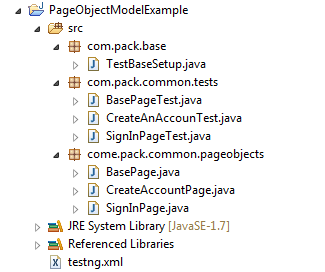
**Simple Page Object Model example**

In this example we will see a very simple Page Object Model example. To explain we have taken Google application and created BasePage, Sign In page and Create Account page.

To know what is page object model framework, Please refer [Page Object Model Introduction](http://seleniumeasy.com/selenium-tutorials/page-object-model-framework-introduction)  
Steps to Create a Simple Page Object Model. The structure of the sample project should look like in the screen shot below



**Step 1:** Create the Test Setup class  
This the main class for page object model, where we will create Webdriver object based on the browser type passed as a [parameter in textng.xml](http://seleniumeasy.com/testng-tutorials/parameterization-in-testng) file. We will also need to pass the base page application URL in testng.xml as parameter.

In this example we have taken only two browsers the default Firefox and chrome browser. To initiate [Chrome browser](http://seleniumeasy.com/selenium-tutorials/how-to-run-webdriver-in-chrome-browser), we need to set the System Property by providing the chromedriver exe path.

We need to create a method to return Webdriver 'driver' where this is used for test execution.

Below is the example TestBaseSetup.class file

package com.pack.base;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.annotations.AfterClass;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Parameters;

public **class** **TestBaseSetup** {

private WebDriver driver;

static String driverPath = "D:\\chromedriver\";

public WebDriver getDriver() {

return driver;

}

private void setDriver(String browserType, String appURL) {

switch (browserType) {

case "chrome":

driver = initChromeDriver(appURL);

break;

case "firefox":

driver = initFirefoxDriver(appURL);

break;

default:

System.out.println("browser : " + browserType

+ " **is** invalid, Launching Firefox as browser **of** choice..");

driver = initFirefoxDriver(appURL);

}

}

private static WebDriver initChromeDriver(String appURL) {

System.out.println("Launching google chrome with **new** profile..");

System.setProperty("webdriver.chrome.driver", driverPath

+ "chromedriver.exe");

WebDriver driver = new ChromeDriver();

driver.manage().window().maximize();

driver.navigate().to(appURL);

return driver;

}

private static WebDriver initFirefoxDriver(String appURL) {

System.out.println("Launching Firefox browser..");

WebDriver driver = new FirefoxDriver();

driver.manage().window().maximize();

driver.navigate().to(appURL);

return driver;

}

@Parameters({ "browserType", "appURL" })

@BeforeClass

public void initializeTestBaseSetup(String browserType, String appURL) {

try {

setDriver(browserType, appURL);

} catch (Exception e) {

System.out.println("Error....." + e.getStackTrace());

}

}

@AfterClass

public void tearDown() {

driver.quit();

}

}

**Step 2:** Now We will create Page Object classes. For each page we will create a separate class with constructor. We identify the locators and keep all together on just below the class. This will help us to re-use the locators for multiple methods and the main important is, we can easily find the locator and change if required.

We need to identify and list of all the possible functionalities on this page and we should write methods in such a way they are re-used. When ever there is a method to click on a button or link, we should return driver object of that page.

We will look into the below example. For method SignInBtn, we are returning after clicking on Sign In button as it is redirecting to sign in page.

**BasePage.class**

**package** come.pack.common.pageobjects;

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

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

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

**public** **class** **BasePage** {

**protected** WebDriver driver;

**private** By signInButton = By.linkText("Sign in");

**public** **BasePage**(WebDriver driver) {

**this**.driver = driver;

}

**public** SignInPage **clickSignInBtn**() {

System.out.println("clicking on sign in button");

WebElement signInBtnElement=driver.findElement(signInButton);

**if**(signInBtnElement.isDisplayed()||signInBtnElement.isEnabled())

signInBtnElement.click();

**else** System.out.println("Element not found");

**return** **new** SignInPage(driver);

}

**public** **void** **clickImagesLink**() {

*//It should have a logic to click on images link*

*//And it should navigate to google images page*

}

**public** String **getPageTitle**(){

String title = driver.getTitle();

**return** title;

}

**public** **boolean** **verifyBasePageTitle**() {

String expectedPageTitle="Google";

**return** getPageTitle().contains(expectedPageTitle);

}

}

In the below *SignInpage.class*, we have methods to click on create account and sign in button. We are just verifying an error in sign in page for now in this example.

