Selenium

Class 9

Agenda

Synchronization & Waits in Selenium

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.implicitwait(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.

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.

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

- 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(45, TimeUnit.SECONDS);
wait.pollingevery(5, TimeUnit.SECONDS);
wait.ignoring(NoSuchElementException.class);
```

Test Case

TC 1: Verify element is present

1. Open chrome browser
2. Go to “https://the-internet.herokuapp.com/”
3. Click on “Click on the “Dynamic Loading” link
4. Click on “Example 1...” and click on “Start”
5. Verify element with text “Hello World!” is displayed
6. Close the browser

TC 2: Verify element is clickable

1. Open chrome browser
2. Go to “https://the-internet.herokuapp.com/”
3. Click on “Click on the “Dynamic Controls” link
4. Select checkbox and click Remove
5. Click on Add button and verify “It's back!” text is displayed
6. Close the browser

TC 3: Verify element is enabled

1. Open chrome browser
2. Go to “https://the-internet.herokuapp.com/”
3. Click on “Click on the “Dynamic Controls” link
4. Click on enable button
5. Enter “Hello” and verify text is entered successfully
6. Close the browser