46. Page Object Model Pattern

Page object model Pattern

- Page object model or Page object repository or Page object factory is not a Framework but Pattern.
- POM talks about how we can manage/maintain the page objects/elements for multiple Test Cases.

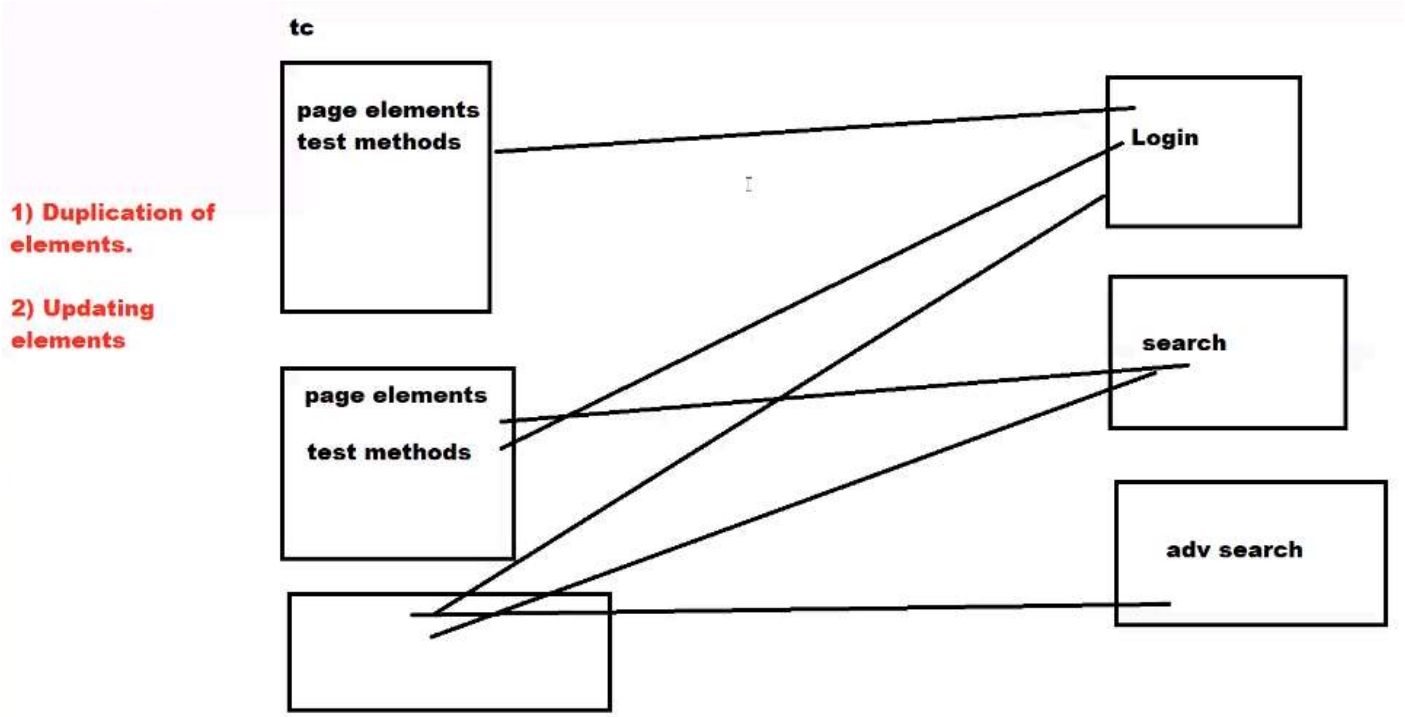
Explanation

Suppose we have multiple test cases in our project, for every test case we will include certain number of elements on the webpage along with the test methods in it.

- Instead of including everything in one file we will maintain test cases and page objects/elements in two different classes.This is called as a Page Object Model Pattern.

Without using Page Object Model Approach (Normal Approach)

Let say 3 different pages (Login, search, advanced search) in application.similarly we have 3 test cases (Login test case, search test case, advanced search test case) along with page elements.



Login.py

```
from selenium import webdriver
from selenium.webdriver.common.by import By
opt = webdriver.ChromeOptions()
opt.add_argument("--start-maximized")
opt.add_experimental_option("detach",True)
driver = webdriver.Chrome(options=opt)
driver.implicitly_wait(10)
driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login")
```

Login steps

```
driver.find_element(By.NAME, "username").send_keys("Admin")
driver.find_element(By.NAME, "password").send_keys("admin123")
driver.find_element(By.XPATH, "//button[normalize-space()='Login']").click()
act_title = driver.title
```

```
driver.quit()
assert act_title == "OrangeHRM"
```

Problems in Normal Approach

1. Same elements we are identifying in every test case (Duplication of Elements)
2. Suppose automated 10 Test Cases all 10 TC used Login Page element and suppose if tomorrow if the page element got modified or some attribute got changed then i need to go and modify those element in every Test Case (Updating Elements)

To overcome above problems we use Page object model pattern

With Page Object Model Approach

- In the Page object model pattern for every page in application we will create a separate class called Page object classes.
- Page object class contains the elements belonging to a specific page along with action methods.
- **so no need to write page elements only specify test methods.test methods talk to the class and get the elements and will perform the action.**

Note

- When we enter into a project, the first task is to create a page object class by analyzing test cases, number of pages.
- Page object should structure contains
  - **Locators**
  - **constructor**
  - **action methods**
- No of Locators = No of action methods

LoginPageObjects.py

```
from selenium.webdriver.common.by import By
class LoginPage:
```

Locators

```
  txtbox_username_name = 'username'
  txtbox_password_name = 'password'
  btn_login_xpath = '//button'
```

Constructor

```
  def __init__(self,driver):
     self.driver = driver  #instance variable
```

Action Methods

```
  def setUserName(self,username):
     self.driver.find_element(By.NAME,self.txtbox_username_name).send_keys(username)
  def setPassWord(self,password):
     self.driver.find_element(By.NAME,self.txtbox_password_name).send_keys(password)
  def clickLogin(self):
     self.driver.find_element(By.XPATH,self.btn_login_xpath).click()
```

test_Login.py

```
from selenium import webdriver
```

```
from LoginPageObjects import LoginPage
class TestLogin:
  def test_login(self):
      opt = webdriver.ChromeOptions()
      opt.add_argument("--start-maximized")
      opt.add_experimental_option("detach", True)
      driver = webdriver.Chrome(options=opt)
      driver.implicitly_wait(10)
      driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login")
      lp = LoginPage(driver)
      lp.setUserName("Admin")
      lp.setPassWord("admin123")
      lp.clickLogin()
      act_title = driver.title
      driver.quit()
      assert act_title == "OrangeHRM"
```

To Execute - pytest -s -v day1\test_Login.py

- Variables creating inside the methods refer to those variables by using the self keyword.

Thumb rules :

1. In the Page object class should have elements belonging to one single page every time i.e, one page object class should always represent only one single page.
2. Page Object class we should not hard code any data .The data should pass through test case
3. We should not include any validations and assertions in the page object class

Assignment

- Automate registration functionality using page object model approach

https://demo.opencart.com/index.php?route=account/register

https://tutorialsninja.com/demo/index.php?route=account/register