TestRules extends TestWatcher {

    @Override
    protected void failed(Throwable e, Description description) {
        // This will be called whenever a test fails.
    }
```

Así que en tu clase de prueba simplemente puedes llamarlo:

```
public class testClass{

@Rule
public TestRules testRules = new TestRules();

@Test
public void doTestSomething() throws Exception{
    // If the test fails for any reason, it will be caught be testrules.
}
```

EventFiringWebDriver

Utilizando el EventFiringWebDriver . Puede adjuntar WebDriverEventListener y anular los métodos, es decir, el método onException:

```
EventFiringWebDriver driver = new EventFiringWebDriver(new FirefoxDriver());
WebDriverEventListener listener = new AbstractWebDriverEventListener() {
    @Override
    public void onException(Throwable t, WebDriver driver) {
        // Take action
    }
};
driver.register(listener);
```

Lea Oyentes en línea: https://riptutorial.com/es/selenium-webdriver/topic/8226/oyentes

Capítulo 21: Programa básico de Selenium Webdriver

Introducción

Este tema tiene como objetivo mostrar el programa de controlador web básico en selenium lenguajes compatibles como C #, Groovy, Java, Perl, PHP, Python y Ruby.

Journey incluye abrir el controlador del navegador -> Página de Google -> apagar el navegador

Examples

DO#

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;

namespace BasicWebdriver
{
    class WebDriverTest
    {
        static void Main()
        {
            using (var driver = new ChromeDriver())
            {
                driver.Navigate().GoToUrl("http://www.google.com");
            }
        }
    }
}
```

El 'programa' anterior navegará a la página de inicio de Google, y luego cerrará el navegador después de cargar completamente la página.

```
using (var driver = new ChromeDriver())
```

Esto crea una instancia de un nuevo objeto WebDriver utilizando la interfaz `IWebdriver` y crea una nueva instancia de la ventana del navegador. En este ejemplo, estamos usando `ChromeDriver` (aunque esto podría ser reemplazado por el controlador apropiado para cualquier navegador que `ChromeDriver` usar). Estamos envolviendo esto con una declaración `using` , porque `IWebDriver` implementa `IDisposable` , por lo que no es necesario que escriba explícitamente en `driver.Quit();` .

En caso de que no haya descargado su WebDriver usando NuGet , entonces deberá pasar un argumento en forma de ruta al directorio donde se encuentra el controlador "chromedriver.exe".

Navegando

```
driver.Navigate().GoToUrl("http://www.google.com");
```

y

```
driver.Url = "http://www.google.com";
```

Ambas líneas hacen lo mismo. Le indican al conductor que navegue a una URL específica y que espere hasta que la página se cargue antes de pasar a la siguiente declaración.

Existen otros métodos relacionados con la navegación, como `Back()` , `Forward()` o `Refresh()` .

Después de eso, el bloque de `using` se cierra de forma segura y elimina el objeto.

Pitón

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

def set_up_driver():
    path_to_chrome_driver = 'chromedriver'
    return webdriver.Chrome(executable_path=path_to_chrome_driver)

def get_google():
    driver = set_up_driver()
    driver.get('http://www.google.com')
    tear_down(driver)

def tear_down(driver):
    driver.quit()

if '__main__' == __name__:
    get_google()
```

El 'programa' anterior navegará a la página de inicio de Google y luego cerrará el navegador antes de completar.

```
if '__main__' == __name__:
    get_google()
```

Primero tenemos nuestra función principal, nuestro punto de entrada en el programa, que llama a `get_google()` .

```
def get_google():
    driver = set_up_driver()
```

`get_google()` luego comienza creando nuestra instancia de `driver` través de `set_up_driver()` :

```
def set_up_driver():
    path_to_chrome_driver = 'chromedriver'
    return webdriver.Chrome(executable_path=path_to_chrome_driver)
```

Por medio del cual indicamos dónde se encuentra `chromedriver.exe` , y `chromedriver.exe` instancia de nuestro objeto controlador con esta ruta. El resto de `get_google()` navega a Google:

```
driver.get('http://www.google.com')
```

Y luego llama a `tear_down()` pasando el objeto controlador:

```
tear_down(driver)
```

`tear_down()` simplemente contiene una línea para cerrar nuestro objeto controlador:

