import time

import unittest

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.support import expected\_conditions as EC

from selenium.webdriver.support.wait import WebDriverWait

"""

########### Libraria unittest ###########

Libraria unittest ofera suport pentru crearea de teste rulabile direct in interiorul clasei.

Se implementeaza prin mostenirea clasei TestCase din libraria unittest

Orice clasa de teste trebuie sa mosteneasca clasa TestCase si sa aiba urmatoarele particularitati:

1. metoda setUp() -> toate activitatile care trebuie sa fie executate inainte de ORICE TEST din clasa respectiva

2. metoda teardown() -> toate activitatile care trebuie sa fie executate dupa de ORICE TEST din clasa respectiva

3. toate metodele de test trebuie sa aiba prefixul test\_

\* = despachetare tuplu

driver.find\_elements(\*BUTTON\_JS\_ALERT\_SIMPLE) <=> driver.find\_elements(By.CSS\_SELECTOR, "[onclick='jsAlert()']")

"""

class Test(unittest.TestCase):

driver = None

LOGIN\_LINK = "https://the-internet.herokuapp.com/login"

BUTTON\_LOGIN = (By.CLASS\_NAME, "fa-sign-in")

MESSAGE\_SUCCESS = (By.CLASS\_NAME, "alert-success")

USERNAME = (By.ID, "username")

PASSWORD = (By.ID, "password")

SECURE\_AREA = (By.XPATH, "//div[@class='flash success']")

# suprascriem metoda setUp care va rula inainte de fiecare test

def setUp(self):

# print(f"Se executa ce este in setUp() pentru {self.\_testMethodName}")

self.driver = webdriver.Chrome()

self.driver.get(self.LOGIN\_LINK)

self.driver.maximize\_window()

time.sleep(1)

# suprascriem metoda tearDown care va rula dupa fiecare test

def tearDown(self):

# print(f"Se executa ce este in tearDown() pentru {self.\_testMethodName}\n")

self.driver.quit()

# TEST 1

# - Verifica daca URL-ul paginii este corect

def test\_url(self):

actual\_url = self.driver.current\_url

# assert expected\_url == actual\_url, f"Unexpected URL, expected {expected\_url}, but found {actual\_url}"

# Varianta unittest pentru assert de egalitate

self.assertEqual(self.LOGIN\_LINK, actual\_url, "Unexpected URL")

# TEST 2

# - Verifica daca titlul paginii apare corect

def test\_title(self):

expected\_title = "The Internet"

actual\_title = self.driver.title

# assert expected\_title == actual\_title, f"Unexpected title, expected {expected\_title}, but found {actual\_title}"

self.assertEqual(expected\_title, actual\_title, "Unexpected title")

# TEST 3

# - Verifica daca textul de pe elementul xpath=//h2 e corect

def test\_h2\_text(self):

text\_h2 = self.driver.find\_element(By.XPATH, "//h2").text

expected\_text = "Login Page"

# assert expected\_text == text\_h2, f"Invalid h2 text, expected {expected\_text}, but found {text\_h2}"

# Observam ca folosind assertEqual, nu mai trebuie scrisa partea cu 'expected x but

# found y' deoarece se specifica implicit prin aceasta metoda

self.assertEqual(expected\_text, text\_h2, "Textul h2 este incorect")

# TEST 4

# - Verifica daca butonul de login este displayed

def test\_buton\_login\_display(self):

login\_button = self.driver.find\_element(\*self.BUTTON\_LOGIN)

assert login\_button.is\_displayed(), "Butonul de Login nu este afisat"

# TEST 5

# - Verifică dacă atributul href al linkului ‘Elemental Selenium’ e corect

def test\_atribut\_href(self):

expected\_href = "http://elementalselenium.com/"

# luam atributul de tip "href" de pe acel element de tip a

actual\_href = self.driver.find\_element(By.XPATH, "//a[text()='Elemental Selenium']").get\_attribute("href")

self.assertEqual(expected\_href, actual\_href, "Link-ul 'href' este incorect")

# TEST 6

# - Lasă goale user și pass;

# - Click login;

# - Verifică dacă eroarea e displayed

def test\_blank\_login(self):

