Selenium Notes

Locators

------------

web element within the web page such as ID, Name, Class Name, Link Text, Partial Link Text,Tag Name and XPATH

1. How to initialize driver in selenium

--------------------------------------

System.setProperty("webdriver.chrome.driver","./drivers/chromedriver.exe")

WebDriver driver = new ChromeDriver()

System.setProperty("webdriver.gecko.driver","C:\\geckodriver.exe")

WebDriver driver = new FirefoxDriver();

----

Example,

package newproject;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

//comment the above line and uncomment below line to use Chrome

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

public class PG1 {

public static void main(String[] args) {

// declaration and instantiation of objects/variables

System.setProperty("webdriver.gecko.driver","C:\\geckodriver.exe");

WebDriver driver = new FirefoxDriver();

//comment the above 2 lines and uncomment below 2 lines to use Chrome

//System.setProperty("webdriver.chrome.driver","G:\\chromedriver.exe");

//WebDriver driver = new ChromeDriver();

String baseUrl = "http://demo.guru99.com/test/newtours/";

String expectedTitle = "Welcome: Mercury Tours";

String actualTitle = "";

// launch Fire fox and direct it to the Base URL

driver.get(baseUrl);

// get the actual value of the title

actualTitle = driver.getTitle();

/\*

\* compare the actual title of the page with the expected one and print

\* the result as "Passed" or "Failed"

\*/

if (actualTitle.contentEquals(expectedTitle)){

System.out.println("Test Passed!");

} else {

System.out.println("Test Failed");

}

//close Fire fox

driver.close();

}

}

2. GUI Commands

-----------------

1. To get tag name of the element.

tagName = driver.findElement(By.id("email")).getTagName();

2. Instead of using the long "driver.findElement(By.locator())" syntax every time you will access a particular element, we can instantiate a WebElement object for it.

WebElement myElement=driver.findelement(By.Id("username");

myElement.sendKeys("Rama);

3. Get commands

get()

getTitle()

getPageSource()

getCurrentUrl()

getText()

4. Navigate commands

navigate().to() --->It does exactly the same thing as the get() method.

navigate().refresh()

navigate().back()

navigate().forward() -->takes you forward by one page on the browser's history.

5. Closing commands

close()

quit()

3. Handling Frame in WebPage

---------------------------

consider if we have 3 frames named frame1,frame2,frame3. and we need to access element in frame3 .

First we should identofy the framenane and then we can idnetify the element using switchTo.frame() method.

ex,

driver.switchTo().frame("classFrame");

driver.findElement(By.linkText("Deprecated")).click();

4. Handling pop up in WebPage

-----------------------------

we must use the "switchTo().alert()" method. In the code below, we will use this method to access the alert box and then retrieve its message using the "getText()" method, and then automatically close the alert box using the "switchTo().alert().accept()" method.

ex,

String alertMessage = "";

driver.findElement(By.cssSelector("input[value=\"Go!\"]")).click();

alertMessage = driver.switchTo().alert().getText();

driver.switchTo().alert().accept();

System.out.println(alertMessage);

5. Waits

----------

Implicit wait - used to set the default waiting time throughout the program

Explicit wait - used to set the waiting time for a particular instance only

Implicit Wait,

ex

driver.manage.timeouts().implicitliWait(10,TimeUnit.SECONDS);

Explicit Wait,

Explicit waits are done using the WebDriverWait and ExpectedCondition classes. For the following example, we shall wait up to 10 seconds for an element whose id is "username" to become visible before proceeding to the next command. Here are the steps.

ex WebDriverWait ,

Webdriver driver = new ChromeDriver();

WebDriverWait wait=new WebDriverWait(driver,10); //wait 10 seconds until chome loads

ex ExpectedCondition ,

wait.until(ExpectedConditions.VisibilityOfElementsLocated(By.Id("username")));

driver.findElement(By.Id("username")).sendKeys("Rama");

6.Conditional loops Note --->always use NoSuchElementFoundException in try catch block

