accenture technology

Learning and Knowledge Management

Module 7: Page Object Model

Module Objectives

At the end of this module, you will be able to:

- Explain creating object repository using Selenium Python API
- Create pages using object repository design pattern
- Integrate object repository with unittest

Topic List

- Page Object Model (POM) Design Pattern
- Advantages of Page Object Model
- Implementation of Page Object Model

Topic List

- **Page Object Model (POM) Design Pattern**
- Advantages of Page Object Model
- Implementation of Page Object Model

Page Object Model (POM) Design Pattern (1)

Challenges in creating tests without design pattern

The Selenium Python API is directly written in to python classes using:

- unittest
- Specifying locators
- WebDriver commands
- Test case steps into these classes

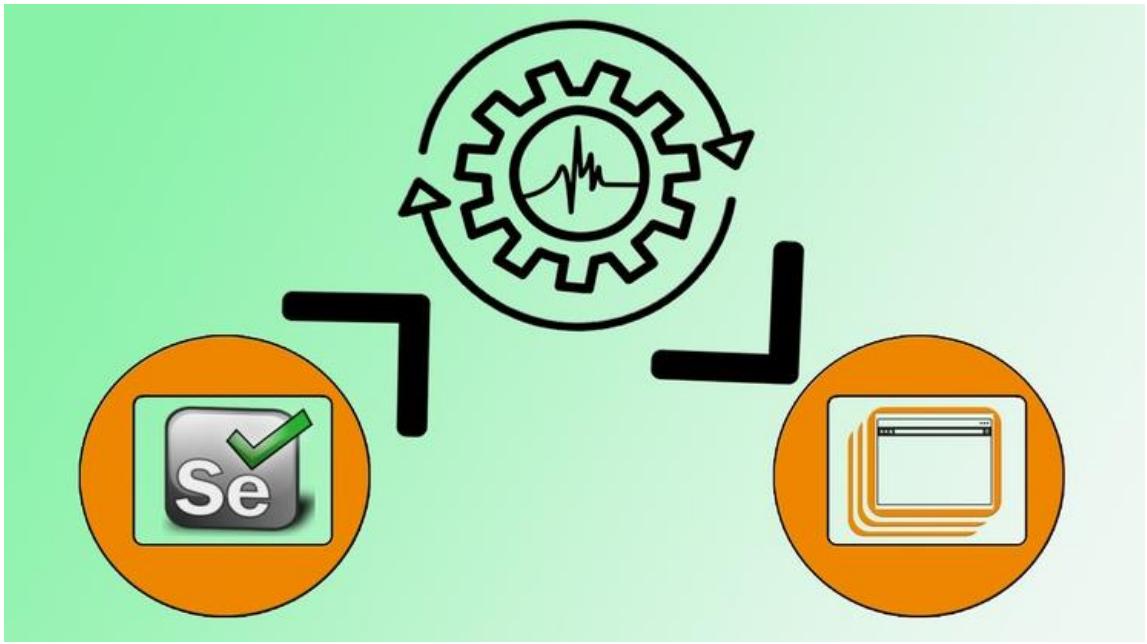
Challenges with the above approach:

- More tests to the test suites: difficult to maintain and makes tests brittle
- Page under test when modified by developers break the test

Page Object Model (POM) Design Pattern (2)

