**Page Object model without PageFactory**

**package** pageObjects;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**public** **class** Home\_Page {

**private** **static** WebElement *element* = **null**;

**public** **static** WebElement lnk\_MyAccount(WebDriver driver){

*element* = driver.findElement(By.*xpath*("//ul[@class='pull-right noo-topbar-right']/li[2]/a"));

**return** *element*;

}

**public** **static** WebElement lnk\_LogOut(WebDriver driver){

*element* = driver.findElement(By.*xpath*("//ul[@class='pull-right noo-topbar-right']/li[3]/a"));

**return** *element*;

}

}

**package** pageObjects;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**public** **class** LogIn\_Page {

**private** **static** WebElement *element* = **null**;

**public** **static** WebElement txtbx\_UserName(WebDriver driver){

*element* = driver.findElement(By.*id*("username"));

**return** *element*;

}

**public** **static** WebElement txtbx\_Password(WebDriver driver){

*element* = driver.findElement(By.*id*("password"));

**return** *element*;

}

**public** **static** WebElement btn\_LogIn(WebDriver driver){

*element* = driver.findElement(By.*name*("login"));

**return** *element*;

}

}

**package** automationFramework;

**import** java.util.concurrent.TimeUnit;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.firefox.FirefoxDriver;

**import** org.testng.annotations.Test;

// Import package pageObject.\*

**import** pageObjects.Home\_Page;

**import** pageObjects.LogIn\_Page;

**public** **class** PageObjectModel {

**private** **static** WebDriver *driver* = **null**;

@Test

**public** **static** **void** withoutPF() {

String exePath1 = "G:\\Selenium RS 2019\\geckodriver-v0.26.0-win32\\geckodriver.exe";

System.*setProperty*("webdriver.gecko.driver", exePath1);

*driver* = **new** FirefoxDriver();

*driver*.manage().timeouts().implicitlyWait(10, TimeUnit.***SECONDS***);

*driver*.get("http://www.store.demoqa.com");

// To click on Dismiss hyperlink on the page

*driver*.findElement(By.*xpath*("//a[@class='woocommerce-store-notice\_\_dismiss-link']")).click();

// Use page Object library now

Home\_Page.*lnk\_MyAccount*(*driver*).click();

LogIn\_Page.*txtbx\_UserName*(*driver*).sendKeys("testuser\_1");

LogIn\_Page.*txtbx\_Password*(*driver*).sendKeys("Test@123");

LogIn\_Page.*btn\_LogIn*(*driver*).click();

System.***out***.println(" Login Successfully, now it is the time to Log Off buddy.");

Home\_Page.*lnk\_LogOut*(*driver*).click();

System.***out***.println(" Logged off Successfully");

*driver*.quit();

}

}

**Output:**

Login Successfully, now it is the time to Log Off buddy.

Logged off Successfully

PASSED: withoutPF

===============================================

Default test

Tests run: 1, Failures: 0, Skips: 0

===============================================

**Page Object model with PageFactory**

**package** pageFactory;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.support.FindBy;

**import** org.openqa.selenium.support.PageFactory;

**public** **class** Guru99Login {

/\*\*

\* All WebElements are identified by **@FindBy** annotation

\*/

WebDriver driver;

@FindBy(name="uid")

WebElement user99GuruName;

@FindBy(name="password")

WebElement password99Guru;

@FindBy(className="barone")

WebElement titleText;

@FindBy(name="btnLogin")

WebElement login;

**public** Guru99Login(WebDriver driver){

**this**.driver = driver;

//This initElements method will create all WebElements

PageFactory.*initElements*(driver, **this**);

}

//Set user name in textbox

**public** **void** setUserName(String strUserName){

user99GuruName.sendKeys(strUserName);

}

//Set password in password textbox

**public** **void** setPassword(String strPassword){

password99Guru.sendKeys(strPassword);

}

//Click on login button

**public** **void** clickLogin(){

login.click();

