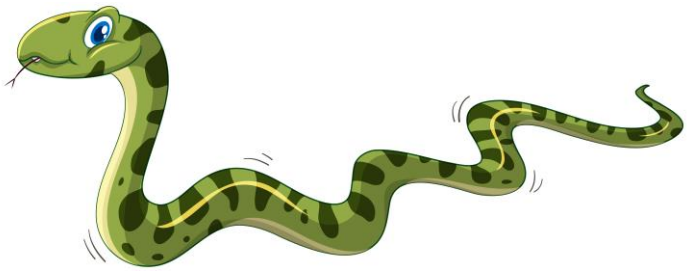


Zródło: Obraz autorstwa brgfx na Freepik

PROJEKT TESTÓW FUNKCJONALNYCH

Z WYKORZYSTANIEM BIBLIOTEKI SELENIUM

ZAPISANY:

https://github.com/przemo933/selenium_project

SPIS TREŚCI:

1. INFORMACJE O PROJEKCIE

Celem niniejszego projektu było wykonanie funkcjonalnych testów z wykorzystaniem biblioteki Selenium.

Projekt został napisany przy użyciu Python 3.8.

Projekt składa się z ośmiu klas z testami oraz jedną klasą bazową z której dziedziczą pozostałe testy (test_case):

1. test_checkbox,
2. test_file_download,
3. test_file_upload,
4. test_login,
5. test_login_store,
6. test_shop_cart,
7. test_shop_checkout,
8. test_broken_image.

Strony internetowe, które są testowane w niniejszym projekcie to:

1. http://the-internet.herokuapp.com/
2. https://admin-demo.nopcommerce.com/admin/
3. https://practice.automationbro.com/shop
4. https://demo.automationtesting.in/FileDownload.html

2. PRZEPROWADZONE TESTY (przykłady)

Poniżej przedstawiono przykładowe testy znajdujące się w projekcie (wszystkie testy zapisane są na GitHubie).

TEST_LOGIN

Strona internetowa na której wykonano testy: http://the-internet.herokuapp.com/

W teście wprowadzono m.in. funkcję testującą logowanie z prawidłowymi danymi:

```
class TestLogin(TestCase):
    VALID_LOGIN = "tomsmith"
    VALID_PASSWORD = "SuperSecretPassword!"
    INVALID_LOGIN = "przemek"
    INVALID_PASSWORD = "haslo"

    # funkcja testujaca logowanie z prawidłowymi danymi
    Przemysław Sobieraj
    def test_valid_login(self):
        self.driver.get("http://the-internet.herokuapp.com/login")
        try:
            username_field = self.driver.find_element(By.ID, "username")
            username_field.click()
            username_field.send_keys(self.VALID_LOGIN)
        except Exception:
            self.fail("Field not found!")

        try:
            password_field = self.driver.find_element(By.ID, "password")
            password_field.click()
            password_field.send_keys(self.VALID_PASSWORD)
        except Exception:
            self.fail("Field not found!")

        try:
            login_button = self.driver.find_element(By.XPATH, "/html/body/div[2]/div/div/form/button/i")
        except Exception:
            self.fail("Button not found!")
        login_button.click()
```

TEST_LOGIN_STORE

Strona internetowa na której wykonano testy: https://admin-demo.nopcommerce.com/

W teście wprowadzono m.in. funkcję testującą logowanie z nieprawidłowymi danymi oraz sprawdzającą czy na stronie wyświetla się wiadomość o błędnym logowaniu:

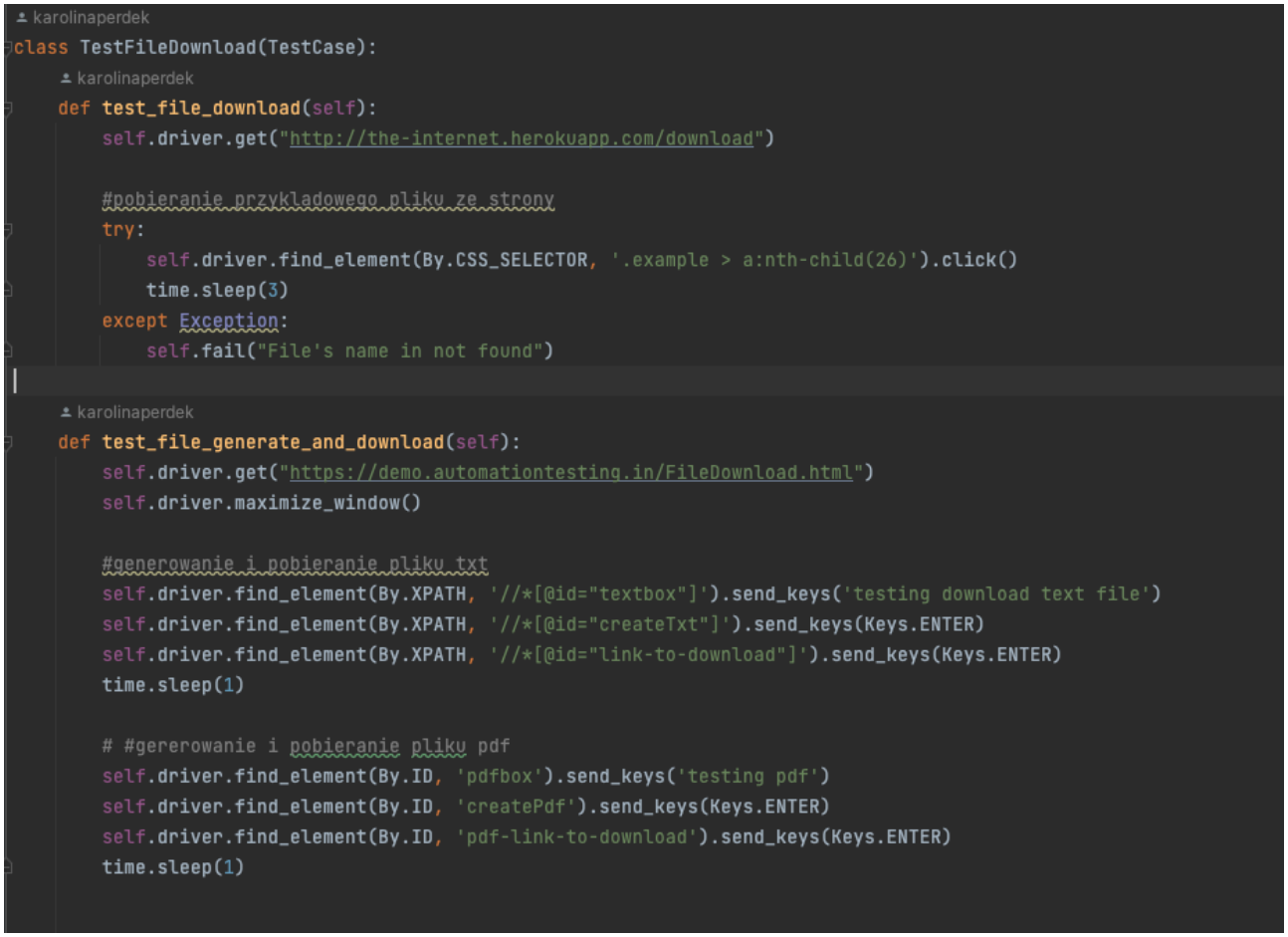
```
# funkcja testujaca logowanie z nieprawidłowymi danymi
Przemysław Sobieraj *
def test_invalid_login(self):
    try:
        self.driver.get("https://admin-demo.nopcommerce.com/login")
        username_field = self.driver.find_element(By.ID, "Email")
        username_field.clear()
        username_field.send_keys(self.INVALID_LOGIN)
        password_field = self.driver.find_element(By.ID, "Password")
        password_field.clear()
        password_field.send_keys(self.INVALID_PASSWORD)
        login_button = self.driver.find_element(By.CSS_SELECTOR, ".button-1")
        login_button.click()
    except Exception:
        self.fail("Login with valid login details failed!")

    # sprawdza czy na stronie wyświetliła się wiadomość o błędnym logowaniu
    self.driver.find_element(By.CSS_SELECTOR,
                             "html.html-login-page body div.master-wrapper-page div.master-wrapper-"
                             "content div.master-column-wrapper div.center-1 div.page.login-page div"
                             ".page-body div.customer-blocks div.returning-wrapper.fieldset form div"
                             ".message-error.validation-summary-errors")
```

