24. WebDriver Waiting Strategies

Synchronization problem

- ➔ Synchronization issues in Selenium occur when the test script tries to interact with a web element that is not yet available, visible, or enabled on the web page.
- ➔ This can happen due to various factors, such as network latency, page loading time, AJAX calls, animations, or user interactions. When this happens, the test script may throw an exception, skip a step, or produce incorrect results.
  - ◆ **NoSuchElementException**
  - ◆ **NoSuchWindowException**
  - ◆ **NoSuchFrameException**
  - ◆ **NoAlertPresentException**
  - ◆ **ElementNotVisibleException**
  - ◆ **ElementNotSelectableException**
  - ◆ **TimeoutException**
- ➔ In Automation Testing, we use waiting strategies to make applications under test and the Test Automation Tool work in parallel.
- ➔ Sometimes instead of using wait commands we use **python's time module**

time.sleep(time in seconds)

- ➔ **syntax** ⇒ time.sleep(10)
- ➔ **Adv**
  - ◆ simple to use
- ➔ **DisAdv**
  - ◆ Performance of the script is very poor.
  - ◆ If the element is not available within the time mentioned , still there is a chance of getting an exception.

SleepDemo.py

```
import time
from selenium import webdriver
from selenium.webdriver.common.by import By
opt = webdriver.ChromeOptions()
opt.add_experimental_option("detach",True)
driver = webdriver.Chrome(options=opt)
driver.get("https://www.selenium.dev/selenium/web/dynamic.html")
driver.maximize_window()
```

Synchronization issue

```
driver.find_element(By.ID, "adder").click()
red_box = driver.find_element(By.ID, "box0")
print(red_box.is_displayed())
```

Synchronization issue handling using time.sleep()

```
driver.find_element(By.ID, "adder").click()
time.sleep(3)  # Sleep for 3 seconds
```

```
red_box = driver.find_element(By.ID, "box0")
print(red_box.is_displayed())  # True
driver.quit()
```

Wait commands in selenium Webdriver

1. Implicit wait
2. Explicit wait
3. Fluent Wait

Implicit Wait

- ➔ The implicit wait tells WebDriver to wait a certain amount of time before it throws an "**Exception**".
- ➔ The default setting of implicit wait is zero.
- ➔ Once you set the time, the web driver will wait for that particular amount of time before throwing an exception.
- ➔ **syntax** ⇒ driver.implicitly_wait(10)
- ➔ **Adv**
  - ◆ Single statement
  - ◆ Performance will not be reduced.(If the element is available within the time it proceeds to execute further statements).
- ➔ **DisAdv**
  - ◆ If the element is not available within the time mentioned , still there is a chance of getting an exception.

ImplicitWaitDemo.py

```
from selenium import webdriver
from selenium.webdriver.common.by import By
opt = webdriver.ChromeOptions()
opt.add_experimental_option("detach",True)
driver = webdriver.Chrome(options=opt)
driver.implicitly_wait(3)
driver.get("https://www.selenium.dev/selenium/web/dynamic.html")
driver.maximize_window()
driver.find_element(By.ID, "adder").click()
red_box = driver.find_element(By.ID, "box0")
print(red_box.is_displayed())  # True
driver.quit()
```

Note

- ➔ If the implicit wait written at the beginning then it is applicable for all the statements in the script that means wherever the synchronization problem occurs implicit wait takes care of it.

Explicit Wait

- ➔ Explicit waits are a concept from the dynamic wait
- ➔ Applies only to the specified element and can be customized with different conditions like visibility, clickability, or presence.
- ➔ When we use explicit wait no need of find element method it is already inclusive in explicit wait.

- ➔ It can be implemented by the **WebDriverWait** class.it contains Declaration and Usage
- ➔ **syntax**
  - ◆ **Declaration**
    - mywait = WebDriverWait(driver, 10)
  - ◆ **Usage**
    - element = mywait.until(EC.visibility_of_element_located((By.ID, "elementId")))
- ➔ **Adv**
  - ◆ More effectively works.
- ➔ **Dis**
  - ◆ Multiple places
  - ◆ Feel some difficulty

ExplicitwaitDemo.py

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
opt = webdriver.ChromeOptions()
opt.add_experimental_option("detach",True)
driver = webdriver.Chrome(options=opt)
driver.get("https://www.selenium.dev/selenium/web/dynamic.html")
```

Declare explicit wait

```
mywait = WebDriverWait(driver, 10)
driver.find_element(By.ID, "adder").click()
```

Usage of explicit wait

Example 1

```
red_box = driver.find_element(By.ID, "box0")
mywait.until(EC.visibility_of(red_box))
```

Example 2

