Synchronization

The process of matching selenium script execution time with the application loading time is called synchronization.

Sometimes application slows down for reasons like server issues, network issues, heavy load etc but the script checks continuously for the particular web element on the web page which might result in selenium exceptions like

* NoSuchElementException
* TimeOutException
* ElementNotInteractableException
* ElementClickInterceptedException

Hence we synchronize application speed with the script execution using wait statements.

Types of synchronization statements:

1. Thread.sleep(long milli\_seconds):

* It is java wait statement which pauses script execution for specified amount of time.
* This wait statement should be specified every time when the script needs to be paused.
* It accepts long data type time, in milli seconds, as arguments.
* It throws InterruptedException.

Usage: Thread.sleep(2000);

1. Implicit Wait:

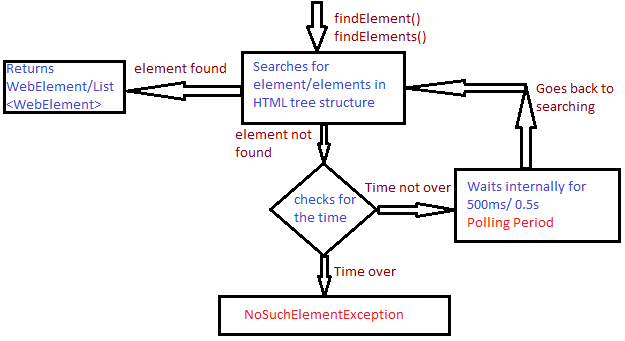
* It is selenium wait statement which synchronizes SearchContext interface methods of WebDriver class diagram – findElement() and findElements()
* Implicit wait is global wait statement it is given only once in the beginning of the script execution. It synchronizes the above methods throughout the script.

Usage: driver.manage().timeouts().implicitlyWait(Duration

.ofSeconds(long time\_in\_seconds));

* Return type of implicitlyWait() is Timeouts.
* implicitlyWait() is present in org.openqa.selenium.WebDriver.Timeouts package

Work Flow Chart:



* When implicitly wait is given it searches for the element/ elements in the HTML tree structure.
* If the element/ elements is found WebElement/ List<WebElement> is returned respectively.
* If the element is not found then it checks for the specified time.
* If the time is over, it throws NoSuchElementException.
* If the time is not over, it waits internally for 500 milli seconds or 0.5 seconds. This internal waiting period is called as polling period. Then it goes back to search element/ elements.
* The above process repeats until element/ elements is found or time is over.

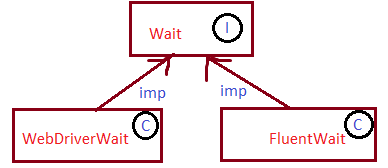
Drawback:

* Synchronizes only findElement() and findElements() methods.

1. Explicit Wait:

* It synchronizes all the WebDriver methods including findElement() and findElements() methods.
* As the name implies explicit wait statement should be specified explicitly every time when a method is used.
* Here condition should be mentioned.
* It has two classes which implement Wait interface.

1. WebDriverWait
2. FluentWait



1. WebDriverWait:

* WebDriverWait is the implementing class of Wait interface.
* It is selenium wait statement which is a part of explicit wait statement.

Usage:

WebDriverWait wait = new WebDriverWait(driver,

Duration.ofSeconds(time\_in\_seconds));

wait.until(ExpectedConditions.visibilityOf(element);

or

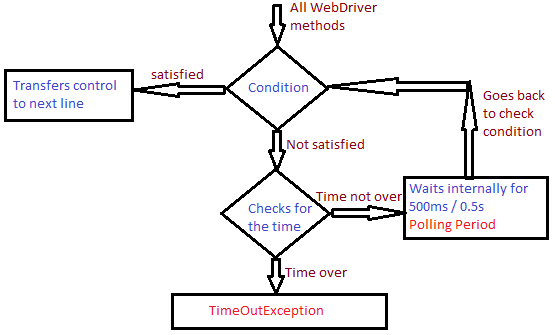
wait.until(ExpectedCondition.titleContains(“title”);

or

wait.until(ExpectedConditions.elementToBeClickable(

element));

Work Flow Chart:



* When explicitly wait is given using WebDriverWait class it checks if condition is satisfied or not.
* If the condition is satisfied, it transfers the control to next line for execution.
* If the condition is not satisfied, it checks for the specified time.
* If the time is over, it throws TimeOutException
* If the time is not over, it internally waits for 500 milli seconds or 0.5 seconds. This internal wait period is called polling period. Then it goes back to check the condition.
* This process repeats until time is over or condition is satisfied.

1. FluentWait:

* FluentWait is the implementing class of Wait interface.
* It is selenium wait statement which is a part of explicit wait.
* The work flow of FluentWait is similar to that of WebDriverWait except that we can customize the polling period and it also helps ignore any exception which might occur before the specified time is over.
* Unlike WebDriverWait, FluentWait doesnot have pre-defined methods to provide the condition for wait. User should give a customized method for condition.

Usage:

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

.withTimeout(Duration.ofSeconds(time))

.pollingEvery(Duration.ofSeconds(time))

.ignoring(Exception\_class));

WebElement element = wait.until(new

Function<WebDriver,WebElement>(){

public WebElement apply(WebDriver driver){

WebElement e = driver.findElement(<locator>);

if(e.isDisplayed())

return e;

else

return null; }});

4.Custom wait:

* It is user defined wait statement.
* Here looping statement with exception handling concept is utilised to design a generic wait method.

Ex:

int count = 20;

for(int i = 0; i < count; i++){

try {

WebElement e = driver.findElement(locator);

break;

}

catch {

Thread.sleep(1000);

}

}