```
driver.quit()
```

Esto le indica al controlador que cierre todas las ventanas abiertas del navegador y elimine el objeto del navegador, ya que después de esta llamada no tenemos ningún otro código, esto termina el programa de manera efectiva.

Java

El código de abajo es todo sobre 3 pasos.

1. Abriendo un navegador chrome
2. Apertura de la página de google
3. Apagar el navegador

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

public class WebDriverTest {
    public static void main(String args[]) {
        System.setProperty("webdriver.chrome.driver", "C:\\path\\to\\chromedriver.exe");
        WebDriver driver = new ChromeDriver();

        driver.get("http://www.google.com");
        driver.quit();
    }
}
```

El 'programa' anterior navegará a la página de inicio de Google y luego cerrará el navegador antes de completar.

```
System.setProperty("webdriver.chrome.driver", "C:\\path\\to\\chromedriver.exe");
WebDriver driver = new ChromeDriver();
```

La primera línea le dice al sistema dónde encontrar el `ChromeDriver` (chromedriver.exe). Luego creamos nuestro objeto controlador llamando al constructor `ChromeDriver()` , nuevamente

podríamos estar llamando a nuestro constructor aquí para cualquier navegador / plataforma.

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

Esto le dice a nuestro conductor que navegue a la url especificada: http://www.google.com . La API de Java WebDriver proporciona el método `get()` directamente en la interfaz de WebDriver, aunque se pueden encontrar más métodos de navegación a través del método de `navigate()` , por ejemplo, `driver.navigate.back()` .

Una vez que la página ha terminado de cargar, llamamos inmediatamente:

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

Esto le indica al controlador que cierre todas las ventanas abiertas del navegador y deseche el objeto del controlador, ya que después de esta llamada no tenemos ningún otro código, esto termina el programa de manera efectiva.

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

Es una instrucción (que no se muestra aquí) del controlador para cerrar solo la ventana activa; en este caso, como solo tenemos una ventana, las instrucciones causarán resultados idénticos a la llamada a `quit()` .

Java - Mejores prácticas con clases de página

Caso de uso: iniciar sesión en la cuenta FB

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

public class FaceBookLoginTest {
    private static WebDriver driver;
    HomePage homePage;
    LoginPage loginPage;
    @BeforeClass
    public void openFBPage(){
        driver = new FirefoxDriver();
        driver.get("https://www.facebook.com/");
        loginPage = new LoginPage(driver);
    }
    @Test
    public void loginToFB(){
       loginPage.enterUserName("username");
       loginPage.enterPassword("password");
       homePage = loginPage.clickLogin();
        System.out.println(homePage.getUserName());
    }

}
```

Clases de página: Página de inicio y Página de inicio Clase de página de inicio:

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;

public class LoginPage {

    WebDriver driver;
    public LoginPage(WebDriver driver){
        this.driver = driver;
    }
    @FindBy(id="email")
    private WebElement loginTextBox;

    @FindBy(id="pass")
    private WebElement passwordTextBox;

    @FindBy(xpath = ".//input[@data-testid='royal_login_button']")
    private WebElement loginBtn;

    public void enterUserName(String userName){
        if(loginTextBox.isDisplayed()) {
            loginTextBox.clear();
            loginTextBox.sendKeys(userName);
        }
        else{
            System.out.println("Element is not loaded");
        }
    }
    public void enterPassword(String password){
        if(passwordTextBox.isDisplayed()) {
            passwordTextBox.clear();
            passwordTextBox.sendKeys(password);
        }
        else{
            System.out.println("Element is not loaded");
        }
    }
    public HomePage clickLogin(){
        if(loginBtn.isDisplayed()) {
            loginBtn.click();
        }
        return new HomePage(driver);
    }
}
```

Página de inicio de la clase:

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;

public class HomePage {

    WebDriver driver;
    public HomePage(WebDriver driver){
        this.driver = driver;
    }
```

```
    @FindBy(xpath=".//a[@data-testid='blue_bar_profile_link']/span")
    private WebElement userName;

    public String getUserName(){
        if(userName.isDisplayed()) {
            return userName.getText();
        }
        else {
            return "Username is not present";
        }
    }