**SignInPage.class**

**package** come.pack.common.pageobjects;

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

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

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

**public** **class** **SignInPage** {

**private** WebDriver driver;

**private** By headerPageText = By.cssSelector(".hidden-small");

**private** By createAccountLink = By.id("link-signup");

**private** By emailTextBox = By.id("Email");

**private** By passwordTextBox = By.id("Passwd");

**private** By loginBtn = By.id("signIn");

**private** By errorMsgTxt = By.id("errormsg\_0\_Passwd");

**public** **SignInPage**(WebDriver driver) {

**this**.driver=driver;

}

**public** String **getSignInPageTitle**() {

String pageTitle = driver.getTitle();

**return** pageTitle;

}

**public** **boolean** **verifySignInPageTitle**() {

String expectedTitle = "Sign in - Google Accounts";

**return** getSignInPageTitle().contains(expectedTitle);

}

**public** **boolean** **verifySignInPageText**() {

WebElement element = driver.findElement(headerPageText);

String pageText = element.getText();

String expectedPageText = "Sign in with your Google Account";

**return** pageText.contains(expectedPageText);

}

**public** CreateAccountPage **clickonCreateAnAccount**() {

WebElement element=driver.findElement(createAccountLink);

**if**(element.isDisplayed()||element.isEnabled())

element.click();

**return** **new** CreateAccountPage(driver);

}

**public** **boolean** **verifySignIn**() {

enterUserName("test");

enterPassword("pass");

clickOnSignIn();

**return** getErrorMessage().contains("incorrect");

}

**public** **void** **enterUserName**(String userName) {

WebElement emailTxtBox = driver.findElement(emailTextBox);

**if**(emailTxtBox.isDisplayed())

emailTxtBox.sendKeys(userName);

}

**public** **void** **enterPassword**(String password) {

WebElement passwordTxtBox = driver.findElement(passwordTextBox);

**if**(passwordTxtBox.isDisplayed())

passwordTxtBox.sendKeys(password);

}

**public** **void** **clickOnSignIn**() {

WebElement signInBtn = driver.findElement(loginBtn);

**if**(signInBtn.isDisplayed())

signInBtn.click();

}

**public** String **getErrorMessage**() {

String strErrorMsg = **null**;

WebElement errorMsg = driver.findElement(errorMsgTxt);

**if**(errorMsg.isDisplayed()&&errorMsg.isEnabled())

strErrorMsg = errorMsg.getText();

**return** strErrorMsg;

}

}

**CreateAccountPage.class**

**package** come.pack.common.pageobjects;

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

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

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

**public** **class** **CreateAccountPage** {

**private** WebDriver driver;

**private** By headerPageTxt = By.cssSelector(".signuponepage.main.content.clearfix>h1");

**public** **CreateAccountPage**(WebDriver driver) {

**this**.driver=driver;

}

**public** String **getPageTitle**() {

String title = driver.getTitle();

**return** title;

}

**public** **boolean** **verifyPageTitle**() {

String pageTitle = "Create your Google Account";

**return** getPageTitle().contains(pageTitle);

}

**public** **boolean** **verifyCreateAccountPageText**() {

WebElement element = driver.findElement(headerPageTxt);

String pageText ="Create your Google Account";

**return** element.getText().contains(pageText);

}

**public** **void** **createAccount**() {

*//need to write steps for creating an account*

}

}

**Step 3:** Now we will see how to write Tests for the above pages. For all the page Objects, we will now create Tests for each page. Which will help us in easy maintainability. if there is any change in the UI, we can simply change in one Page one place. We will see first example test for *"BasePageTest.class"*.

In the below example we verifying home page by taking the title. We have written a method in BasePage and we are just calling it here.