```
revealed_box = mywait.until(EC.visibility_of_element_located((By.ID, "box0")))
print(red_box.is_displayed())  # True
driver.quit()
```

Fluent Wait

- ➔ Explicit Wait with **polling interval** and **ignoring certain exceptions** is Fluent Wait.
- ➔ Useful for scenarios where the element's appearance is unpredictable or requires retries at intervals.
- ➔ **Declaration**
  - ◆
```
mywait = WebDriverWait(driver,10,poll_frequency=2,ignored_exceptions=[NoSuchElementException,
                                                                      ElementNotVisibleException,
                                                                      ElementNotSelectableException,
                                                                      Exception]
                                                                    )
```
- ➔ **Usage**
  - ◆ element = mywait.until(EC.visibility_of_element_located((By.ID, "elementId")))

FluentwaitDemo.py

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import NoSuchElementException, ElementNotVisibleException,
ElementNotSelectableException
opt = webdriver.ChromeOptions()
opt.add_experimental_option("detach",True)
driver = webdriver.Chrome(options=opt)
driver.get("https://www.selenium.dev/selenium/web/dynamic.html")
```

Declare Fluent wait

```
mywait = WebDriverWait(driver,10,poll_frequency=2,ignored_exceptions=[NoSuchElementException,
                                    ElementNotVisibleException,
                                    ElementNotSelectableException,
                                    Exception]
                                )
driver.find_element(By.ID, "adder").click()
```

Usage of fluent wait

```
red_box = mywait.until(EC.visibility_of_element_located((By.ID, "box0")))
print(red_box.is_displayed())  # True
driver.quit()
```

Note

- If the element is not present, the condition never becomes true so execution will stop there and throw an exception error.
- But at a certain point of time we have to exit from the statement and then continue with the rest of the statement for that we specify this cut off time.
- To handle exception error we use Ignored_conditions in webdriver declaration otherwise no difference between implicit and explicit wait.
- Poll_frequency defines how frequently the wait should check the condition. By default, it is 0.5 second.

Page Load Time

- ➔ When testing web application If the page does not load within the specified time, a **TimeoutError** will be thrown.
- ➔ To avoid error we use set_page_load_timeout(10) in Selenium WebDriver to specify the **maximum amount of time to wait for a web page to load** before throwing an exception.
  - ◆ **driver.set_page_load_timeout(10)**
- ➔ This method is useful when testing web applications with variable page load times. It allows you to set a maximum loading time, ensuring that your tests do not hang indefinitely if a page fails to load.

PageLoadTime.py

```
from selenium import webdriver
opt = webdriver.ChromeOptions()
opt.add_experimental_option("detach",True)
driver = webdriver.Chrome(options=opt)
```

```
try:
  driver.set_page_load_timeout(10)  # 10 seconds
  driver.get("https://testautomationpractice.blogspot.com/")
  print("Page loaded successfully!")
except TimeoutError:
  print("Page took too long to load and timed out..")
finally:
  driver.quit()
```

SeleniumWaitUtility.py

```
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
class SeleniumWaitUtility:
  def __init__(self, driver, timeout):
    self.mywait = WebDriverWait(driver, timeout)
                                    Waits for a given element to be visible
  def wait_for_element_visible(self, locator: By):
    return self.mywait.until(EC.visibility_of_element_located(locator))
                                    Waits for a given element to be clickable
  def wait_for_element_clickable(self, locator: By):
    return self.mywait.until(EC.element_to_be_clickable(locator))
                                    Waits for a given element to be present
  def wait_for_element_presence(self, locator: By):
    return self.mywait.until(EC.presence_of_element_located(locator))
```

Test.py

```
from SeleniumWaitUtility import SeleniumWaitUtility
from selenium import webdriver
from selenium.webdriver.common.by import By
opt = webdriver.ChromeOptions()
opt.add_experimental_option("detach",True)
driver = webdriver.Chrome(options=opt)
driver.get("https://demowebshop.tricentis.com/")
driver.maximize_window()
wait_util = SeleniumWaitUtility(driver, 10)
logo = wait_util.wait_for_element_visible((By.XPATH, "//img[@alt='Tricentis Demo Web Shop']"))
print("Logo visibility:", logo.is_displayed())
search_box = wait_util.wait_for_element_visible((By.XPATH, "//input[@id='small-searchterms']"))
search_box.send_keys("Smartphone")
search_btn = wait_util.wait_for_element_clickable((By.XPATH, "//input[@value='Search']"))
search_btn.click()
phone_image = wait_util.wait_for_element_presence((By.XPATH, "//img[@title='Show details for
Smartphone']"))
print("Smartphone presence:", phone_image.is_displayed())
driver.quit()
```