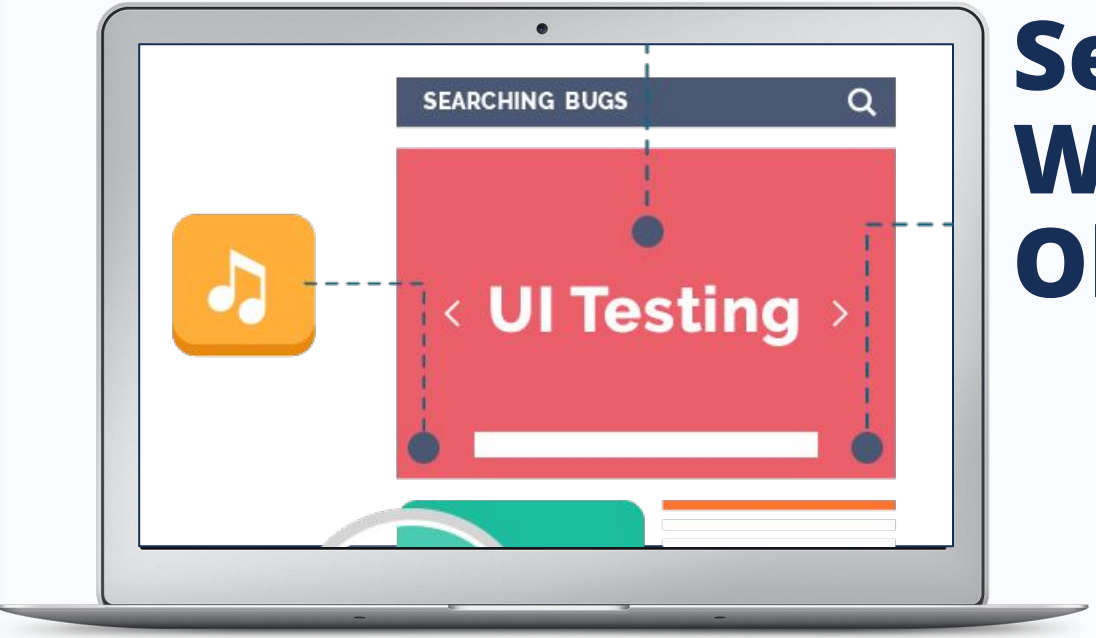
PROMETHEUS



Selenium WebDriver. Page Object Model

Бутенко Сергій

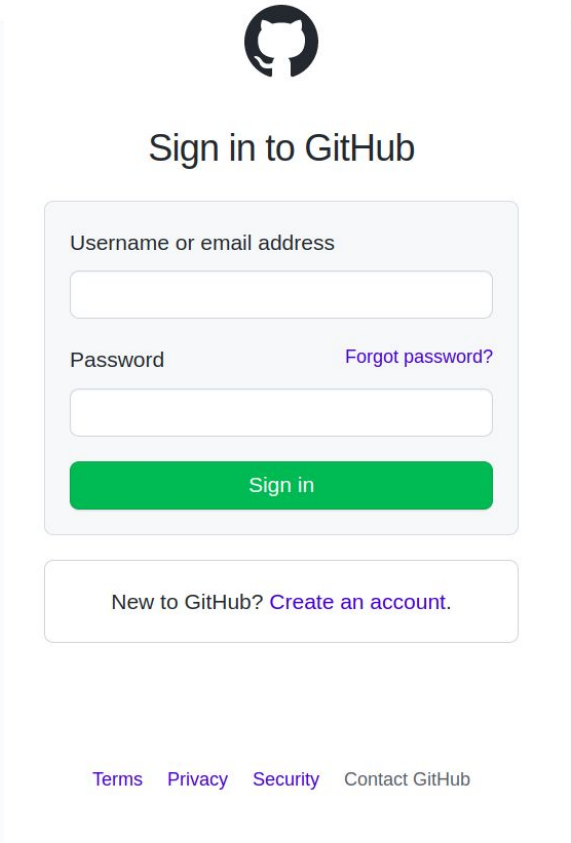
План лекції

- Визначення
- Переваги використання

Page Object Model

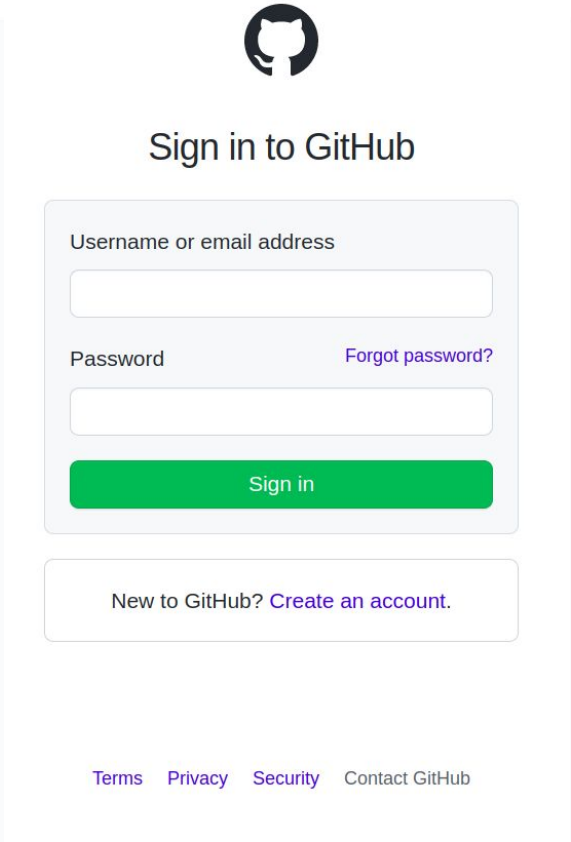
```python
@pytest.mark.ui
def test_check_incorrect_username():
    # Створення об'єкту для керування бразуером
    driver = webdriver.Chrome(
        service=Service(r"/home/sbutenko/repos/LnD/Become QA Auto/" + "chromedriver")
        )

    # відкриваємо сторінку https://github.com/login
    driver.get("https://github.com/login")

    # Знаходимо поле, в яке будемо вводити неправильне ім'я користувача або поштову адресу
    login_elem = driver.find_element(By.ID, "login_field")

    # Вводимо неправильне ім'я користувача або поштову адрІесу
    login_elem.send_keys("sergiibutenko@mistakeinemail.com")

    # Знаходимо поле, в яке будемо вводити неправильний пароль
    pass_elem = driver.find_element(By.ID, "password")

    # Вводимо неправильний пароль
    pass_elem.send_keys("wrong password")

    # Знаходимо кнопку sign in
    btn_elem = driver.find_element(By.NAME, "commit")

    # Емулюємо клік лівою кнопкою мишки
    btn_elem.click()

    # Перевіряємо, що назва сторінки така, яку ми очікуємо
    assert driver.title == "Sign in to GitHub · GitHub"

    # Закриваємо браузер
    driver.close()
```

