przygotuj bibliotękę w pythonie bazując na wzorcu projektowym page object pattern dla strony https://demoqa.com/ dla podstrony elements Text Box, Check Box, Web Tables, a wszystkie lokatory trzymaj w osobnym folderze locators. W Web Tables przyogtuj możliwość usuwania danych tabeli i ich edytowania

Oto implementacja biblioteki w Pythonie zgodnie z wzorcem **Page Object Pattern** dla strony demoqa.com. Obsługiwane są sekcje **Text Box**, **Check Box** oraz **Web Tables**, a wszystkie lokatory przechowywane są w osobnym folderze locators. Sekcja **Web Tables** obejmuje funkcjonalności usuwania oraz edytowania wierszy tabeli.

**Struktura projektu**

bash

Skopiuj kod

/demoqa

/locators

\_\_init\_\_.py

elements\_locators.py

text\_box\_locators.py

check\_box\_locators.py

web\_tables\_locators.py

/pages

\_\_init\_\_.py

base\_page.py

elements\_page.py

text\_box\_page.py

check\_box\_page.py

web\_tables\_page.py

/tests

test\_text\_box.py

test\_check\_box.py

test\_web\_tables.py

requirements.txt

**1. Lokatory**

**Plik: elements\_locators.py**

python

Skopiuj kod

class ElementsLocators:

TEXT\_BOX\_MENU = "id:item-0"

CHECK\_BOX\_MENU = "id:item-1"

WEB\_TABLES\_MENU = "id:item-3"

**Plik: text\_box\_locators.py**

python

Skopiuj kod

class TextBoxLocators:

FULL\_NAME\_FIELD = "id:userName"

EMAIL\_FIELD = "id:userEmail"

CURRENT\_ADDRESS\_FIELD = "id:currentAddress"

PERMANENT\_ADDRESS\_FIELD = "id:permanentAddress"

SUBMIT\_BUTTON = "id:submit"

OUTPUT\_SECTION = "id:output"

**Plik: check\_box\_locators.py**

python

Skopiuj kod

class CheckBoxLocators:

TOGGLE\_HOME = "css:.rct-icon.rct-icon-expand-close"

CHECK\_DOCUMENTS = "xpath://label[@for='tree-node-documents']//span[@class='rct-checkbox']"

OUTPUT\_SECTION = "id:result"

**Plik: web\_tables\_locators.py**

python

Skopiuj kod

class WebTablesLocators:

ADD\_BUTTON = "id:addNewRecordButton"

EDIT\_BUTTON = "xpath://span[text()='{}']/following-sibling::span[@title='Edit']"

DELETE\_BUTTON = "xpath://span[text()='{}']/following-sibling::span[@title='Delete']"

ROWS = "css:.rt-tr-group"

FIRST\_NAME\_FIELD = "id:firstName"

LAST\_NAME\_FIELD = "id:lastName"

EMAIL\_FIELD = "id:userEmail"

AGE\_FIELD = "id:age"

SALARY\_FIELD = "id:salary"

DEPARTMENT\_FIELD = "id:department"

SUBMIT\_BUTTON = "id:submit"

**2. Biblioteka stron**

**Plik: base\_page.py**

python

Skopiuj kod

from selenium.webdriver.common.by import By

from selenium.webdriver.support.ui import WebDriverWait

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

class BasePage:

def \_\_init\_\_(self, driver):

self.driver = driver

self.wait = WebDriverWait(driver, 10)

def open\_url(self, url):

"""Otwiera URL."""

self.driver.get(url)

def click\_element(self, locator):

"""Kliknięcie elementu."""

element = self.wait.until(EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, locator)))

element.click()

def enter\_text(self, locator, text):

"""Wprowadzenie tekstu."""

element = self.wait.until(EC.visibility\_of\_element\_located((By.CSS\_SELECTOR, locator)))

element.clear()

element.send\_keys(text)

def get\_element\_text(self, locator):

"""Pobiera tekst elementu."""

element = self.wait.until(EC.visibility\_of\_element\_located((By.CSS\_SELECTOR, locator)))

return element.text

def get\_dynamic\_locator(self, template, \*args):

"""Zwraca dynamicznie wygenerowany lokator."""

return template.format(\*args)

**Plik: elements\_page.py**

python

Skopiuj kod

from demoqa.pages.base\_page import BasePage

from demoqa.locators.elements\_locators import ElementsLocators

class ElementsPage(BasePage):

def navigate\_to\_text\_box(self):

"""Przejście do sekcji Text Box."""

self.click\_element(ElementsLocators.TEXT\_BOX\_MENU)

def navigate\_to\_check\_box(self):

"""Przejście do sekcji Check Box."""

self.click\_element(ElementsLocators.CHECK\_BOX\_MENU)

def navigate\_to\_web\_tables(self):

"""Przejście do sekcji Web Tables."""

self.click\_element(ElementsLocators.WEB\_TABLES\_MENU)