Selenium Python Framework

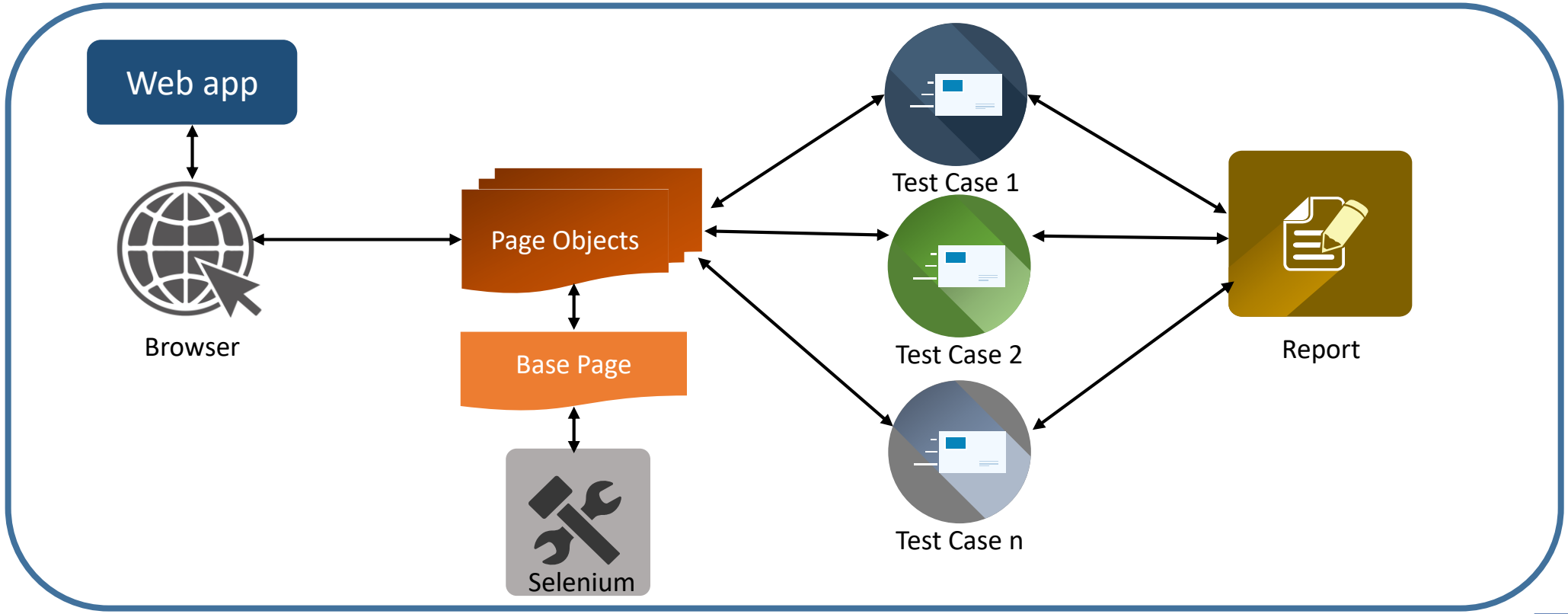
- Page Object Model is an object repository design pattern for designing the tests (Web UI functional testing)
- This object repository design pattern is widely used among Selenium users to structure the tests
- The pattern separates tests from Web UI objects and also provides a high level abstraction



Page Object Model (POM) Design Pattern (3)

How POM Works?

- Page Object pattern allows to create an object to represent each web page from the Application Under Test (AUT)
- Using this pattern, classes can be defined for each page, modelling all attributes and actions for that page
- The pattern creates a layer of separation between the test code and technical implementation of pages



Topic List

Page Object Model (POM) Design Pattern

Advantages of Page Object Model

Implementation of Page Object Model

Advantages of Page Object Model

Advantages

POM helps testing with the following:

- Creates a high level abstraction that helps minimize changes when the page under test is modified by the developers
- The change will be only in the page object and the calling tests will be unaffected
- Creates object repository independent of tests
- Creates reusable code that can be shared across multiple test cases
- Provisions more clear, flexible and maintainable tests

Topic List

Page Object Model (POM) Design Pattern

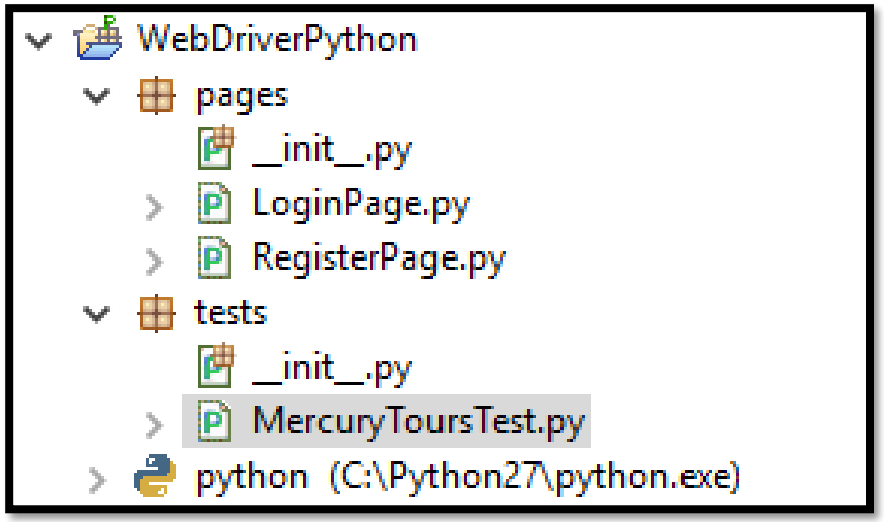
Advantages of Page Object Model

Implementation of Page Object Model

Implementation of Page Object Model

Step 1 :