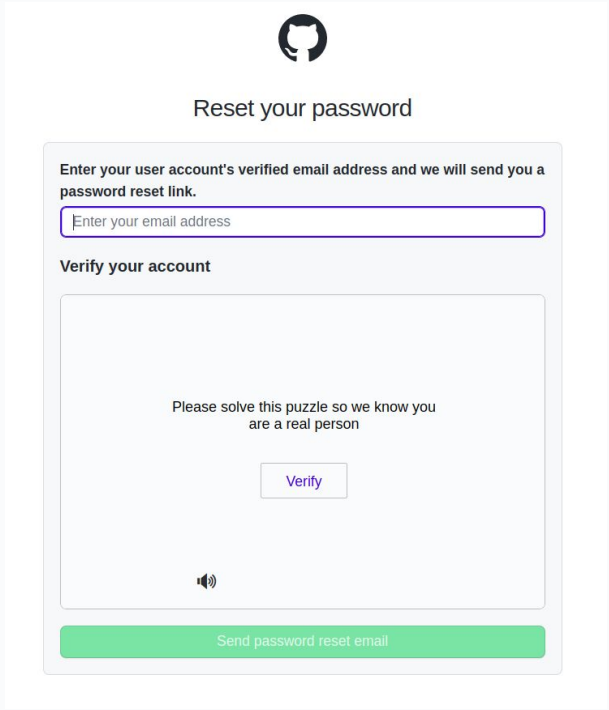
Page Object Model



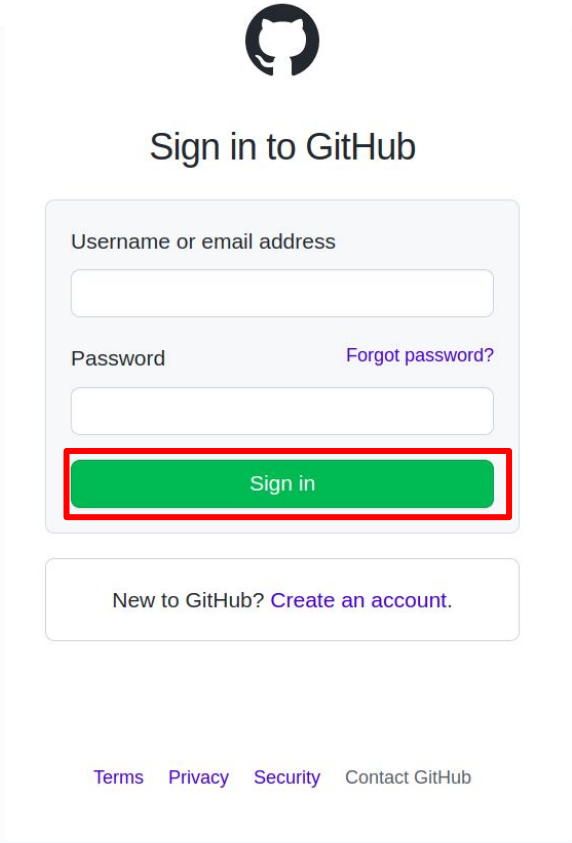
Page Object Model - це архітектурний підхід, при якому в коді для кожної сторінки, яку ви використовуєте, в тестах створюється окремий клас, що описує в собі логіку та технічний опис сторінки.