---------------------

isEnabled() is used when you want to verify whether a certain element is enabled or not before executing a command

ex,

WebElement user = driver.findElement(By.Id("username"));

if(user.isEnabled())

user.sendKeys("rama);

isDisabled() is used when you want to verify whether a certain element is displayed or not before executing a command.

do{something here}

while(driver.findElement(By.Id("username").isDisabled());

isDisplayed() is used when you want to verify whether a certain element is displayed or not before executing a command.

do{something here}

while(driver.findElement(By.Id("username").isDisabled());

isSelected() is used when you want to verify whether a certain check box, radio button, or option in a drop-down box is selected. It does not work on other elements.

if(driver.findElement(By.Id("username").isSelected())

driver.findElement(By.Id("Rama").click()

7. ExpectedConditions --->always use TimeOutException in try catch block

The ExpectedConditions class offers a wider set of conditions that you can use in conjunction with WebDriverWait's until() method.

alertIsPresent() - waits until an alert box is displayed.

if (wait.until(ExpectedConditions.alertIsPresent())!=null)

{

Sysoum.out.println("Alert is present")

elementToBeClickable() - Waits until an element is visible and, at the same time, enabled

The sample code below will wait until the element with id="username" to become visible and enabled first before assigning that element as a WebElement variable named "txtUserName".

WebElememt txtname=wait.until(ExpectedConditions.elementToBeClickable(By.Id("username")))

-------------------

Example of NoSuchElementFoundException and TimeOutException

NoSuchElementFoundException

--------------------------

WebElement txt\_username=driver.findElement(By.Id("username"))

try

if(txt\_username.isSelected())

{

txt\_username.sendKeys("Rama");

}

catch(NoSuchElementFoundException nef)

{

System.out.println(nef.toString());

}

TimeOutException

-----------------

WebDriverWait wait=new WebDriverWait(driver, 5)

try

{

wait.until(ExpectedConditions.visibilityOfElementLocated(By.Id("username")))

driver.findElement(By.Id("username")).sendKeys("Rama");

}

catch(TimeOutException toe)

{

System.out.println(toe.toStirng());

}

------------------

Summary

To start using the WebDriver API, you must import at least these two packages.

org.openqa.selenium.\*

org.openqa.selenium.firefox.FirefoxDriver

The get() method is the equivalent of Selenium IDE's "open" command.

Locating elements in WebDriver is done by using the findElement() method.

The following are the available options for locating elements in WebDriver:

By.className

By.cssSelector

By.id

By.linkText

By.name

By.partialLinkText

By.tagName

By.xpath

The By.cssSelector() does not support the "contains" feature.

You can instantiate an element using the WebElement class.

Clicking on an element is done by using the click() method.

WebDriver provides these useful get commands:

get()

getTitle()

getPageSource()

getCurrentUrl()

getText()

WebDriver provides these useful navigation commands

navigate().forward()

navigate().back()

navigate().to()

navigate().refresh()

The close() and quit() methods are used to close browser windows. Close() is used to close a single window; while quit() is used to close all windows associated to the parent window that the WebDriver object was controlling.

The switchTo().frame() and switchTo().alert() methods are used to direct WebDriver's focus onto a frame or alert, respectively.

Implicit waits are used to set the waiting time throughout the program, while explicit waits are used only on specific portions.

You can use the isEnabled(), isDisplayed(),isSelected(), and a combination of WebDriverWait and ExpectedConditions methods when verifying the state of an element. However, they do not verify if the element does not exists.

When isEnabled(), isDisplayed(),or isSelected() was called while the element was not existing, WebDriver will throw a NoSuchElementException.

When WebDriverWait and ExpectedConditions methods were called while the element was not existing, WebDriver would throw a TimeoutException.