}
```

Lea Programa básico de Selenium Webdriver en línea: https://riptutorial.com/es/selenium-webdriver/topic/3990/programa-basico-de-selenium-webdriver

Capítulo 22: Rejilla de selenio

Examples

Configuración del nodo

La configuración de Selenium Grid Node reside en el propio Node y contiene la información sobre la configuración de red y las capacidades de Node. La configuración se puede aplicar de varias maneras:

- Configuración predeterminada
- Configuración JSON
- Configuración de línea de comandos

Configuración JSON

La configuración del nodo en el archivo JSON se divide en 2 secciones:

- Capacidades
- Configuración

Las capacidades definen áreas como qué tipos y versiones de navegador son compatibles, ubicaciones de binarios del navegador, número de instancias máximas de cada tipo de navegador.

La configuración se ocupa de los ajustes, como las direcciones de hub y nodo y los puertos.

A continuación se muestra un ejemplo de un archivo de configuración JSON:

```
{
  "capabilities": [
    {
      "browserName": "firefox",
      "acceptSslCerts": true,
      "javascriptEnabled": true,
      "takesScreenshot": false,
      "firefox_profile": "",
      "browser-version": "27",
      "platform": "WINDOWS",
      "maxInstances": 5,
      "firefox_binary": "",
      "cleanSession": true
    },
    {
      "browserName": "chrome",
      "maxInstances": 5,
      "platform": "WINDOWS",
      "webdriver.chrome.driver": "C:/Program Files (x86)/Google/Chrome/Application/chrome.exe"
    },
    {
      "browserName": "internet explorer",
      "maxInstances": 1,
```

```
      "platform": "WINDOWS",
      "webdriver.ie.driver": "C:/Program Files (x86)/Internet Explorer/iexplore.exe"
    }
  ],
  "configuration": {
    "_comment" : "Configuration for Node",
    "cleanUpCycle": 2000,
    "timeout": 30000,
    "proxy": "org.openqa.grid.selenium.proxy.WebDriverRemoteProxy",
    "port": 5555,
    "host": ip,
    "register": true,
    "hubPort": 4444,
    "maxSessions": 5
  }
}
```

Cómo crear un nodo

Para crear un nodo, primero necesita tener un concentrador. Si no tienes un hub, puedes crearlo así:

```
java -jar selenium-server-standalone-<version>.jar -role hub
```

Entonces puedes crear un nodo:

```
java -jar selenium-server-standalone-<version>.jar -role node -hub
http://localhost:4444/grid/register // default port is 4444
```

Más información aquí: https://github.com/SeleniumHQ/selenium/wiki/Grid2

Lea Rejilla de selenio en línea: https://riptutorial.com/es/selenium-webdriver/topic/1359/rejilla-de-selenio

Capítulo 23: Robot En Selenium

Sintaxis

- retraso (int ms)
- keyPress (código clave int)
- keyRelease (código clave int)
- mouseMove (int x, int y)
- mousepress (botones int)
- mouseRelease (botones int)
- MouseWheel (int wheelAmt)

Parámetros

Parámetro	Detalles
Sra	Tiempo para dormir en milisegundos
clave	Constante para presionar la tecla especificada, por ejemplo, para presionar `A` código `A` es `VK_A` . Consulte para obtener más detalles: https://docs.oracle.com/javase/7/docs/api/java/awt/event/KeyEvent.html
x, y	Coordenadas de pantalla
botones	La máscara de botón; una combinación de una o más máscaras de botón del mouse
wheelAmt	Número de muescas para mover la rueda del mouse, valor negativo para moverse hacia arriba / hacia afuera del valor positivo del usuario para moverse hacia abajo / hacia el usuario

Observaciones

Esta sección contiene detalles sobre la implementación de Robot API con Selenium Webdriver. La clase Robot se usa para generar entradas nativas del sistema cuando el selenio no es capaz de hacerlo, por ejemplo, al presionar la tecla derecha del mouse, presionar la tecla F1, etc.

Examples

Evento de pulsación de tecla utilizando Robot API (JAVA)

```
import java.awt.AWTException;
import java.awt.Robot;
import java.awt.event.KeyEvent;
```

```
public class KeyBoardExample {
    public static void main(String[] args) {
        try {
            Robot robot = new Robot();
            robot.delay(3000);
            robot.keyPress(KeyEvent.VK_Q); //VK_Q for Q
        } catch (AWTException e) {
            e.printStackTrace();
        }
    }
}
```

Con selenio

A veces necesitamos presionar cualquier tecla para probar el evento de presionar una tecla en la aplicación web. Para una instancia para probar la tecla ENTER en el formulario de inicio de sesión, podemos escribir algo como abajo con Selenium WebDriver