Page Object Model

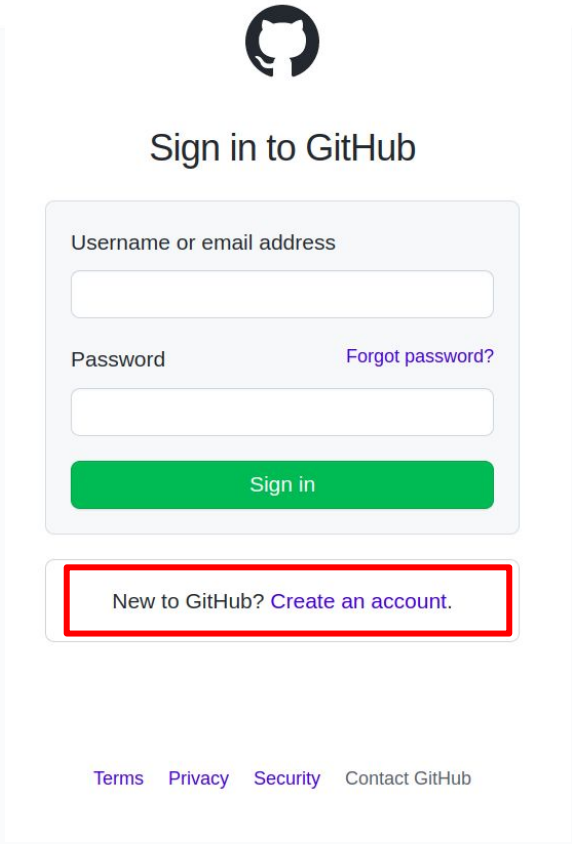




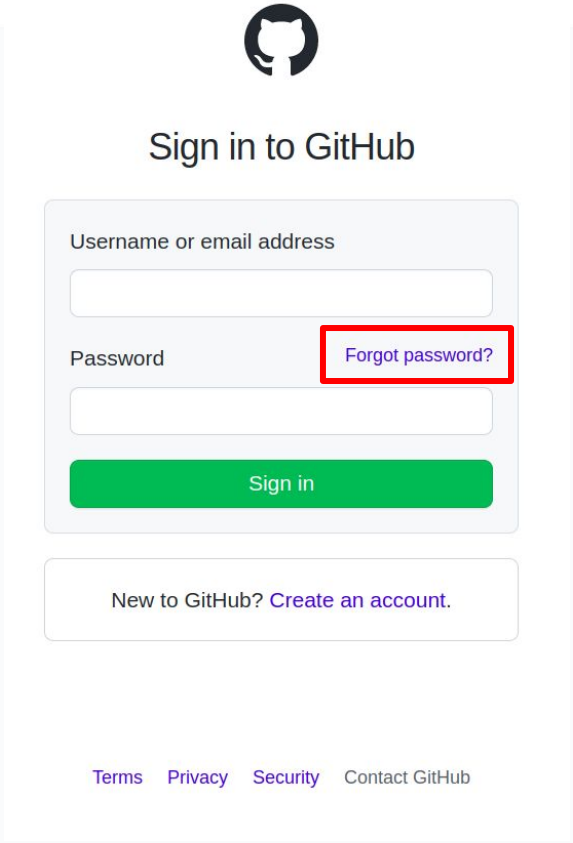
Page Object Model



Page Object Model



Page Object Model



Page Object Model

```python
@pytest.mark.ui
def test_check_incorrect_username():
    # Створення об'єкту для керування браузером
    driver = webdriver.Chrome(
        service=Service(r"/home/sbutenko/repos/LnD/Become QA Au
        )

    # відкриваємо сторінку https://github.com/login
    driver.get("https://github.com/login")

    # Знаходимо поле, в яке будемо вводити неправильне ім'я кор
    login_elem = driver.find_element(By.ID, "login_field")

    # Вводимо неправильне ім'я користувача або поштову адресу
    login_elem.send_keys("sergiibutenko@mistakeinemail.com")

    # Знаходимо поле, в яке будемо вводити неправильний пароль
    pass_elem = driver.find_element(By.ID, "password")

    # Вводимо неправильний пароль
    pass_elem.send_keys("wrong password")

    # Знаходимо кнопку sign in
    btn_elem = driver.find_element(By.NAME, "commit")

    # Емулюємо клік лівою кнопкою мишки
    btn_elem.click()

    # Перевіряємо, що назва сторінки така, яку ми очікуємо
    assert driver.title == "Sign in to GitHub · GitHub"

    # Закриваємо браузер
    driver.close()
```

```python
@pytest.mark.ui
def test_check_incorrect_username_page_object():
    # створення об'єкту сторінки
    sign_in_page = SignInPage()

    # відкриваємо сторінку https://github.com/login
    sign_in_page.go_to()

    # виконуємо спробу увійти в систему GitHub
    sign_in_page.try_login("page_object@gmail.com", "wrong password")

    # Перевіряємо, що назва сторінки така, яку ми очікуємо
    assert sign_in_page.check_title("Sign in to GitHub · GitHub")

    # Закриваємо браузер
    sign_in_page.close()
```

Page Object Model

```python
@pytest.mark.ui
def test_check_incorrect_username_page_object():
    # створення об'єкту сторінки
    sign_in_page = SignInPage()

    # відкриваємо сторінку https://github.com/login
    sign_in_page.go_to()

    # виконуємо спробу увійти в систему GitHub
    sign_in_page.try_login("page_object@gmail.com", "wrong password")

    # Перевіряємо, що назва сторінки така, яку ми очікуємо
    assert sign_in_page.check_title("Sign in to GitHub · GitHub")

    # Закриваємо браузер
    sign_in_page.close()