# Facem click pe butonul de login fara sa completam vreun camp

self.driver.find\_element(\*self.BUTTON\_LOGIN).click()

eroare = self.driver.find\_element(By.ID, "flash")

assert eroare.is\_displayed(), "Eroarea nu este afisata dupa logare fara Username/Password"

# TEST 7

# - Completează cu user și pass invalide;

# - Click login;

# - Verifică dacă mesajul de pe eroare e corect;

# - Este și un x pus acolo extra așa că vom folosi soluția de mai jos:

# expected = 'Your username is invalid!'

# self.assertTrue(expected in actual, 'Error message text is incorrect')

def test\_invalid\_login(self):

username = self.driver.find\_element(\*self.USERNAME)

username.send\_keys("wrong\_user")

password = self.driver.find\_element(\*self.PASSWORD)

password.send\_keys("wrong\_password")

self.driver.find\_element(\*self.BUTTON\_LOGIN).click()

expected\_message = "Your username is invalid!"

actual\_message = self.driver.find\_element(By.ID, "flash").text # luam textul; am testat si ia inclusiv \*

# Assert pentru a verifica daca o expresie este adevarata

# Metoda assertTrue() primeste 2 parametri:

# \* primul param - expresia de evaluat

# \* al doilea param - (optional) mesajul de eroare afisat

self.assertTrue(expected\_message in actual\_message, "Mesajul de eroare nu este corect")

# metoda ajutatoare - ea nu va fi rulata ca test pentru ca nu are prefixul test\_

# metoda are 2 parametri:

# 1 - element\_locator: locatorul elementului dupa care asteptam sa apara

# 2 - seconds\_to\_wait: numarul maxim de secunde de asteptare pentru ca elementul sa apara

def wait\_for\_element\_to\_be\_present(self, element\_locator, seconds\_to\_wait):

wait = WebDriverWait(self.driver, seconds\_to\_wait)

return wait.until(EC.presence\_of\_element\_located(element\_locator))

# metoda ajutatoare - asteapta ca un element sa devina ABSEENT in HTML (absent != invizibil)

# metoda are 2 parametri:

# 1 - element\_locator: locatorul elementului dupa care asteptam sa dispara

# 2 - seconds\_to\_wait: numarul maxim de secunde de asteptare pentru ca elementul sa dispare

# Observate: cu EC.none\_of() se neaga conditia din EC (EC = expected\_condition)

def wait\_for\_element\_to\_disappear(self, element, timp):

wait = WebDriverWait(self.driver, timp)

return wait.until(EC.none\_of(EC.presence\_of\_element\_located(element)))

# metoda ajutatoare - returneaza TRUE daca elementul este PREZENT (prezent nu inseamna ca e si vizibil neaparat)

# - metoda are ca parametru variabila element\_locator: locatorul elementului dupa care asteptam sa dispara

# - folosind driver.find\_elements() putem sa ne dam seama daca un element este sau nu prezent

# - driver.find\_elements() nu da eroare daca nu gaseste un element dupa un locator dat, ci returneaza o lista goala

# - daca lista e goala, inseamna ca nu e elementul prezent [len(lista) = 0 deci return FALSE]

# - daca lista nu e goala, inseamna ca avem cel putin un element gasit [len(lista) > 0 deci return TRUE]

def is\_element\_present(self, locator):

return len(self.driver.find\_elements(\*locator)) > 0 # daca nu gaseste nimic, returneaza o lista goala

# TEST 8

# - Lasă goale user și pass;

# - Click login;

# - Apasă x la eroare;

# - Verifică dacă eroarea a dispărut

def test\_eroare\_message\_disappears\_on\_click(self):

self.driver.find\_element(\*self.BUTTON\_LOGIN).click()

self.driver.find\_element(By.CLASS\_NAME, "close").click()

self.wait\_for\_element\_to\_disappear((By.ID, "flash"),

3) # pt ca nu dispare din prima, dureaza putin sa dispara dupa ce dai x

# self.assertFalse(self.is\_element\_present((By.ID, "flash")), "Mesajul de eroare nu a disparut")

self.assertTrue(not self.is\_element\_present((By.ID, "flash")), "Mesajul de eroare nu a disparut")