```
import java.awt.AWTException;
import java.awt.Robot;
import java.awt.event.KeyEvent;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.annotations.Test;

public class LoginTest {

    @Test
    public void testEnterKey() throws InterruptedException
    {
        WebDriver driver=new FirefoxDriver();
        Robot robot=null;
        driver.get("test-url");
        driver.manage().window().maximize();
        driver.findElement(By.xpath("xpath-expression")).click();
        driver.findElement(By.xpath("xpath-expression")).sendKeys("username");
        driver.findElement(By.xpath("xpath-expression")).sendKeys("password");
        try {
            robot=new Robot();
        } catch (AWTException e) {
            e.printStackTrace();
        }
        //Keyboard Activity Using Robot Class
        robot.keyPress(KeyEvent.VK_ENTER);
    }
}
```

Evento de ratón utilizando Robot API (JAVA)

Movimiento del ratón:

```
import java.awt.Robot;

public class MouseClass {
 public static void main(String[] args) throws Exception {
```

```
     Robot robot = new Robot();

     // SET THE MOUSE X Y POSITION
     robot.mouseMove(300, 550);
     }
}
```

Presione el botón izquierdo / derecho del ratón:

```
import java.awt.Robot;
import java.awt.event.InputEvent;

public class MouseEvent {
 public static void main(String[] args) throws Exception {
     Robot robot = new Robot();

     // LEFT CLICK
     robot.mousePress(InputEvent.BUTTON1_MASK);
     robot.mouseRelease(InputEvent.BUTTON1_MASK);

     // RIGHT CLICK
     robot.mousePress(InputEvent.BUTTON3_MASK);
     robot.mouseRelease(InputEvent.BUTTON3_MASK);
     }
}
```

Haga clic y desplace la rueda:

```
import java.awt.Robot;
import java.awt.event.InputEvent;

public class MouseClass {
 public static void main(String[] args) throws Exception {
     Robot robot = new Robot();

     // MIDDLE WHEEL CLICK
     robot.mousePress(InputEvent.BUTTON3_DOWN_MASK);
     robot.mouseRelease(InputEvent.BUTTON3_DOWN_MASK);

     // SCROLL THE MOUSE WHEEL
     robot.mouseWheel(-100);
     }
}
```

Lea Robot En Selenium en línea: https://riptutorial.com/es/selenium-webdriver/topic/4877/robot-en-selenium

Capítulo 24: Seleccionar clase

Sintaxis

- **Java**
- deseleccionar todo ()
- deselectByIndex (índice int)
- deselectByValue (java.lang.String value)
- deselectByVisibleText (java.lang.String text)
- getAllSelectedOptions ()
- getFirstSelectedOption ()
- getOptions ()
- isMultiple ()
- selectByIndex (índice int)
- selectByValue (java.lang.String value)
- selectByVisibleText (java.lang.String text)

Parámetros

Parámetros	Detalles
índice	La opción en este índice será seleccionada.
valor	El valor para emparejar contra
texto	El texto visible para hacer coincidir contra

Observaciones

`Select` clase de `Select` de Selenium WebDriver proporciona métodos útiles para interactuar con las opciones de `select` . El usuario puede realizar operaciones en un menú desplegable de selección y también deseleccionar la operación utilizando los métodos a continuación.

En **C #** la clase Select es en realidad `SelectElement`

Examples

Diferentes maneras de seleccionar de la lista desplegable

A continuación se muestra una página HTML.

```
<html>
<head>
<title>Select Example by Index value</title>
```

```
</head>
<body>
<select name="Travel"><option value="0" selected> Please select</option>
<option value="1">Car</option>
<option value="2">Bike</option>
<option value="3">Cycle</option>
<option value="4">Walk</option>
</select>
</body>
</html>
```

JAVA

Seleccionar por índice

Seleccionar la opción por índice utilizando Java.