```

1. Спрощення розуміння коду
2. Спрощення підтримку коду
3. Перевикористання коду

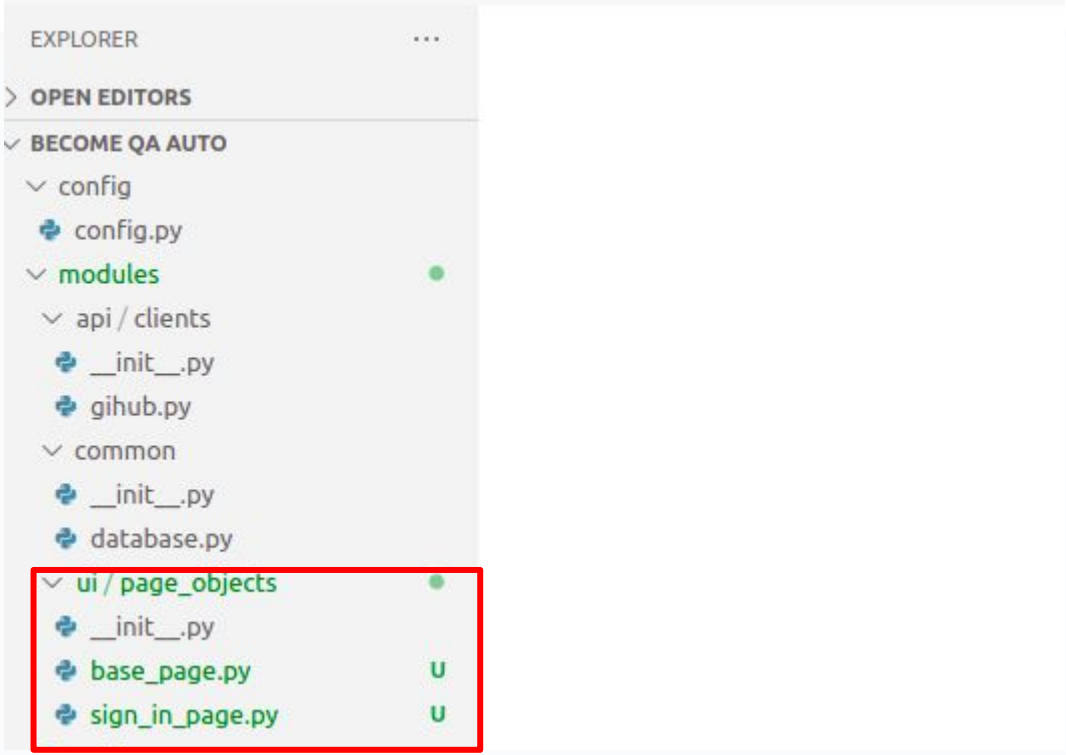
Практика

test_ui.py

tests > ui > test_ui.py > test_check_incorrect_username

```python
import pytest

from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By


@pytest.mark.ui
def test_check_incorrect_username():
    # Створення об'єкту для керування браузером
    driver = webdriver.Chrome(
        service=Service(r"/home/sbutenko/repos/LnD/Become QA Auto/" + "chromedriver")
        )

    # відкриваємо сторінку https://github.com/login
    driver.get("https://github.com/login")

    # Знаходимо поле, в яке будемо вводити неправильне ім'я користувача або поштову адресу
    login_elem = driver.find_element(By.ID, "login_field")

    # Вводимо неправильне ім'я користувача або поштову адрІесу
    login_elem.send_keys("sergiibutenko@mistakeinemail.com")

    # Знаходимо поле, в яке будемо вводити неправильний пароль
