Practical No 7

Synchronization in selenium

**Synchronization** meaning: when two or more components involved to perform any action, we expect these components to work together with the same pace. The co-ordination between these components to run parallel is called Synchronization.

Synchronization (Wait) in [Selenium](https://www.ecanarys.com/software-testing) has a great significant value.

Now a day, we see most of the web applications are developed by using Javacript and Ajax where it might happen that some of the elements may load at distant time intervals, due to which we may encounter “ElementNotVisibleException” or “NoSuchElementException” exceptions. In such scenarios, we use Synchronization/wait to overcome these kinds of exceptions.

.

Synchronization can be classified into two categories:

1. **Unconditional Synchronization**
2. **Conditional Synchronization**

Unconditional synchronization indicates timeout value alone and makes tool to continue further only when specified time is elapsed. One such functions is:

*System.Threading.Thread.Sleep();*

*What will happen?*

1. This statement is helpful only when we are interacting with third party systems like interfaces because it is not possible to write conditions to check the status.
2. Now, we want tool to wait for certain amount of time and then perform the next actions sequence.
3. This statement makes the tool to wait for certain time even though the application is ready, thus increasing the chance of unnecessary wait and hence giving the scope to **Conditional Synchronization.**

**Conditional** Synchronization: By using conditional synchronization we can specify timeout duration along with some desired conditions to check and then throw an error if nothing happens.

**Types of Conditional Synchronization:**

1. **Implicit Wait**
2. **Explicit Wait**

**Implicit Wait**: Implicit Wait tells the WebDriver to Wait until the stated time before throwing the NoSuchElement/ElementNotVisible exception. Waiting time across the test script between each consecutive steps are taken by default. Hence, next testStep will execute only when the specified time is elapsed post executing the previous testStep.

**Syntax:**

*driver.Manage().Timeouts().ImplicitlyWait(TimeSpan.FromSeconds(30));*

The Implicit wait takes one argument: TimeSpan (timeToWait)

* FromMilliSeconds
* FromSeconds
* FromMinutes
* FromHours
* FromDays

**Eg:** The wait time specified in Implicit Wait command is say 30seconds. The specified time (30seconds) is default for all the succeeding testSteps thus giving rise to execution time. To overcome this issue, Explicit waits are introduced to apply waits whenever it is required while executing the testSteps.

**Explicit Wait:** Explicit waits are very good to use when page loads dynamically. Explicit Wait tells the WebDriver to Wait until the specified condition is met or maximum time elapses before throwing NoSuchElement (or) ElementNotVisible Exceptions. Explicit waits are applied for the specified testStep in test script.

To use Explicit waits, we must first create instance for **“WebDriverWait”** class.

*WebDriverWait wait = new* *WebDriverWait(driver, TimeSpan.FromSeconds(30));*

*wait.Until(ExpectedConditions.VisibilityOfAllElementsLocatedBy(By.Id(element\_ID)));*

The reference variable “wait” informs the WebDriver to wait until the Expected condition to occur (or) Wait for the specified time of 30seconds, whichever shows in first place before throwing an exception.

Following are the **ExpectedConditions:**

* AlertIsPresent()
* ElementSelectionStateToBe()
* VisibilityOfAllElementsLocatedBy()
* ElementToBeSelected()
* FrameToBeAvaliableAndSwitchToIt()
* InvisibilityOfTheElementLocated()
* InvisibilityOfElementWithText()
* ElementToBeClickable()
* PresenceOfAllElementsLocatedBy()
* PresenceOfElementLocated()
* TextToBePresentInElement()
* TextToBePresentInElementLocated()
* TextToBePresentInElementValue()
* TitleIs()
* TitleContains()
* VisibilityOf()
* VisibilityOfAllElements()
* VisibilityOfElementLocated()

The **Implicit wait** is simple when compared to Explicit wait and can be declared in setup method of the test script in single line of code. However, the Implicit wait has drawbacks despite being easy and simple to apply. Hence rising scope to use **Explicit Waits** because these waits can be applied for a specified testStep in test script.

**Practical No 7 A based on Implicit wait**

**package** Pracs;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.firefox.FirefoxDriver;

**public** **class** Seven\_A {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

System.*setProperty*("webdriver.gecko.driver","C:\\Users\\Admin\\Downloads\\SoftwareTestingLab\\geckodriver.exe");

WebDriver wd = **new** FirefoxDriver(); wd.get("https://opensource-demo.orangehrmlive.com/");

wd.findElement(By.*id*("txtUsername")).sendKeys("admin");//locator id wd.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);//implicit wait

wd.findElement(By.*name*("txtPassword")).sendKeys("admin123");//locate r name

wd.findElement(By.*className*("button")).click();//locator className

wd.findElement(By.*partialLinkText*("Welcome")).click();//locator partiallinkText

wd.findElement(By.*xpath*("//\*[@id=\"welcome\"]")).click(); wd.findElement(By.*linkText*("Logout")).click();//locator linkText

}

}

**Practical No 7 B based on Explicit wait**

**package** Pracs;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.firefox.FirefoxDriver;

**import** org.openqa.selenium.support.ui.ExpectedConditions;

**import** org.openqa.selenium.support.ui.WebDriverWait;

**public** **class** Seven\_B {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

System.*setProperty*("webdriver.gecko.driver","C:\\Users\\Admin\\Downloads\\SoftwareTestingLab\\geckodriver.exe");

WebDriver wd = **new** FirefoxDriver(); wd.get("https://opensource-demo.orangehrmlive.com/");

WebDriverWait wt = **new** WebDriverWait(wd,10); wd.findElement(By.*id*("txtUsername")).sendKeys("admin");//locator id

wd.findElement(By.*name*("txtPassword")).sendKeys("admin123");//locate r name

wd.findElement(By.*className*("button")).click();//locator className

wd.findElement(By.*partialLinkText*("Welcome")).click();//locator partiallinkText

wd.findElement(By.*xpath*("//\*[@id=\"welcome\"]")).click();

wt.until(ExpectedConditions.*visibilityOfElementLocated*(By.*linkText*("Log out")));

wd.findElement(By.*linkText*("Logout")).click();//locator linkText

}

}