### **Task #1**

### **What is POM**

Pom is Java design pattern for designing of classes for that purpose we need to maintain

all the web elements which we have to handle into a separate class.Also each web page will be handled in seprate class.

POM follows encapsulation concept in any project.As we know encapsulation is one of the OOPS principal in that

we declare data member as Private and method as publc (get an set method) so same thing we are doing in POM classes

Data members should be declared globaly with the access modifier as private and method as public.

Now intialize the data members within a cosntructor with access level as public.

Number of data member that need to be cretaed under POM class will depend upon number of component that need to be handled in our webpage,

POM class will not contain main method so to run a POM class we need to declare another class,

TesT class with main Method.Test class will contain all the navigantion steps to test an application .

This approach is called Page Object Model in Selenium. It helps make the code more readable, maintainable, and reusable.

### **Why Do We Use The Page Object Model?**

POM is a combination of data-driven, modular and hybrid frameworks. It’s an approach to systematically organizing the scripts in such a way that it makes it easy for the QA to maintain the code free of hassles and also helps to prevent redundant or duplicate code.

For instance, if there is a change in the locator value on a specific page, then it is very easy to identify and make that quick change only in the script of the respective page without impacting the code elsewhere.

**Implementation Of Simple POM In Selenium**

#### #1) Scenario To Automate

Now we automate the given scenario using the Page Object Model.

****The scenario is explained below:****

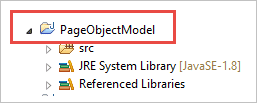
****Step 1:**** Launch the site “ https: //demo.vtiger.com ”.  
****Step 2:**** Enter the valid credential.  
****Step 3:**** Login to the site.  
****Step 4:**** Verify the Home page.  
****Step 5:**** Logout the site.  
****Step 6:**** Close the Browser.

#### #2) Selenium Scripts For The Above Scenario In POM

****Now we create the POM Structure in Eclipse, as explained below:****

****Step 1:**** Create a Project in Eclipse – POM based Structure:

a) Create Project “ Page Object Model ”.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/09/Page-Object-Model.png)

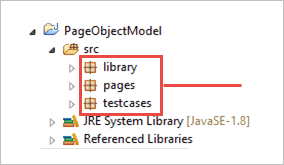
b) Create 3 Package under the project.

* library
* pages
* test cases

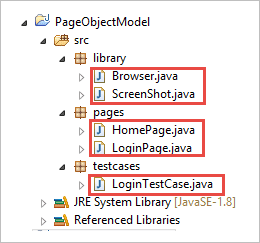
****Library:**** Under this, we put those codes that need to be called again and again in our test cases like Browser launch, Screenshots, etc. The user can add more classes under it based on the project need.

****Pages:**** Under this, classes are created for each page in the web application and can add more page classes based on the number of pages in the application.

****Test cases:**** Under this, we write the login test case and can add more test cases as required to test the whole application.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/09/page-obj-model-src.png)

c) Classes under the Packages are shown in the below image.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/09/Classes-under-the-Packages.png)

****Step**** ****2:**** Create the following classes under the library package.

****Browser.java:****In this class, 3 browsers ( Firefox, Chrome and Internet Explorer ) are defined and it is called in the login test case. Based on the requirement, the user can test the application in different browsers as well.

****package**** library;

****import**** org.openqa.selenium.WebDriver;****import**** org.openqa.selenium.chrome.ChromeDriver;****import**** org.openqa.selenium.firefox.FirefoxDriver;****import**** org.openqa.selenium.ie.InternetExplorerDriver;

****public**** ****class**** Browser {

****static**** WebDriver driver;

****public**** ****static**** WebDriver StartBrowser(String browsername , String url)

{

// If the browser is Firefox ****if****(browsername.equalsIgnoreCase("Firefox"))

// Set the path for geckodriver.exe

System.setProperty("webdriver.firefox.marionette"," E://Selenium//Selenium\_Jars//geckodriver.exe ");

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

}

// If the browser is Chrome ****else**** ****if****(browsername.equalsIgnoreCase("Chrome"))

{

// Set the path for chromedriver.exe

System.setProperty("webdriver.chrome.driver","E://Selenium//Selenium\_Jars//chromedriver.exe");

driver = ****new**** ChromeDriver();

}

// If the browser is IE ****else**** ****if****(browsername.equalsIgnoreCase("IE"))

{

// Set the path for IEdriver.exe

System.setProperty("webdriver.ie.driver","E://Selenium//Selenium\_Jars//IEDriverServer.exe");

driver = ****new**** InternetExplorerDriver();

}

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

driver.get(url); ****return**** driver;

}

}

****Step 3 :**** Create page classes under Page package.

****HomePage.java:****This is the Home page class, in which all the elements of the home page and methods are defined.

****package**** pages;

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

****public**** ****class**** HomePage {

WebDriver driver;

By logout = By.id("p\_lt\_ctl03\_wSOB\_btnSignOutLink");

By home = By.id("p\_lt\_ctl02\_wCU2\_lblLabel");

//Constructor to initialize object****public**** HomePage(WebDriver dr)

{ ****this****.driver=dr;

} ****public**** String pageverify()

{ ****return**** driver.findElement(home).getText();

} ****publicvoid**** logout()

{

driver.findElement(logout).click();

}

}

****LoginPage.java:**** This is the Login page class, in which all the elements of the login page and methods are defined.