**Plik: text\_box\_page.py**

python

Skopiuj kod

from demoqa.pages.base\_page import BasePage

from demoqa.locators.text\_box\_locators import TextBoxLocators

class TextBoxPage(BasePage):

def fill\_text\_box\_form(self, name, email, current\_address, permanent\_address):

"""Wypełnia formularz Text Box."""

self.enter\_text(TextBoxLocators.FULL\_NAME\_FIELD, name)

self.enter\_text(TextBoxLocators.EMAIL\_FIELD, email)

self.enter\_text(TextBoxLocators.CURRENT\_ADDRESS\_FIELD, current\_address)

self.enter\_text(TextBoxLocators.PERMANENT\_ADDRESS\_FIELD, permanent\_address)

self.click\_element(TextBoxLocators.SUBMIT\_BUTTON)

def get\_output\_text(self):

"""Pobiera tekst z wyników."""

return self.get\_element\_text(TextBoxLocators.OUTPUT\_SECTION)

**Plik: check\_box\_page.py**

python

Skopiuj kod

from demoqa.pages.base\_page import BasePage

from demoqa.locators.check\_box\_locators import CheckBoxLocators

class CheckBoxPage(BasePage):

def expand\_home(self):

"""Rozwija katalog główny."""

self.click\_element(CheckBoxLocators.TOGGLE\_HOME)

def select\_documents(self):

"""Zaznacza opcję Documents."""

self.click\_element(CheckBoxLocators.CHECK\_DOCUMENTS)

def get\_selected\_items(self):

"""Zwraca wybrane elementy."""

return self.get\_element\_text(CheckBoxLocators.OUTPUT\_SECTION)

**Plik: web\_tables\_page.py**

python

Skopiuj kod

from demoqa.pages.base\_page import BasePage

from demoqa.locators.web\_tables\_locators import WebTablesLocators

class WebTablesPage(BasePage):

def delete\_row(self, name):

"""Usuwa wiersz na podstawie imienia."""

delete\_button = self.get\_dynamic\_locator(WebTablesLocators.DELETE\_BUTTON, name)

self.click\_element(delete\_button)

def edit\_row(self, name, new\_data):

"""Edytuje wiersz na podstawie imienia."""

edit\_button = self.get\_dynamic\_locator(WebTablesLocators.EDIT\_BUTTON, name)

self.click\_element(edit\_button)

# Wypełnianie formularza

self.enter\_text(WebTablesLocators.FIRST\_NAME\_FIELD, new\_data["first\_name"])

self.enter\_text(WebTablesLocators.LAST\_NAME\_FIELD, new\_data["last\_name"])

self.enter\_text(WebTablesLocators.EMAIL\_FIELD, new\_data["email"])

self.enter\_text(WebTablesLocators.AGE\_FIELD, new\_data["age"])

self.enter\_text(WebTablesLocators.SALARY\_FIELD, new\_data["salary"])

self.enter\_text(WebTablesLocators.DEPARTMENT\_FIELD, new\_data["department"])

self.click\_element(WebTablesLocators.SUBMIT\_BUTTON)

def get\_rows\_count(self):

"""Zwraca liczbę wierszy w tabeli."""

rows = self.driver.find\_elements\_by\_css\_selector(WebTablesLocators.ROWS)

return len(rows)

**3. Testy**

**Test Web Tables: test\_web\_tables.py**

python

Skopiuj kod

import unittest

from selenium import webdriver

from demoqa.pages.elements\_page import ElementsPage

from demoqa.pages.web\_tables\_page import WebTablesPage

class TestWebTables(unittest.TestCase):

@classmethod

def setUpClass(cls):

cls.driver = webdriver.Chrome()

cls.driver.maximize\_window()

cls.driver.get("https://demoqa.com/elements")

cls.elements\_page = ElementsPage(cls.driver)

cls.web\_tables\_page = WebTablesPage(cls.driver)

@classmethod

def tearDownClass(cls):

cls.driver.quit()

def test\_delete\_row(self):

self.elements\_page.navigate\_to\_web\_tables()

initial\_count = self.web\_tables\_page.get\_rows\_count()

self.web\_tables\_page.delete\_row("Cierra")

final\_count = self.web\_tables\_page.get\_rows\_count()

self.assertEqual(initial\_count - 1, final\_count)

def test\_edit\_row(self):

self.elements\_page.navigate\_to\_web\_tables()

self.web\_tables\_page.edit\_row("Cierra", {

"first\_name": "UpdatedName",

"last\_name": "UpdatedLast",

"email": "updated@example.com",

"age": "30",

"salary": "5000",

"department": "HR"

})

# Możesz dodać dodatkowe sprawdzenia w zależności od wymagań.

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()

**4. Instalacja zależności**

Plik requirements.txt:

makefile

Skopiuj kod

selenium==4.10.0

Uruchom instalację:

bash

Skopiuj kod

pip install -r requirements.txt

Uruchom testy:

bash

Skopiuj kod

python -m unittest tests/test\_web\_tables.py