    pass_elem = driver.find_element(By.ID, "password")

    # Вводимо неправильний пароль
    pass_elem.send_keys("wrong password")

    # Знаходимо кнопку sign in
    btn_elem = driver.find_element(By.NAME, "commit")
      You, 2 days ago • 19.3
    # Емулюємо клік лівою кнопкою мишки
    btn_elem.click()

    # Перевіряємо, що назва сторінки така, яку ми очікуємо
    assert driver.title == "Sign in to GitHub · GitHub"

    # Закриваємо браузер
    driver.close()

```

◎ Практика


EXPLORER
OPEN EDITORS
BECOME QA AUTO
config
config.py
modules
api / clients
__init__.py
gihub.py
common
__init__.py
database.py
ui / page_objects
__init__.py
base_page.py
U
sign_in_page.py
U

◎ Практика

EXPLORER ···

OPEN EDITORS

BECOME QA AUTO

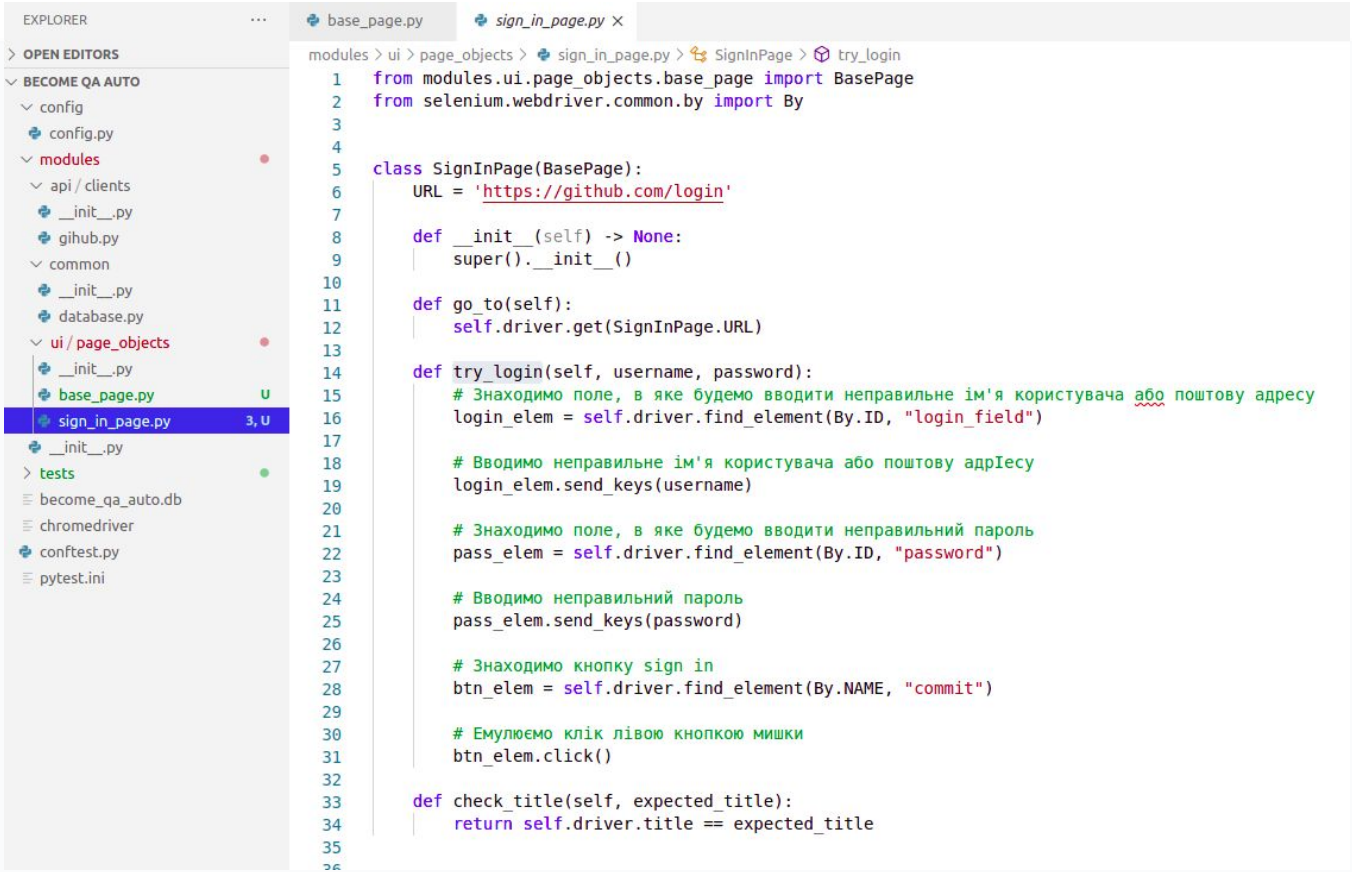
- config
  - config.py
- modules
  - api / clients
    - __init__.py
    - gihub.py
  - common
    - __init__.py
    - database.py
  - ui / page_objects
    - __init__.py
    - base_page.py U
    - sign_in_page.py U
- __init__.py