TEST_FILE_DOWNLOAD

Strony internetowe na których wykonano testy: http://the-internet.herokuapp.com/ oraz https://demo.automationtesting.in/FileDownload.html"

W teście sprawdzono możliwość pobierania przykładowego pliku ze strony oraz generowania oraz pobierania plików w formacie .txt oraz .pdf:



```
karolinaperdek
class TestFileDownload(TestCase):
    karolinaperdek
    def test_file_download(self):
        self.driver.get("http://the-internet.herokuapp.com/download")

        #pobieranie przykladowego pliku ze strony
        try:
            self.driver.find_element(By.CSS_SELECTOR, '.example > a:nth-child(26)').click()
            time.sleep(3)
        except Exception:
            self.fail("File's name in not found")

    karolinaperdek
    def test_file_generate_and_download(self):
        self.driver.get("https://demo.automationtesting.in/FileDownload.html")
        self.driver.maximize_window()

        #generowanie i pobieranie pliku txt
        self.driver.find_element(By.XPATH, '//*[@id="textbox"]').send_keys('testing download text file')
        self.driver.find_element(By.XPATH, '//*[@id="createTxt"]').send_keys(Keys.ENTER)
        self.driver.find_element(By.XPATH, '//*[@id="link-to-download"]').send_keys(Keys.ENTER)
        time.sleep(1)

        # #gererowanie i pobieranie pliku pdf
        self.driver.find_element(By.ID, 'pdfbox').send_keys('testing pdf')
        self.driver.find_element(By.ID, 'createPdf').send_keys(Keys.ENTER)
        self.driver.find_element(By.ID, 'pdf-link-to-download').send_keys(Keys.ENTER)
        time.sleep(1)
```

TEST_SHOP_CART

Strona internetowa na której wykonano testy: https://practice.automationbro.com/shop

W teście wprowadzono m.in. funkcję testującą wyszukiwanie produktu, dodanie oraz usuwanie z koszyka:

```
class TestShopCart(TestCase):

    # funkcja testujaca wyszukiwanie produktu, dodanie do koszyka i usuwanie z koszyka
    karolinaperdek
    def test_shop_search_add_remove_cart(self):
        self.driver.get("https://practice.automationbro.com/shop")
        search_field = self.driver.find_element(By.ID, "woocommerce-product-search-field-0")
        search_field.send_keys("watch")
        add_cart_1 = self.driver.find_element(By.XPATH, "/html/body/div[1]/main/div/div/div/ul/li[1]/a[2]")
        add_cart_1.click()
        add_cart_2 = self.driver.find_element(By.XPATH, "/html/body/div[1]/main/div/div/div/ul/li[2]/a[2]")
        add_cart_2.click()
        sleep(2)
        cart = self.driver.find_element(By.XPATH, "/html/body/div[1]/header/div/div/div[2]/div[1]/ul/li[2]/a/i")
        cart.click()
```

TEST_SHOP_CHECKOUT

Strona internetowa na której wykonano testy: https://practice.automationbro.com/shop