'package com.pack.common.tests;

import org.openqa.selenium.WebDriver;

import org.testng.Assert;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Test;

import com.pack.base.TestBaseSetup;

import come.pack.common.pageobjects.BasePage;

public class BasePageTest extends TestBaseSetup{

private WebDriver driver;

@BeforeClass

public void setUp() {

driver=getDriver();

}

@Test

public void verifyHomePage() {

System.out.println("Home page test...");

BasePage basePage = new BasePage(driver);

Assert.assertTrue(basePage.verifyBasePageTitle(), "Home page title doesn't match");

}

}

We will look into the next test 'SignInPageTest'. In the below test, we are verifying page title, page text, and Sign in functionality. For all these verifications we have defined methods in SignInPage class which we are calling from tests.

**package** com.pack.common.tests;

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

**import** org.testng.Assert;

**import** org.testng.annotations.BeforeClass;

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

**import** com.pack.base.TestBaseSetup;

**import** come.pack.common.pageobjects.BasePage;

**import** come.pack.common.pageobjects.SignInPage;

**public** **class** **SignInPageTest** **extends** **TestBaseSetup**{

**private** WebDriver driver;

**private** SignInPage signInPage;

**private** BasePage basePage;

@BeforeClass

**public** **void** **setUp**() {

driver=getDriver();

}

@Test

**public** **void** **verifySignInFunction**() {

System.out.println("Sign In functionality details...");

basePage = **new** BasePage(driver);

signInPage = basePage.clickSignInBtn();

Assert.assertTrue(signInPage.verifySignInPageTitle(), "Sign In page title doesn't match");

Assert.assertTrue(signInPage.verifySignInPageText(), "Page text not matching");

Assert.assertTrue(signInPage.verifySignIn(), "Unable to sign in");

}

}

Now the create test *'CreateAnAccountTest'*. Now we should be able to understand the verification that we are doing in the below test.

**package** com.pack.common.tests;

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

**import** org.testng.Assert;

**import** org.testng.annotations.BeforeClass;

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

**import** com.pack.base.TestBaseSetup;

**import** come.pack.common.pageobjects.BasePage;

**import** come.pack.common.pageobjects.CreateAccountPage;

**import** come.pack.common.pageobjects.SignInPage;

**public** **class** **CreateAnAccounTest** **extends** **TestBaseSetup** {

**private** WebDriver driver;

**private** SignInPage signInPage;

**private** BasePage basePage;

**private** CreateAccountPage createAccountPage;

@BeforeClass

**public** **void** **setUp**() {

driver=getDriver();

}

@Test

**public** **void** **verifyCreateAnAccounPage**() {

System.out.println("Create An Account page test...");

basePage = **new** BasePage(driver);

signInPage = basePage.clickSignInBtn();

createAccountPage = signInPage.clickonCreateAnAccount();

Assert.assertTrue(createAccountPage.verifyPageTitle(), "Page title not matching");

Assert.assertTrue(createAccountPage.verifyCreateAccountPageText(), "Page text not matching");

}

**public** **void** **verifySignInFunction**() {

}

}

Step 4: We are done with Base Setup, Page Objects creation, Tests creation. Now we will execute them using 'testng.xml' file. We will add the classes which we want to test.

Remember we need to pass the parameters *'browserType'* and *'appURL*' for the base Setup class.

<!DOCTYPE suite SYSTEM "[http://testng.org/testng-1.0.dtd">](http://testng.org/testng-1.0.dtd)

[<](http://testng.org/testng-1.0.dtd)**[suite](http://testng.org/testng-1.0.dtd)** name="Page Object test example">

<**parameter** name="appURL" value="[https://www.google.co.in/"/>](https://www.google.co.in/)

[<](https://www.google.co.in/)**[parameter](https://www.google.co.in/)** name="browserType" value="firefox"/>

  <**test** name="sample test">

    <**classes**>

      <**class** name="com.pack.common.tests.HomePageTest"/>

      <**class** name="com.pack.common.tests.SignInPageTest"/>

      <**class** name="com.pack.common.tests.CreateAnAccounTest"/>

    </**classes**>

  </**test**>

</**suite**>

The above Page Object Model framework is the simple one without using any Build tools, Loggers, Listeners and Utilities. In the next coming post we will add all these things and design a Robust Page Object Model Framework which are used in the Industry.

The below are the list of things which are commonly used in the Real time Page Object Model Framework.  
1. Selenium Webdriver with Java  
2. Maven Build Tool  
3. TestNG (We will user TestNG Listeners also)  
4. Log4j  
5. Eclipse IDE