```
public class selectByIndexExample {
    WebDriver driver;
    @Test
    public void selectSamples()
    {
        driver = new FirefoxDriver();
        driver.get("URL GOES HERE");
        WebElement element=driver.findElement(By.name("Travel")); //This is the 'Select'
element locator
        Select sel=new Select(element);
        sel.selectByIndex(1); //This will select the first 'Option' from 'Select' List i.e.
Car
    }
}
```

Seleccionar por valor

```
public class selectByValueExample {
    WebDriver driver;
    @Test
    public void selectSamples()
    {
        driver = new FirefoxDriver();
        driver.get("URL GOES HERE");
        WebElement element=driver.findElement(By.name("Travel")); //This is the 'Select'
element locator
        Select sel=new Select(element);
        sel.selectByValue("Bike"); //This will select the 'Option' from 'Select' List which
has value as "Bike".
        //NOTE: This will be case sensitive
    }
}
```

Seleccionar por texto de visibilidad

```
public class selectByVisibilityTextExample {
    WebDriver driver;
    @Test
    public void selectSamples()
    {
        driver = new FirefoxDriver();
        driver.get("URL GOES HERE");
        WebElement element=driver.findElement(By.name("Travel")); //This is the 'Select'
element locator
        Select sel=new Select(element);
        sel.selectByVisibleText("Cycle"); //This will select the 'Option' from 'Select' List
who's visibility text is "Cycle".
        //NOTE: This will be case sensitive
    }
}
```

DO#

Todos los ejemplos a continuación se basan en la interfaz genérica de `IWebDriver`

Seleccionar por índice

```
IWebElement element=driver.FindElement(By.name("Travel"));
SelectElement selectElement = new SelectElement(title);
selectElement.SelectByIndex(0);
```

Seleccionar por valor

```
IWebElement element=driver.FindElement(By.name("Travel"));
SelectElement selectElement = new SelectElement(title);
selectElement.SelectByIndex("1");
//NOTE: This will be case sensitive
```

Seleccionar por texto

```
IWebElement element=driver.FindElement(By.name("Travel"));
SelectElement selectElement = new SelectElement(title);
selectElement.SelectByText("Walk");
```

Lea Seleccionar clase en línea: https://riptutorial.com/es/selenium-webdriver/topic/6426/seleccionar-clase

Capítulo 25: Selenium e2e setup

Introducción

Este tema cubre la configuración de extremo a extremo de Selenium, es decir, Selenium Webdriver + TestNG + Maven + Jenkins.

Para la adición de informes, consulte el tema Informes HTML

Examples

Configuración de TestNG

TestNG es su marco de prueba actualizado para Junit. Vamos a utilizar **testng.xml** para invocar suites de prueba. Esto es útil cuando vamos a usar CI por delante.

testng.xml

En la carpeta raíz de su proyecto, cree un archivo xml con el nombre testng.xml. Tenga en cuenta que el nombre también puede ser diferente, pero por conveniencia se usa como "prueba" en todas partes.

A continuación se muestra el código simple para el archivo testng.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Smoke"> //name of the suite
    <test name="Test1"> //name of the test
        <classes>
            <class name="test.SearchTest">
            <methods>
                <include name="searchTest"/>
            </methods>
            </class>
        </classes>
    </test>
</suite>
```

Maven Setup

TBD. Cómo configurar pom.xml para llamar a testng.xml

Instalación de Jenkins

TBD. Cubrirá la configuración de Jenkins para extraer código de git / bitbucket, etc.

Lea Selenium e2e setup en línea: https://riptutorial.com/es/selenium-

webdriver/topic/10724/selenium-e2e-setup

Capítulo 26: Selenium-webdriver con Python, Ruby y Javascript junto con la herramienta CI

Introducción

Esta es una forma de realizar pruebas de selenio con CircleCI.

Examples

Integración de CircleCI con Selenium Python y Unittest2

Circle.yml

```
machine:
  python:
    # Python version to use - Selenium requires python 3.0 and above
    version: pypy-3.6.0
dependencies:
    pre:
        # Install pip packages
        - pip install selenium
        - pip install unittest
test:
  override:
    # Bash command to run main.py
    - python main.py
```

main.py

```
import unittest2

# Load and run all tests in testsuite matching regex provided
loader = unittest2.TestLoader()
# Finds all the tests in the same directory that have a filename that ends in test.py
testcases = loader.discover('.', pattern="*test.py")
test_runner = unittest2.runner.TextTestRunner()
# Checks that all tests ran
success = test_runner.run(testcases).wasSuccessful()
```

example_test.py

```
class example_test(unittest.TestCase):
    def test_something(self):
        # Make a new webdriver instance
        self.driver = webdriver.Chrome()
        # Goes to www.goooogle.com
        self.driver.get("https://www.google.com")
```