# TEST 9 - Ia ca o listă toate //label; - Verifică textul ca textul de pe ele să fie cel așteptat (Username și Password)

# - Aici e ok să avem 2 asserturi

def test\_text\_asteptat(self):

lista\_label = self.driver.find\_elements(By.XPATH, "//label")

# print(lista\_label)

expected\_label\_text\_1 = "Username"

actual\_label\_text\_1 = lista\_label[0].text

expected\_label\_text\_2 = "Password"

actual\_label\_text\_2 = lista\_label[1].text

self.assertEqual(expected\_label\_text\_1, actual\_label\_text\_1, "Textul de pe label nu coincide")

self.assertEqual(expected\_label\_text\_2, actual\_label\_text\_2, "Textul de pe label nu coincide")

# TEST 10

# - Completează cu user și pass valide;

# - Click login;

# - Verifică ca noul url CONTINE stringul /secure;

# - Folosește un explicit wait pentru elementul cu clasa ’flash succes’

# - Verifică dacă elementul cu clasa=’flash succes’ este displayed;

# - Verifică dacă mesajul de pe acest element CONȚINE textul ‘secure area!’

def test\_valid\_login(self):

username = self.driver.find\_element(\*self.USERNAME)

username.send\_keys("tomsmith")

password = self.driver.find\_element(\*self.PASSWORD)

password.send\_keys("SuperSecretPassword!")

self.driver.find\_element(\*self.BUTTON\_LOGIN).click()

actual\_url = self.driver.current\_url

self.assertIn("/secure", actual\_url), "Noul URL nu contine '/secure'"

element\_flash\_success = self.wait\_for\_element\_to\_be\_present(self.SECURE\_AREA, 3)

self.assertTrue(element\_flash\_success.is\_displayed(), "Elementul nu este afisat")

mesaj = self.driver.find\_element(\*self.SECURE\_AREA).text

self.assertIn("secure area", mesaj, "Mesajul nu contine textul 'secure area!'")

# TEST 11

# - Completează cu user și pass valide;

# - Click login;

# - Click logout;

# - Verifică dacă ai ajuns pe https://the-internet.herokuapp.com/login

def test\_logout(self):

self.test\_valid\_login()

logout\_button = self.driver.find\_element(By.CLASS\_NAME, "icon-signout")

logout\_button.click()

actual\_url = self.driver.current\_url

self.assertEqual(self.LOGIN\_LINK, actual\_url, "Linkurile nu sunt la fel")

# TEST 12 - Completează user tomsmith;

# - Găsește elementul //h4;

# - Ia textul de pe el și fă split după spațiu.

# - Consideră fiecare cuvânt ca o potențială parolă;

# - Folosește o structură iterativă prin care să introduci rând pe rând parolele și să apeși pe login;

# - La final trebuie afisam in consola fie ‘Nu am reușit să găsesc parola’, fie ‘Parola secretă este... [parola]’

def test\_brute\_force(self):

element\_h4 = self.driver.find\_element(By.XPATH, "//h4").text

potentiale\_parole = element\_h4.split(" ")

for parola in potentiale\_parole:

username = self.driver.find\_element(\*self.USERNAME)

username.send\_keys("tomsmith")

password = self.driver.find\_element(\*self.PASSWORD)

password.send\_keys(str(parola))

self.driver.find\_element(\*self.BUTTON\_LOGIN).click()

mesaj\_eroare = (By.XPATH, "//div[@class='flash error']")

if self.is\_element\_present(mesaj\_eroare):

print("Inca nu am gasit parola")

else:

print(f"Parola secreta este '{parola}'")

break

# Excludem testul in timpul rularii cu ajutorul decoratorului @unittest.skip

@unittest.skip

def test\_picat(self):

assert False, "Test picat"

# def test\_login1(self):

# pass

#

# def test\_login2(self):

# self.login3()

# pass

#

# def login3(self):

# print("Se executa codul din login3()")

# pass