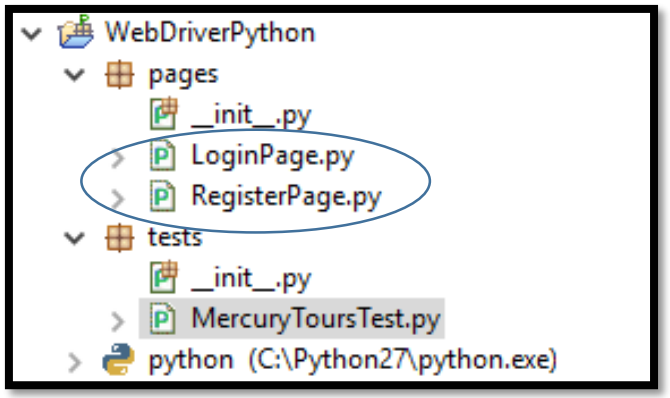
- Create two different packages by names **Pages** and **Tests** respectively
  - Tests: This package contains the unittest classes, which in turn uses page objects
  - Pages: This package contains the page classes specifying locators as attributes and call to WebDriver commands inside function definitions



Implementation of Page Object Model

Step 2:

- Create a python class in the **Pages** package
  - A separate page class is created for each page identified in the UI



Implementation of Page Object Model

Step 3:

- A page class contains
  - ✓ Constructor (WebDriver injection)
  - ✓ Attributes (Object Repository)
  - ✓ Function Definitions (Selenium Commands)
- Create the **LoginPage** class with the details

```
Created on Jan 2, 2018

@author: aswani.kumar.avilala
'''
from selenium.webdriver.common.by import By
class LoginPage(object):
    USER_NAME=(By.NAME,"userName")
    PASSWORD=(By.NAME,"password")
    SIGN_IN=(By.XPATH,"//input[@value='Login']")
    REGISTER=(By.LINK_TEXT,"REGISTER")

    def login(self):
        self.driver.find_element(*self.USER_NAME).send_keys("selenium")
        self.driver.find_element(*self.PASSWORD).send_keys("selenium")
        self.driver.find_element(*self.SIGN_IN).click()
        return self.driver.title
    def clickRegister(self):
        self.driver.find_element(*self.REGISTER).click()

    def __init__(self,driver):
        self.driver=driver
```

Implementation of Page Object Model

Step 4:

- Create the **RegisterPage** class with details

```
Created on Jan 2, 2018

@author: aswani.kumar.avilala
'''
from selenium.webdriver.common.by import By
class RegisterPage(object):
    USER_NAME=(By.ID,"email")
    PASSWORD=(By.NAME,"password")
    CONF_PASSWORD=(By.NAME,"confirmPassword")
    SUBMIT=(By.NAME,"register")
    SIGN_OFF=(By.LINK_TEXT,"SIGN-OFF")

    def register(self):
        self.driver.find_element(*self.USER_NAME).send_keys("selenium");
        self.driver.find_element(*self.PASSWORD).send_keys("selenium");
        self.driver.find_element(*self.CONF_PASSWORD).send_keys("selenium");
        self.driver.find_element(*self.SUBMIT).click();
    def signoff(self):
        self.driver.find_element(*self.SIGN_OFF).click()

    def __init__(self,driver):
        self.driver=driver
```

Implementation Of Page Object Model

Step 5:

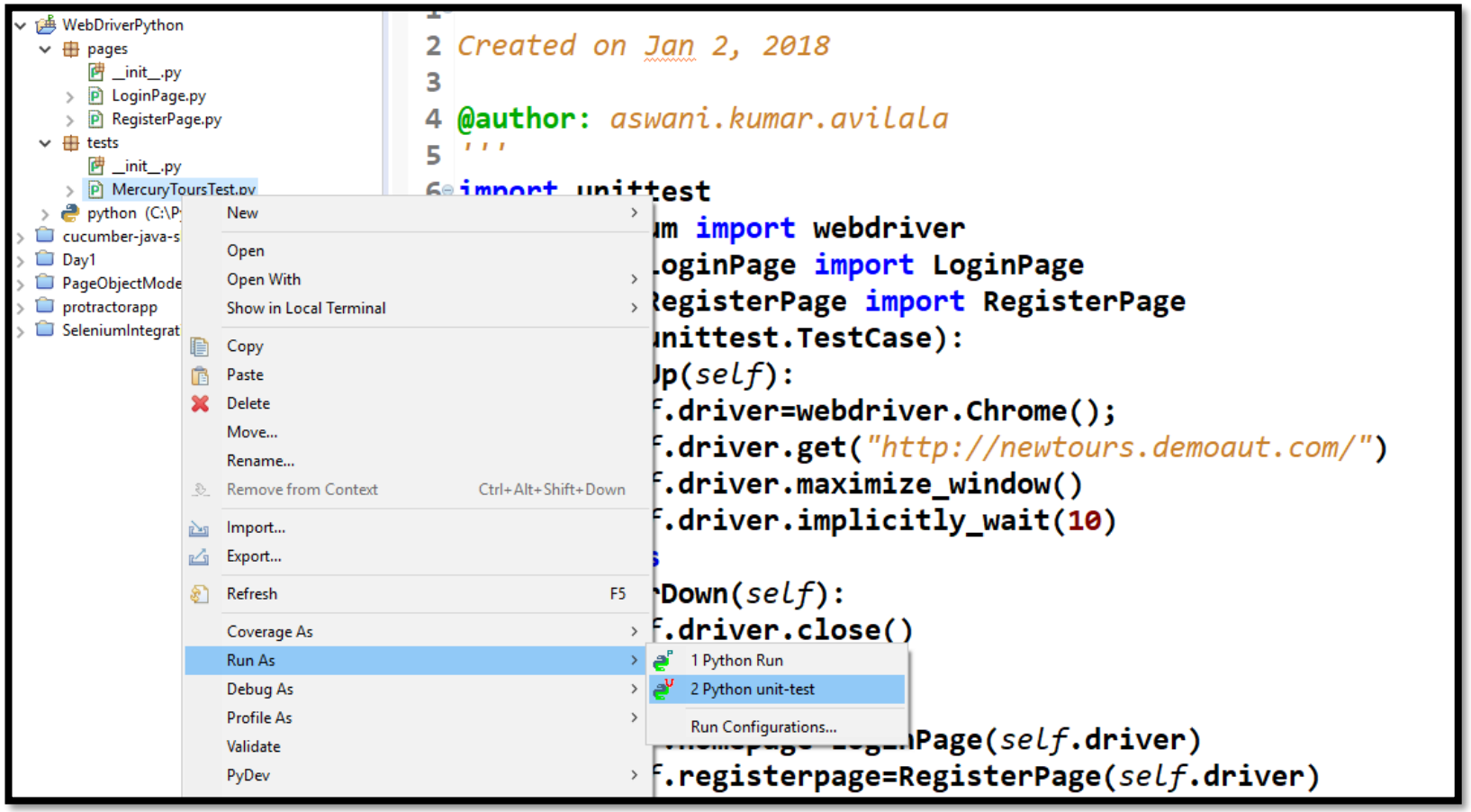
- Create unittest class **Test** under the Tests package

```
import unittest
from selenium import webdriver
from pages.LoginPage import LoginPage
from pages.RegisterPage import RegisterPage
class Test(unittest.TestCase):
    def setUp(self):
        self.driver=webdriver.Chrome();
        self.driver.get("http://newtours.demoaut.com/")
        self.driver.maximize_window()
        self.driver.implicitly_wait(10)
        pass
    def tearDown(self):
        self.driver.close()
        pass
    def testName(self):
        self.homepage=LoginPage(self.driver)
        self.registerpage=RegisterPage(self.driver)
        self.homepage.clickRegister()
        self.registerpage.register()
        self.registerpage.signoff()
        title=self.homepage.login()
        self.assertEquals(title,'Find a Flight: Mercury Tours:')
        pass
```

Implementation Of Page Object Model

Step 6:

- Run the test as python unittest



Exercise 7.1: Apply POM for Register Page Functionality

Implement Page Object Model as design pattern for TestMeAPP

- Test the Login and Register page functionality of Test Me App
  - Login Page
  - Register Page
  - Unit test on Test Me App

Refer Exercise 7.1 in Selenium with Python_wbk.doc for the detailed steps.

Module Summary

Now, you should be able to:

- Explain creating object repository using Selenium Python API
- Create pages using object repository design pattern
- Integrate object repository with unittest

Thank You