}

//Get the title of Login Page

**public** String getLoginTitle(){

**return** titleText.getText();

}

/\*\*

\* This POM method will be exposed in test case to login in the application

\* **@param** strUserName

\* **@param** strPasword

\* **@return**

\*/

**public** **void** loginToGuru99(String strUserName,String strPasword){

//Fill user name

**this**.setUserName(strUserName);

//Fill password

**this**.setPassword(strPasword);

//Click Login button

**this**.clickLogin();

}

}

**package** pageFactory;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.support.FindBy;

**import** org.openqa.selenium.support.PageFactory;

**public** **class** Guru99HomePage {

WebDriver driver;

@FindBy(xpath="//table//tr[@class='heading3']")

WebElement homePageUserName;

**public** Guru99HomePage(WebDriver driver){

**this**.driver = driver;

//This initElements method will create all WebElements

PageFactory.*initElements*(driver, **this**);

}

//Get the User name from Home Page

**public** String getHomePageDashboardUserName(){

**return** homePageUserName.getText();

}

}

**package** test;

**import** java.util.concurrent.TimeUnit;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.firefox.FirefoxDriver;

**import** org.testng.Assert;

**import** org.testng.annotations.BeforeTest;

**import** org.testng.annotations.Test;

**import** pageFactory.Guru99HomePage;

**import** pageFactory.Guru99Login;

**public** **class** Test99GuruLoginWithPageFactory {

String driverPath = "G:\\Selenium RS 2019\\geckodriver-v0.26.0-win32\\geckodriver.exe";

WebDriver driver;

Guru99Login objLogin;

Guru99HomePage objHomePage;

@BeforeTest

**public** **void** setup(){

System.*setProperty*("webdriver.gecko.driver", driverPath);

driver = **new** FirefoxDriver();

driver.manage().timeouts().implicitlyWait(10, TimeUnit.***SECONDS***);

driver.get("http://demo.guru99.com/V4/");

}

/\*\*

\* This test go to http://demo.guru99.com/V4/

\* Verify login page title as guru99 bank

\* Login to application

\* Verify the home page using Dashboard message

\*/

@Test(priority=1)

**public** **void** test\_Home\_Page\_Appear\_Correct(){

//Create Login Page object

objLogin = **new** Guru99Login(driver);

//Verify login page title

String loginPageTitle = objLogin.getLoginTitle();

Assert.*assertTrue*(loginPageTitle.toLowerCase().contains("guru99 bank"));

//login to application

objLogin.loginToGuru99("mgr123", "mgr!23");

// go the next page

objHomePage = **new** Guru99HomePage(driver);

//Verify home page

Assert.*assertTrue*(objHomePage.getHomePageDashboardUserName().toLowerCase().contains("manger id : mgr123"));

}

}

**Output:**

PASSED: test\_Home\_Page\_Appear\_Correct

===============================================

Default test

Tests run: 1, Failures: 0, Skips: 0

===============================================

**What Is Page Object Model (POM)?**

**Page Object Model** isa design pattern, used to create an object repository (object locators) for the web elements available in the application under test. The advantage of the model is that it reduces code duplication and improves test maintenance.

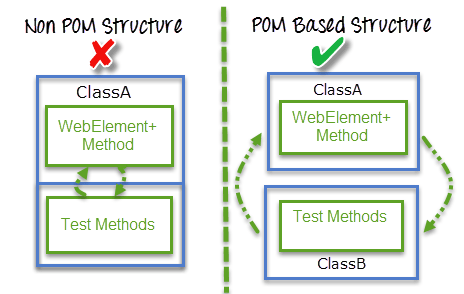
For each web page in the application, there should be a corresponding Page Class (Java class). This Page class will identify the WebElements of that web page and also contains Page methods which perform operations on those WebElements.

Name of these methods should be given as per the task they are performing, i.e., if a loader is waiting for the payment gateway to appear, POM method name can be waitForPaymentScreenDisplay().

The chief problem with script maintenance is that if 10 different scripts are using the same page element, with any change in that element, you need to change all 10 scripts. This is time consuming and error prone.

A better approach to script maintenance is to create a separate class file which would find web elements, fill them or verify them. This class can be reused in all the scripts using that element. In future, if there is a change in the web element, we need to make the change in just 1 class file and not 10 different scripts.