Lea Selenium-webdriver con Python, Ruby y Javascript junto con la herramienta CI en línea:

https://riptutorial.com/es/selenium-webdriver/topic/10091/selenium-webdriver-con-python--ruby-y-javascript-junto-con-la-herramienta-ci

Capítulo 27: Tomando capturas de pantalla

Introducción

Tomando capturas de pantalla y guardando en un camino particular

Sintaxis

- Archivo src = ((TakesScreenshot) driver) .getScreenshotAs (OutputType.FILE);
- FileUtils.copyFile (src, nuevo archivo ("D: \ screenshot.png"));

Examples

JAVA

Código para tomar y guardar captura de pantalla:

```
public class Sample
{
    public static void main (String[] args)
    {
        *//Initialize Browser*
        System.setProperty("webdriver.gecko.driver", "**E:\\path\\to\\geckodriver.exe**");
        WebDriver driver = new FirefoxDriver();
        driver.manage().window().maximize();
        driver.get("https://www.google.com/");

        //Take Screesnshot
        File src = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
        try {
            //Save Screenshot in destination file
            FileUtils.copyFile(src, new File("D:\\screenshot.png"));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Toma la captura de pantalla:

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

Guarda la captura de pantalla desde el origen al destino:

```
FileUtils.copyFile(src, new File("D:\\screenshot.png"));
```

Lea Tomando capturas de pantalla en línea: https://riptutorial.com/es/selenium-webdriver/topic/10094/tomando-capturas-de-pantalla

Capítulo 28: Usando @FindBy anotaciones en Java

Sintaxis

- CLASS_NAME: @FindBy (className = "classname")
- CSS: @FindBy (css = "css")
- ID: @FindBy (id = "id")
- ID_OR_NAME: @FindBy (how = How.ID_OR_NAME, utilizando = "idOrName")
- LINK_TEXT: @FindBy (linkText = "texto")
- NOMBRE: @FindBy (nombre = "nombre")
- PARTIAL_LINK_TEXT: @FindBy (partialLinkText = "texto")
- TAG_NAME: @FindBy (tagName = "tagname")
- XPATH: @FindBy (xpath = "xpath")

Observaciones

Tenga en cuenta que hay dos formas de utilizar la anotación. Ejemplos:

```
@FindBy(id = "id")
```

y

```
@FindBy(how = How.ID, using  ="id")
```

Son iguales y ambos buscan elemento por su ID. En el caso de `ID_OR_NAME` solo puedes usar

```
@FindBy(how = How.ID_OR_NAME, using ="idOrName")
```

`PageFactory.initElements()` debe usar después de la `PageFactory.initElements()` instancias del objeto de la página para encontrar elementos marcados con la anotación `@FindBy` .

Examples

Ejemplo basico

Supongamos que estamos utilizando el modelo de objeto Page . Clase de objeto de página:

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;
```

```
public class LoginPage {

    @FindBy(id="loginInput") //method used to find WebElement, in that case Id
    WebElement loginTextbox;

    @FindBy(id="passwordInput")
    WebElement passwordTextBox;

    //xpath example:
    @FindBy(xpath="//form[@id='loginForm']/button(contains(., 'Login')")
    WebElement loginButton;

    public void login(String username, String password){
        // login method prepared according to Page Object Pattern
        loginTextbox.sendKeys(username);
        passwordTextBox.sendKeys(password);
        loginButton.click();
        // because WebElements were declared with @FindBy, we can use them without
        // driver.find() method
    }
}
```

Y pruebas de clase:

```
class Tests{
    public static void main(String[] args) {
        WebDriver driver = new FirefoxDriver();
        LoginPage loginPage = new LoginPage();

        //PageFactory is used to find elements with @FindBy specified
        PageFactory.initElements(driver, loginPage);
        loginPage.login("user", "pass");
    }
}
```

Hay algunos métodos para encontrar WebElements usando @FindBy - ver la sección de Sintaxis.