****package**** pages;****import**** org.openqa.selenium.By;****import**** org.openqa.selenium.WebDriver;****public**** ****class**** LoginPage {

WebDriver driver;

By UserID = By.xpath("//\*[contains(@id,'Login1\_UserName')]");

By password = By.xpath("//\*[contains(@id,'Login1\_Password')]");

By Submit = By.xpath("//\*[contains(@id,'Login1\_LoginButton')]");

//Constructor to initialize object****public**** LoginPage(WebDriver driver)

{ ****this****.driver = driver;

}****public**** ****void**** loginToSite(String Username, String Password)

{ ****this****.enterUsername(Username); ****this****.enterPasssword(Password); ****this****.clickSubmit();

}****publicvoid**** enterUsername(String Username)

{

driver.findElement(UserID).sendKeys(Username);

}****publicvoid**** enterPasssword(String Password)

{

driver.findElement(password).sendKeys(Password);

}****publicvoid**** clickSubmit()

{

driver.findElement(Submit).click();

}

}

****Step 4:**** Create Test Cases for the login scenario.

****LoginTestCase.java:****This is the LoginTestCase class, where the test case is executed. The user can also create more test cases as per the project need.

****package**** testcases;****import**** java.util.concurrent.TimeUnit;****import**** library.Browser;****import**** org.openqa.selenium.WebDriver;****import**** org.testng.Assert;****import**** org.testng.ITestResult;****import**** org.testng.annotations.AfterMethod;****import**** org.testng.annotations.AfterTest;****import**** org.testng.annotations.BeforeTest;****import**** org.testng.annotations.Test;****import**** pages.HomePage;****import**** pages.LoginPage;****public**** ****class**** LoginTestCase {

WebDriver driver;

LoginPage lp;

HomePage hp; ****int**** i = 0;

// Launch of the given browser.

@BeforeTest ****public**** ****void**** browserlaunch()

{

driver = Browser.StartBrowser("Chrome", "http://demostore.kenticolab.com/Special-Pages/Logon.aspx");

driver.manage().timeouts().implicitlyWait(30,TimeUnit.****SECONDS****);

lp = ****new**** LoginPage(driver);

hp = ****new**** HomePage(driver);

}

// Login to the Site.

@Test(priority = 1) ****public**** ****void**** Login()

{

lp.loginToSite("gaurav.3n@gmail.com","Test@123");

}

// Verifing the Home Page.

@Test(priority = 2) ****public**** ****void**** HomePageVerify()

{

String HomeText = hp.pageverify();

Assert.assertEquals(HomeText, "Logged on as");

}

// Logout the site.

@Test(priority = 3) ****public**** ****void**** Logout()

{

hp.logout();

}@AfterTest ****public**** ****void**** closeBrowser()

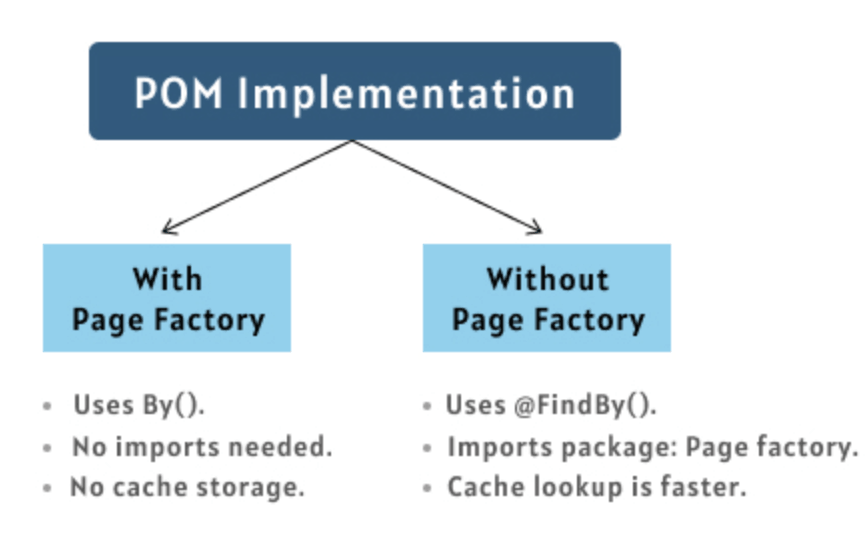
{ driver.close(); }}

****Step 5:**** Execute “ LoginTestCase.java “.

****Step 6:**** Output of the Page Object Model:

* Launch the Chrome browser.
* The demo website is opened in the browser.
* Login to the demo site.
* Verify the home page.
* Logout the site.
* Close the browser.

**POM Implementation :**

  
**What is the difference between with Page Factory and without page Factory?**

**Page Object** is a class that represents a web page and hold the functionality and members. **Page Factory** is a way to initialize the web elements you want to interact with within the page object when you create an instance of it.