​

Selenium Java VS Selenium Python:

1. Creating WebDriver instance:

Java:

WebDriver driver = new FirefoxDriver();

System.setProperty("webdriver.chrome.driver", "/path/to/chromedriver");

WebDriver driver = new ChromeDriver();

Python:

import os

from selenium import webdriver

driver = webdriver.FireFox();

chromedriver = "/Users/adam/Downloads/chromedriver"

os.environ["webdriver.chrome.driver"] = chromedriver

driver = webdriver.Chrome(chromedriver)

2. Implicit Wait

Java:

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

Python:

driver.implicitly\_wait(10)

3 - Explicit Wait:

Java:

WebDriver driver = new FirefoxDriver();

driver.get("http://somedomain/url\_that\_delays\_loading");

WebElement myDynamicElement = (new WebDriverWait(driver,10)).until(ExpectedConditions.presenceOfElementLocated(By.id("myDynamicElement")));

Python:

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import expected\_conditions as EC

driver = webdriver.Firefox()

driver.get("http://somedomain/url\_that\_delays\_loading")

try:

element = WebDriverWait(driver, 10).until(

EC.presence\_of\_element\_located((By.ID, "myDynamicElement"))

)

finally:

driver.quit()

4. Fluent Wait:

Java:

Wait<WebDriver> wait = new FluentWait<WebDriver>(driver)

.withTimeout(30, SECONDS)

.pollingEvery(5, SECONDS)

.ignoring(NoSuchElementException.class);

WebElement foo = wait.until(new Function<WebDriver, WebElement>()

{

public WebElement apply(WebDriver driver) {

return driver.findElement(By.id("foo"));

}

});

Python:

In python the WebDriverWait class can be used for this purpose by providing some extra functionality:

wait = WebDriverWait(driver, 10, poll\_frequency=1, ignored\_exceptions=[ElementNotVisibleException, ElementNotSelectableException])

5.Drag and Drop :

Java

Actions a = new Actions(driver);

a.dragAndDrop(source, target).build().perform()

Python:

from selenium.webdriver.common.action\_chains import ActionChains

ActionChains(driver).drag\_and\_drop(source\_element, dest\_element).perform()

6.Slider Action

Java:

Actions a = new Actions(driver);

a.dragAndDropBy(elementToSlide, xoffset, yoffset).build().perform(

Python:

ActionChains(driver).drag\_and\_drop\_by\_offset(elementToSlide, xoffset, yoffset)

7.Working with Alerts:

Java:

Alert a = driver.switchTo().alert().accept()

driver.switchTo().alert().dismiss()

driver.switchTo().alert().getText()

driver.switchTo().alertI().sendKeys(“text”)

Python:

alert = driver.switch\_to\_alert()

text = alert.text

alert.dismiss()

alert.accept()

8.Taking Screenshots

Java:

File scrFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);

 //The below method will save the screen shot in d drive with name "screenshot.png"

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

Python:

driver.take\_screenshot\_as\_file(‘Complete\_path/name’)

Page Scrolling:

Java:

WebDriver driver = new FirefoxDriver();

JavascriptExecutor jse = (JavascriptExecutor)driver;

jse.executeScript("window.scrollBy(0,250)", "");

Scrolling to the bottom of the page:

((JavascriptExecutor) driver).executeScript("window.scrollTo(0, document.body.scrollHeight)");

Python:

driver.execute\_script("window.scrollTo(0, Y)")

Switch Tabs:

ArrayList<String> tabs2 = new ArrayList<String> (driver.getWindowHandles());

driver.switchTo().window(tabs2.get(1));

driver.close();

driver.switchTo().window(tabs2.get(0));

Switch to Frames:

driver.switchTo().frame(driver.findElement(By.id("frameId")));

//do your stuff

driver.switchTo().defaultContent();

Selenium Grid:

Running multiple tests across different browsers, operating systems and machines in parallel.

Selenium Grid uses a hub-node concept where you only run the test on a single machine called a hub, but the execution will be done by different machines called nodes.

Used when:

We want to run tests against different browsers, operation system and machines all at the same time.

Save time for execution of our suite.

HUB:

The hub is the central point where you load your tests into.

There should only be one hub in a grid.

The hub is launched only on a single machine, say, a computer whose O.S is Windows 7 and whose browser is IE.

The machine containing the hub is where the tests will be run, but you will see the browser being automated on the node.

NODE:

Nodes are the Selenium instances that will execute the tests that you loaded on the hub.

There can be one or more nodes in a grid.

Nodes can be launched on multiple machines with different platforms and browsers.

The machines running the nodes need not be the same platform as that of the hub.

Starting NODE and HUB:

— Note down the IP address of the Hub Machine.

Run Selenium Grid in java -jar selenium-server-standalone-2.30.0.jar -role hub

Go to in the browser to check if the Grid is running - http://localhost:4444/grid/console

In the Node machine, go to ‘http://192.168.1.3:4444/grid/console' to check if the the node can access the HUB’s web interface.

In the HUB machine, launch the HUB machine by following command:

Go to the Node machine and type:

￼

Designing Scripts:

we need to use DesiredCapabilites and the RemoteWebDriver to run the grid.

DesiredCapabilites is used to set the type of browser and OS that we will automate

RemoteWebDriver is used to set which node (or machine) that our test will run against.

JAVA:

￼

￼

￼

￼