This approach is called **Page Object Model (POM)**. It helps make the code **more readable, maintainable**, and **reusable.**



## Advantages of POM

1. Page Object Pattern says operations and flows in the UI should be separated from verification. This concept makes our code cleaner and easy to understand.
2. The Second benefit is the **object repository is independent of test cases**, so we can use the same object repository for a different purpose with different tools. For example, we can integrate POM with TestNG/JUnit for functional Testing and at the same time with JBehave/Cucumber for acceptance testing.
3. Code becomes less and optimized because of the reusable page methods in the POM classes.
4. **Methods** get **more realistic names** which can be easily mapped with the operation happening in UI. i.e. if after clicking on the button we land on the home page, the method name will be like 'gotoHomePage()'.

## What is Page Factory?

Page Factory is an inbuilt Page Object Model concept for Selenium WebDriver but it is much optimized.

Here as well, we follow the concept of separation of Page Object Repository and Test Methods. Additionally, with the help of PageFactory class, we use annotations **@FindBy** to find WebElement. We use initElements method to initialize web elements

**package** com.wordpress.Pages;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.support.CacheLookup;

**import** org.openqa.selenium.support.FindBy;

**import** org.openqa.selenium.support.How;

**public** **class** LoginPageNew

{

WebDriver driver;

**public** LoginPageNew(WebDriver ldriver)

{

**this**.driver=ldriver;

}

@FindBy(id="user\_login")

@CacheLookup

WebElement username;

@FindBy(how=How.***ID***,using="user\_pass")

@CacheLookup

WebElement password;

@FindBy(how=How.***XPATH***,using=".//\*[@id='wp-submit']")

@CacheLookup

WebElement submit\_button;

@FindBy(how=How.***LINK\_TEXT***,using="Lost your password?")

@CacheLookup

WebElement forget\_password\_link;

**public** **void** login\_wordpress(String uid,String pass)

{

username.sendKeys(uid);

password.sendKeys(pass);

submit\_button.click();

}

}

**package** com.wordpress.Testcases;

**import** java.util.concurrent.TimeUnit;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.firefox.FirefoxDriver;

**import** org.openqa.selenium.support.PageFactory;

**import** org.testng.annotations.Test;

//import com.wordpress.Pages.LoginPage;

**import** com.wordpress.Pages.LoginPageNew;

//import Helper.BrowserFactory;

**public** **class** VerifyValidLogin

{

String driverPath = "G:\\Selenium RS 2019\\geckodriver-v0.26.0-win32\\geckodriver.exe";

WebDriver driver;

@Test

**public** **void** checkValidUser()

{

System.*setProperty*("webdriver.gecko.driver", driverPath);

driver = **new** FirefoxDriver();

driver.manage().timeouts().implicitlyWait(10, TimeUnit.***SECONDS***);

driver.get("http://demosite.center/wordpress/wp-login.php");

// This will launch browser and specific url

// Created Page Object using Page Factory

LoginPageNew login\_page=PageFactory.*initElements*(driver, LoginPageNew.**class**);

// Call the method

login\_page.login\_wordpress("admin", "demo123");

}

}

**Reference Links:**

<https://www.guru99.com/page-object-model-pom-page-factory-in-selenium-ultimate-guide.html>

<https://www.janbasktraining.com/blog/pom-with-page-factory-in-selenium-webdriver/>

Demo URL

<http://shop.demoqa.com/my-account/>

<https://www.toolsqa.com/selenium-webdriver/object-repository/>

<https://www.toolsqa.com/selenium-webdriver/page-object-model/>

<http://learn-automation.com/page-object-model-using-selenium-webdriver/>

<https://www.softwaretestinghelp.com/page-object-model-pom-with-pagefactory/>

<https://www.toptal.com/selenium/test-automation-in-selenium-using-page-object-model-and-page-factory>

<https://www.softwaretestingmaterial.com/page-object-model/>

<https://www.edureka.co/blog/page-object-model-in-selenium/>

<https://chercher.tech/java/page-object-model-selenium-webdriver>

<https://chercher.tech/java/featured-page-object-model-selenium-webdriver>

<https://www.seleniumeasy.com/selenium-tutorials/page-factory-pattern-in-selenium-webdriver>

<https://www.swtestacademy.com/page-factory-selenium-webdriver/>

<https://artoftesting.com/pageObjectModel>

<https://artoftesting.com/page-factory-in-selenium>