base_page.py

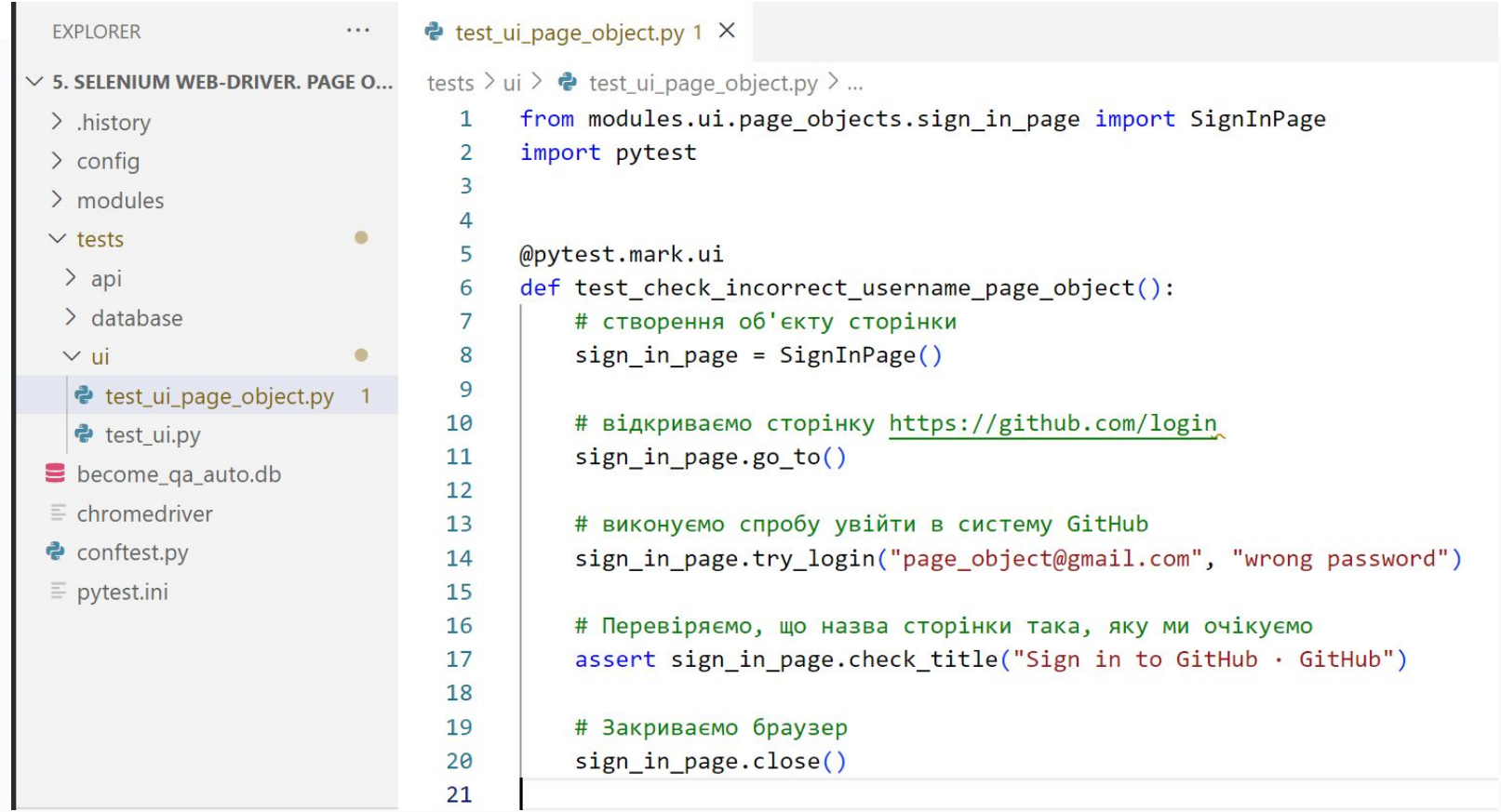
modules > ui > page_objects > base_page.py > ...

```python
from selenium import webdriver
from selenium.webdriver.chrome.service import Service


class BasePage:
    PATH = r"/home/sbutenko/repos/LnD/Become QA Auto/"
    DRIVER_NAME = "chromedriver"

    def __init__(self) -> None:
        self.driver = webdriver.Chrome(
            service=Service(BasePage.PATH + BasePage.DRIVER_NAME)
            )

    def close(self):
        self.driver.close()