Note:

driver.get() : It's used to go to the particular website , But it doesn't maintain the browser History and cookies so , we can't use forward and backward button , if we click on that , page will not get schedule

driver.navigate() : it's used to go to the particular website , but it maintains the browser history and cookies, so we can use forward and backward button to navigate between the pages during the coding of Testcase

8. FindElement and FindElements difference

Findelement is use to find an unique element in a webpage using certain propertise such as ID or name whereas findElements is used to find the multiple elements in a web page.

please note, To clear the values in Textbox. use. txt\_usernane.clear();

ex,

List<WebElement> listofElements = dirver.findElements(By.xpath("//div"));

java.util.List<WebElement> elements = driver.findElements(By.name("name"));

System.out.println("Number of elements:" + elements.size());

for (int i = 0; i < elements.size(); i++) {

System.out.println("Radio button text:" + elements.get(i).getAttribute("value"));

}

9. Image in Selenium Webdriver

we can using either By.cssSelector or By.xpath. The first method is more preferred because of its simplicity.

ex,

String baseUrl = "https://www.facebook.com/login/identify?ctx=recover";

driver.findElement(By.cssSelector("a[title=\"Go to Facebook home\"]")).click();

//verify that we are now back on Facebook's homepage

if (driver.getTitle().equals("Facebook - log in or sign up")) {

System.out.println("We are back at Facebook's homepage");

} else {

System.out.println("We are NOT in Facebook's homepage");

}

driver.close();

10. Working on Dropdowns

To handle drop down, we need to use select class.

By using selectByVisibleText() method in selecting multiple options in multi SELECT element.

ex,

Select fruit=new Select(driver.findElement(By.id("Fruits")))

fruit.selectByVisibleText("Banana");

fruit.selectByIndex(2);

friut.deselectByVisibleText("Apple");

fruit.selectByValue("243");

methods,

deselectByIndex("233");

deSelectAll();

11. Handling LinkText and partial link text

Use, By.LinkText("Google").click;

By.PartialLinkText("go").click();

Note, when handling Multiple links with the same Link Text,

locators viz... By.xpath(), By.cssSelector() or By.tagName() are used.

Most commonly used is By.xpath(). It is the most reliable one but it looks complex and non-readable too, since link text are case sensitive.

ex,

driver.findElement(By.partialLinkText("rigister")).click();

driver.findElement(By.partialLinkText("REGISTER")).click();

To access inner and outer linkText,

driver.findElement(By.partialLinkText("rigister")).click();

driver.navigate.back();

driver.findElement(By.partialLinkText("REGISTER")).click();

findElements() & By.tagName("a") method finds all the elements in the page matching the locator criteria

12. Handling Keyboard & Mouse Events

Handling special keyboard and mouse events are done using the Advanced User Interactions API. It contains the Actions and the Action classes

clickAndHold() Clicks (without releasing) at the current mouse location.

contextClick() Performs a context-click at the current mouse location. (Right Click Mouse Action)

doubleClick() Performs a double-click at the current mouse location.

dragAndDrop(source, target) Performs click-and-hold at the location of the source element, moves to the location of the target element, then releases the mouse.

Parameters:

source- element to emulate button down at.

target- element to move to and release the mouse at.

dragAndDropBy(source, x-offset, y-offset) - Performs click-and-hold at the location of the source element, moves by a given offset, then releases the mouse.

|  |  |
| --- | --- |
| **keyDown(modifier\_key)** | Performs a modifier key press. Does not release the modifier key - subsequent interactions may assume it's kept pressed. **Parameters**:  modifier\_key - any of the modifier keys (Keys.ALT, Keys.SHIFT, or Keys.CONTROL) |
| **keyUp(modifier \_key)** | Performs a key release. **Parameters**:  modifier\_key - any of the modifier keys (Keys.ALT, Keys.SHIFT, or Keys.CONTROL) |
| **moveByOffset(x-offset, y-offset)** | Moves the mouse from its current position (or 0,0) by the given offset. **Parameters**:  x-offset- horizontal offset. A negative value means moving the mouse left.  y-offset- vertical offset. A negative value means moving the mouse down. |
| **moveToElement(toElement)** | Moves the mouse to the middle of the element. **Parameters**:  toElement- element to move to. |
| **release()** | Releases the depressed left mouse button at the current mouse location |
| **sendKeys(onElement, charsequence)** | Sends a series of keystrokes onto the element. **Parameters**:  onElement - element that will receive the keystrokes, usually a text field  charsequence - any string value representing the sequence of keystrokes to be sent |