Selenium

Class 7

Agenda

Handle Windows
Synchronization & Waits in Selenium

Multiple Windows in Selenium Webdriver

There will be times when we will required to perform some testing, where the testing operations open a new browser/tab, test case may required you to perform some tasks on the newly opened browser window/tab and return back to original window to perform the remaining tasks.

Even if the window/tab is currently on focus but still it is not an active window, so to perform some tasks we need to switch to new browser window/tab in selenium webdriver.

Situations when we are likely to deal with multiple windows:

- Filling forms may require selecting the date from a separately opened window
- Clicking on some link/button can open another window
- Handling Advertisement windows

Switch Window Commands

Selenium WebDriver assigns an alphanumeric id to each window as soon as the WebDriver object is instantiated.

This unique alphanumeric id is called window handle. Selenium uses this unique id to switch control among several windows.

In simple terms, each unique window has a unique ID, so that Selenium can differentiate when it is switching controls from one window to the other.

GetWindowHandle Command
Purpose: To get the window handle of the current window.

```
//Return a string of alphanumeric window handle
String  handle= driver.getWindowHandle();
```

Switch Window Commands

GetWindowHandles Command
Purpose: To get the window handle of all the current windows.

```
//Return a set of window handle
 Set<String> handle= driver.getWindowHandles();
```

SwitchTo Window Command
Purpose: WebDriver supports moving between named windows using the “switchTo” method.

```
driver.switchTo().window("windowName");
```

Synchronization Waits in Selenium

What is wait ? Why we need it ?

It is the process of matching the speed of action and reaction, selenium does the action and web application shows the reaction.

Example: Click Login button gmail login (selenium does this action) web application navigates to the inbox(reaction).

Selenium Wait commands play an important role while executing Selenium tests. When a page is loaded to browser, the elements within that page may load at different time intervals.

This makes locating elements difficult, if the element is not present in the DOM, it will raise **ElementNotVisibleExceptionexception.**

Waits help the user to troubleshoot issues while redirecting to different web pages by refreshing the entire web page and reloading the new web elements.

Synchronization Waits in Selenium

In most of the cases the selenium does the action in micro or milliseconds but the application takes time in seconds.

At the same time selenium will not wait till it loads or becomes visible at the time selenium will not able to find the element within the java processing time so selenium throws “**NoSuchElementException**’.

We cannot alter the speed of the application, so we must slow down selenium.

Page Load Timeout in Selenium Webdriver

Page load timeout in selenium requests/set the time limit for a page to load, if the page is not loaded within the given time frame selenium throws TimeOutException exception.

Page Load timeout is applicable only to driver.get() and driver.navigate().to() methods in selenium webdriver.

Page load timeout is not applicable when user clicks a link to open a page.

driver.manage().timeouts().pageLoadTimeout(30, TimeUnit.SECONDS);

Waits in Selenium Webdriver

Selenium Webdriver provides two types of waits:

ImplicitWait
ExplicitWait: WebDriver Wait
Fluent Wait

Implicit wait

An implicit wait is to tell WebDriver to poll the HTML DOM for a certain amount of time when trying to find an element or elements if they are not immediately available.

The default setting is 0. Once when we define the implicit wait, it will set for the life of the WebDriver object instance.

It is a mechanism which will be written once and applied for entire session automatically. It should be applied immediately once we initiate the Webdriver.

Implicit wait

Implicit wait will not work for all the commands/statements in the application.

It will work only for "FindElement" and "FindElements" statements.

If we set implicit wait, find element will not throw an exception if the element is not found in first instance, instead it will poll for the element until the timeout and then proceeds further.

We should always remember to add the below syntax immediately below the Webdriver statement.

driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);

Implicit wait

import java.util.concurrent.TimeUnit – To be able to access and apply implicit wait in our test scripts, we are bound to import this package into our test script

In implicit wait, if webdriver cannot find webelement in starting, it will wait for specified time duration and only then throw NoSuchElementException

Webdriver will not search during this wait interval. Once specified time is over, it will try to search again for last time before throwing any exception.

If element will be found before the specified time, it will return that element and the the rest of the time will be ignored

In Fluent wait and explicit wait, if wait time is 30 seconds, webdriver tries to search for element after some specified time say 500 milliseconds.

Explicit wait

Types of Explicit Waits:

WebDriverWait

FluentWait

- We need to define a wait statement for certain condition to be satisfied until the specified timeout period.
- If the Webdriver finds the element within the timeout period the code will get executed.
- Explicit wait is mostly used when we need to Wait for a specific content/attribute change after performing any action, like when application gives AJAX call to system and get dynamic data and render on UI.

  Example: Like there are drop-downs Country and State, based on the country value selected, the values in the state drop-down will change, which will take few seconds of time to get the data based on user selection.

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

Types of ExpectedCondition in Explicit wait

We can use WebDriverWait class in many different cases. When ever we need to perform any operation on element, we can use webdriver wait to check if the element is present or visible or enabled or disabled or clickable etc.

ExpectedConditions class provides a great help to deal with scenarios where we have to ascertain for a condition to occur before executing the actual test step.

ExpectedConditions class comes with a wide range of expected conditions that can be accessed with the help of the WebDriverWait reference variable and until() method.

Types of ExpectedCondition in Explicit wait

- **elementToBeClickable()**

Selenium waits for an element to become clickable like disabled state to normal state, selenium moves to next line of code if the element becomes clickable before the timeout otherwise selenium throws 'TimeOutException'

- **visibilityOfElementLocated()**

Selenium waits for visibility of element when we use 'visibilityOfElementLocated()', when element is visible, it moves to next line of code, in case if element is not visible before the specified time, then selenium Throws 'TimeOutException'

- **elementToBeSelected()**

Selenium waits for an element to be selected when we use 'elementToBeSelected()', when selenium finds the element is selected it moves to next line of code, in case if element is not selected before the specified time, then selenium Throws 'TimeOutException'

Types of ExpectedCondition in Explicit wait

- **textToBePresentInElement()**

Selenium waits for an element to have particular text when we use 'textToBePresentInElement()', when selenium finds the element have particular text it moves to next line of code, in case if element does not have text before the specified time, then selenium Throws 'TimeOutException'

- **alertIsPresent()**

Selenium waits for an alert to be present when user writes 'alertIsPresent()', when selenium finds the alert it moves to next line of code, in case if alert is not present before the specified time, then selenium Throws 'TimeOutException'

Note- 95 % of our test can be handled via explicit wait

Fluent Wait

FluentWait can define the maximum amount of time to wait for a specific condition and frequency with which to check the condition before throwing an “ElementNotVisibleException” exception.

We use FluentWait commands mainly when we have web elements which sometimes visible in few seconds and sometimes take more time than usual. Mainly in Ajax applications. We could set the default polling period based on our requirement. We could ignore any exception while polling an element.

```
FluentWait wait = new FluentWait(driver);
wait.withTimeout(Duration.ofSeconds(30));
wait.pollingEvery(Duration.ofSeconds(1));
wait.ignoring(NoSuchElementException.class);
```