```

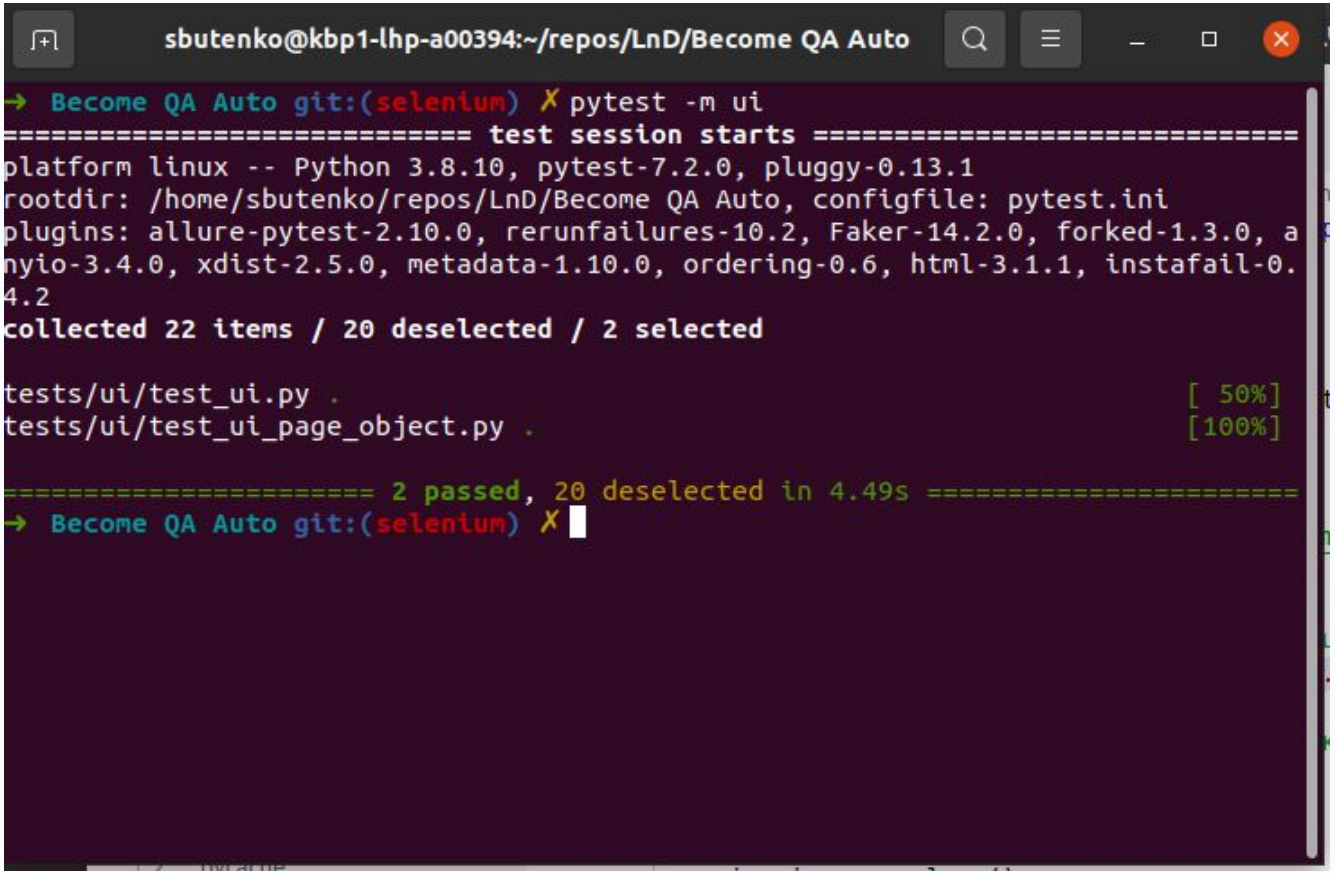
Практика


EXPLORER
OPEN EDITORS
BECOME QA AUTO
config
config.py
modules
api / clients
__init__.py
gihub.py
common
__init__.py
database.py
ui / page_objects
__init__.py
base_page.py U
sign_in_page.py 3, U
__init__.py
tests
become_qa_auto.db
chromedriver
conftest.py
pytest.ini
base_page.py
sign_in_page.py
modules > ui > page_objects > sign_in_page.py > SignInPage > try_login
1 from modules.ui.page_objects.base_page import BasePage
2 from selenium.webdriver.common.by import By
3
4
5 class SignInPage(BasePage):
6 URL = 'https://github.com/login'
7
8 def __init__(self) -> None:
9 super().__init__()
10
11 def go_to(self):
12 self.driver.get(SignInPage.URL)
13
14 def try_login(self, username, password):
15 # Знаходимо поле, в яке будемо вводити неправильне ім'я користувача або поштову адресу
16 login_elem = self.driver.find_element(By.ID, "login_field")
17
18 # Вводимо неправильне ім'я користувача або поштову адрIесу
19 login_elem.send_keys(username)
20
21 # Знаходимо поле, в яке будемо вводити неправильний пароль
22 pass_elem = self.driver.find_element(By.ID, "password")
23
24 # Вводимо неправильний пароль
25 pass_elem.send_keys(password)
26
27 # Знаходимо кнопку sign in
28 btn_elem = self.driver.find_element(By.NAME, "commit")
29
30 # Емулюємо клік лівою кнопкою мишки
31 btn_elem.click()
32
33 def check_title(self, expected_title):
34 return self.driver.title == expected_title
35

Практика


EXPLORER
5. SELENIUM WEB-DRIVER. PAGE O...
.history
config
modules
tests
api
database
ui
test_ui_page_object.py 1
test_ui.py
become_qa_auto.db
chromedriver
conftest.py
pytest.ini
test_ui_page_object.py 1
tests > ui > test_ui_page_object.py > ...