W teście wprowadzono funkcję testującą dodawanie produktu do koszyka i zrealizowanie zamówienia testowymi danymi:

```
Przemysław Sobieraj
class TestShopCheckout(TestCase):
    FIRST_NAME = 'Jan'
    LAST_NAME = 'Kowalski'
    COUNTRY = 'Poland'
    STREET_ADDRESS = 'Klonowa 5'
    POSTCODE = '23-232'
    CITY = 'Wroclaw'
    PHONE = '787000999'
    EMAIL = 'test'.join(str(randint(1, 99999))) + '@test.pl'
    USERNAME = 'testing'.join(str(randint(1, 99999)))
    PASSWORD = 'Test123'

    # funkcja testujaca dodanie produktu do koszyka i zrealizowania zamówienia testowymi danymi
    Przemysław Sobieraj
    def test_shop_checkout(self):
        self.driver.get("https://practice.automationbro.com/shop")
        self.driver.maximize_window()
        item = self.driver.find_element(By.XPATH, "/html/body/div[1]/main/div/div/div/ul/li[1]/a[2]")
        item.click()
        sleep(2)
        cart = self.driver.find_element(By.XPATH, "/html/body/div[1]/header/div/div/div[2]/div[1]/ul/li[2]/a/i")
        cart.click()
        proceed_xpath = "/html/body/div[1]/main/div/div/div/article/div/div[2]/div[2]/div/div/a"
        proceed = WebDriverWait(self.driver, 10).until(presence_of_element_located((By.XPATH, proceed_xpath)))
        proceed.click()

        self.driver.find_element(By.ID, "billing_first_name").send_keys(self.FIRST_NAME)
        self.driver.find_element(By.ID, "billing_last_name").send_keys(self.LAST_NAME)
        self.driver.find_element(By.ID, "select2-billing_country-container").click()
        self.driver.find_element(By.XPATH, "/html/body/span[2]/span/span[1]/input").send_keys(self.COUNTRY)
        self.driver.find_element(By.XPATH, "/html/body/span[2]/span/span[1]/input").send_keys(Keys.ENTER)
        self.driver.find_element(By.ID, "billing_address_1").send_keys(self.STREET_ADDRESS)
        self.driver.find_element(By.ID, "billing_postcode").send_keys(self.POSTCODE)
        self.driver.find_element(By.ID, "billing_city").send_keys(self.CITY)
        self.driver.find_element(By.ID, "billing_phone").send_keys(self.PHONE)
        self.driver.find_element(By.ID, "billing_email").send_keys(self.EMAIL)
        self.driver.find_element(By.ID, "account_username").send_keys(self.USERNAME)
        self.driver.find_element(By.ID, "account_password").send_keys(self.PASSWORD)
        self.driver.find_element(By.ID, "place_order").click()
        sleep(6)
        self.driver.save_screenshot("image.png")
```

TEST_BROKEN_IMAGE

Strona internetowa na której wykonano testy: http://the-internet.herokuapp.com/

W teście sprawdzono ile jest uszkodzonych obrazków na stronie:

```python
from test_case import TestCase
from selenium.webdriver.common.by import By
from requests import get


# test sprawdza ile jest uszkodzonych obrazków na stronie
 Przemysław Sobieraj
class TestBrokenImage(TestCase):
    Przemysław Sobieraj
    def test_broken_image(self):
        self.driver.get("http://the-internet.herokuapp.com/broken_images")
        images = self.driver.find_elements(By.TAG_NAME, "img")
        counter = 0
        for image in images:
            response = get(image.get_attribute('src'), stream=True)
            if response.status_code != 200:
                print(image.get_attribute('outerHTML') + " is broken.")
                counter += 1

        print('The page has ' + str(counter) + ' broken images')
```

3. WYKONAWCY

Projekt został wykonany przez Przemysława Sobieraja oraz Karolinę Perdek.

Kontakt: przemo.sob@gmail.com / perdekkarolina@gmail.com