Synchronization

- Synchronization is a mechanism which involves two or more components working parallel with each other.
- Process of matching Selenium speed with application is called as Synchronization.

Generally, there are two different categories of synchronization in test automation.

1. Unconditional synchronization.
2. Conditional synchronization.

Unconditional Synchronization

In this case, only the timeout value to be specified. The tool will wait till certain time before proceeding.

Example: sleep (5)

Import the "sleep" method from the following module

```
from time import sleep
```

The major disadvantage is that at sometimes, the tool will be made to wait unnecessarily even when the application is ready.

Write a script to synchronize using the unconditional synchronization method sleep?

```
1 from selenium import webdriver
2 import time
3
4 driver = webdriver.Chrome()
5 driver.implicitly_wait(30)
6 driver.get('https://demo.actitime.com')
7
8 driver.find_element_by_partial_link_text("psfdsassword?").click()
9 time.sleep(5)
10 driver.find_element_by_id("forgetPasswordEmailOrUsername")
11
12 driver.close()
```

This method will stops the execution for 5 seconds.

Drawback of unconditional synchronization

- If we use **time.sleep** method we should specify it in all the locations where application is slow. This will increase the time taken to write script, it consumes lot of space and increases maintenance of the script and it always waits for specified duration.
- Ex: if the duration is 20 sec, it will always waits for 20 sec even though element is displayed in the 5 sec.
- To overcome all these limitations, we should use the synchronization option provided by Selenium.

Conditional Synchronization

In this case, a condition also will be specified along with the timeout values.

The tool will wait to check the condition and will come out if nothing happens. However it is important to set a time and value also in conditional synchronization, so that the tool will proceed even if the condition is not met.

Following are the different types of conditional statements which are supported by selenium.

1. Implicit wait
2. Explicit wait
3. Page load time out
4. Set script time out

Note:

- On real time applications when Selenium try to find the element it may through 'NoSuchElementException' even though specified locator is correct. To handle this we can use synchronization methods.

Implicit Wait

- Implicitly wait is a method which is used to tell the webdriver to wait for an element for certain amount of time before throwing the no such element exception.
- The default setting is 0. Once set, the implicit wait is set for the life of the Web Driver object instance until it changed again.
- This means that we can tell Selenium that we would like it to wait for a certain amount of time before throwing an exception that if it cannot find the element on the page.

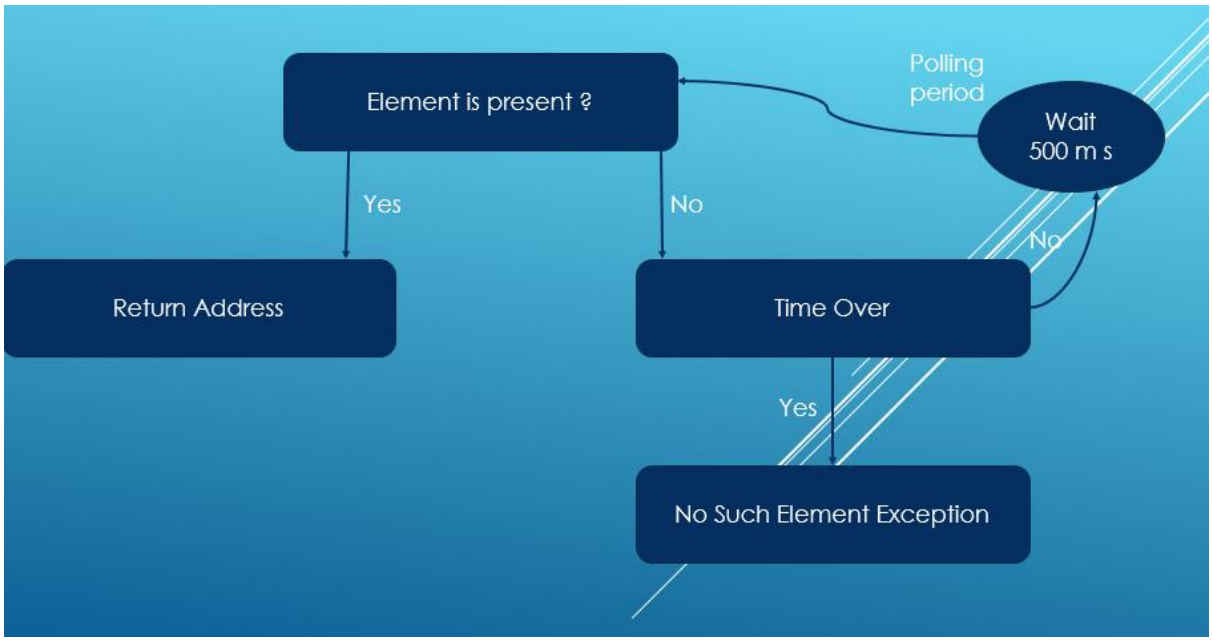
```
4 driver = webdriver.Chrome()
5
6 driver.implicitly_wait(30)
7
```

Implicitly wait method takes one arguments,

1. Time span -> it takes the time span as a seconds

- If we use **implicitly_wait** then if the element is not located, the **find_element** method will keep searching for the element after every 500 MILLISECONDS. This duration is called as "**Polling Period**".
- If the element is not located even after the duration then we get **NoSuchElementException**.

Flow diagram of implicit wait



Explanation:

1. When the control comes to find element it will check whether the element is present or not.
2. If the element is present returns the address of the matching element (Web element).
3. If the element is not present it will check for the time out, if the time is over it will throw the no such element exception.
4. If the time is not over then it waits for 500 mille seconds, the time duration taken to wait for the element is called as polling period. After waiting it will continue to check for the element.
5. Even after the specified time, if the element is not available then we will get the no such element exception.

Important:

Q. Can we specify **Implicitly_wait** statement multiple times in the Selenium script?

A. Yes

Q. Is it necessary to write **Implicitly_wait** statement multiple times?'

A. No

Example Program:

```
from selenium import webdriver

driver = webdriver.Chrome()
driver.implicitly_wait(30)
driver.maximize_window()
driver.get("https://docs.seleniumhq.org/")

driver.find_element_by_id("username").send_keys("admin")
driver.find_element_by_name("pwd").send_keys("manager")
driver.find_element_by_id("loginButton").click()
driver.find_element_by_id("logoutLink")

driver.quit()
```

Drawback of implicit wait:

- The duration specified in **implicitly_wait** statement is used only by **find_element** and **find_elements.** But do not by any other methods.

Program to synchronize the script without using implicit wait and sleep method.

```
1 from selenium import webdriver
2 from selenium.common.exceptions import NoSuchElementException
3
4 driver = webdriver.Chrome()
5 driver.get('https://demo.actitime.com/')
6
7 driver.find_element_by_css_selector("input#username").send_keys("admin")
8 driver.find_element_by_css_selector("input.textField.pwdfield").send_keys("manager")
9 driver.find_element_by_id("loginButton").click()
10
11 while(True):
12     try:
13         driver.find_element_by_id("logoutLink").click()
14         print('Clicking on the logout button')
15         break
16     except NoSuchElementException:
17         print('')
18
19 driver.quit()
```