1 from modules.ui.page_objects.sign_in_page import SignInPage
2 import pytest
3
4
5 @pytest.mark.ui
6 def test_check_incorrect_username_page_object():
7 # створення об'єкту сторінки
8 sign_in_page = SignInPage()
9
10 # відкриваємо сторінку https://github.com/login
11 sign_in_page.go_to()
12
13 # виконуємо спробу увійти в систему GitHub
14 sign_in_page.try_login("page_object@gmail.com", "wrong password")
15
16 # Перевіряємо, що назва сторінки така, яку ми очікуємо
17 assert sign_in_page.check_title("Sign in to GitHub · GitHub")
18
19 # Закриваємо браузер
20 sign_in_page.close()
21

Практика


sbutenko@kbp1-lhp-a00394:~/repos/LnD/Become QA Auto
→ Become QA Auto git:(selenium) ✗ pytest -m ui
============================= test session starts ==============================
platform linux -- Python 3.8.10, pytest-7.2.0, pluggy-0.13.1
rootdir: /home/sbutenko/repos/LnD/Become QA Auto, configfile: pytest.ini
plugins: allure-pytest-2.10.0, rerunfailures-10.2, Faker-14.2.0, forked-1.3.0, a
nyio-3.4.0, xdist-2.5.0, metadata-1.10.0, ordering-0.6, html-3.1.1, instafail-0.
4.2
collected 22 items / 20 deselected / 2 selected
tests/ui/test_ui.py . [50%]
tests/ui/test_ui_page_object.py . [100%]
====================== 2 passed, 20 deselected in 4.49s ======================
→ Become QA Auto git:(selenium) ✗

Підсумки

Page Object Model - це архітектурний підхід, при якому в коді для кожної сторінки, яку ви використовуєте, в тестах створюється окремий клас, що описує в собі логіку та технічний опис сторінки.

Переваги використання архітектурного стилю розробки тестів page object model:

- спрощення розуміння коду;
- спрощення підтримку коду;
- перевикористання коду.