Lea Usando @FindBy anotaciones en Java en línea: https://riptutorial.com/es/selenium-webdriver/topic/6777/usando--findby-anotaciones-en-java

Capítulo 29: Usando Selenium Webdriver con Java

Introducción

Selenium webdriver es un marco de automatización web que le permite probar su aplicación web en diferentes navegadores web. A diferencia de Selenium IDE, webdriver le permite desarrollar sus propios casos de prueba en el lenguaje de programación de su elección. Es compatible con Java, .Net, PHP, Python, Perl, Ruby.

Examples

Abriendo la ventana del navegador con una URL específica usando Selenium Webdriver en Java

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

class test_webdriver{
    public static void main(String[] args) {
       WebDriver driver = new FirefoxDriver();
       driver.get("http://stackoverflow.com/");
       driver.close();
    }
}
```

Abriendo una ventana del navegador con el método to ()

Abrir un navegador con el método a ().

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
class navigateWithTo{
    public static void main(String[] args) {
       WebDriver driver = new FirefoxDriver();
       driver.navigate().to("http://www.example.com");
       driver.close();
    }
}
```

Lea Usando Selenium Webdriver con Java en línea: https://riptutorial.com/es/selenium-webdriver/topic/9158/usando-selenium-webdriver-con-java

Creditos

S. No	Capítulos	Contributors
1	Empezando con Selenium-webdriver	Abhilash Gupta, Alice, Community, Eugene S, iamdanchiv, Jakub Lokša, Josh, Kishor, Michal, Mohit Tater, Pan, Priyanshu Shekhar, rg702, Santoshsarma, Tomislav Nakic-Alfirevic, vikingben
2	Acciones (Emulando gestos de usuario complejos)	Josh, Kenil Fadia, Liam, Priyanshu Shekhar, Tom Mc
3	Configuración / Obtención del tamaño de la ventana del navegador	Abhilash Gupta
4	Configuración de cuadrícula de selenio	mnoronha, Prasanna Selvaraj, selva, Thomas
5	De desplazamiento	Andersson, Sagar007
6	Ejecución de Javascript en la página.	Brydenr, Liam
7	Espere	Jakub Lokša, Josh, Kenil Fadia, Liam, Moshisho, noor, Sajal Singh, Saurav
8	Excepciones en Selenium-WebDriver	Brydenr
9	Informes HTML	Ashish Deshmukh
10	Interacción con elemento web	Jakub Lokša, Liam, Moshisho, Siva, Sudha Velan
11	Interactuando con la (s) ventana (s) del navegador	Andersson, Josh, Sakshi Singla
12	Localización de elementos web	alecxe, daOnlyBG, Jakub Lokša, Josh, Liam, Łukasz Piaszczyk, Moshisho, NarendraR, noor, Priya, Sakshi Singla, Siva

13	Manejar una alerta	Andersson, Aurasphere, Priya, SlightlyKosumi
14	Manejo de errores en la automatización usando Selenium	Andersson
15	Marcos de conmutación	Andersson, dreamwork801, Jakub Lokša, Java_deep, Jim Ashworth, Karthik Taduvai, Kyle Fairns, Liam, lloyd, Priyanshu Shekhar, SlightlyKosumi
16	Modelo de objetos de página	JeffC, Josh, Moshisho, Priyanshu Shekhar, Sakshi Singla
17	Navega entre múltiples cuadros	Pavan T, Raghvendra
18	Navegación	Andersson, Liam, Santoshsarma, viralpatel
19	Navegadores sin cabeza	Abhilash Gupta, Jakub Lokša, Liam, r_D, Tomislav Nakic-Alfirevic
20	Oyentes	Erki M.
21	Programa básico de Selenium Webdriver	Jakub Lokša, Josh, Liam, Priya, Sudha Velan, Thomas, vikingben
22	Rejilla de selenio	Eugene S, Y-B Cause
23	Robot En Selenium	Priyanshu Shekhar
24	Seleccionar clase	Gaurav Lad, Liam
25	Selenium e2e setup	Ashish Deshmukh
26	Selenium-webdriver con Python, Ruby y Javascript junto con la herramienta CI	Brydenr
27	Tomando capturas de pantalla	Abhilash Gupta, Sanchit
28	Usando @FindBy anotaciones en Java	Alex Wittig, Łukasz Piaszczyk
29	Usando Selenium Webdriver con Java	r_D, the_coder