**Page Object Model**

**Page Object Model (POM) Design With Selenium WebDriver- Part -1:**

* Complete Page Object Model (POM) using Page Factory in Selenium
* Learn Page Object Model Step by Step
* POM design - Architecture & Pattern
* What is Page Chaining Model
* Use of PageFactory API in Selenium

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

**Technologies Used:**

• Selenium WebDriver - v3.0.4 (Open Source)

• JDK 1.8 (Java Development Kit)

• TestNG (Test Unit Framework) • Log4j (logging API)

• Maven (Build Automation Tool)

• Apache POI API (Read-Write utilities for Excel - Test Data Handling) Eclipse/IntelliJ (Java Editor)

• Browser - Google Chrome/FF

**Automation Framework Architecture:**

• POM (Page Object Model) Design Page Factory API of WebDriver

• Maven (Build Automation Tool) • Test Libraries for different UI Pages

• Test Utilities for different generic functions

• Report - Dashboard (Pass/Fail Test) by using Extent Report

• API Jenkins - Continuous Integration Tool

• GITHub Repo (Code Versioning Tool)

* **Page Object Model** is not a Framework, it is a Design Pattern, an Automation Approach, it is also called **Page Chaining Model**

**Page Layer/Component-1:**

* Lets take an E-Commerce site, there will be an different pages like Home Page, Login Page, Registration Page, Search Page, Add to Cart Page, Payment Page, so in POM for each & every Page we need to create an separate Java Class like HomePage.java…etc,(If we have 100 Pages in our Project, we have to create 100 Java Classes)
* For these Pages we have to Design the Objects like WebElements(page libraries or Object Repositories, whatever the Login Page elements are there(uname,pwd, SigIn…these are LoginPage Elements)we have to store that in LoginPage.java

**Object Repositories – Collection of all WebElements/WebObjects**

* After storing all this, next second thing is we have to define Actions/methods Class for Features of that particular page, Features(Login Functionalities like click on login, click on forgot pwd, click on Registration, check box, header, logo, footer…whatever we see on Login Page we call it as WebElements/WebObjects), & Actions(clickonLogin(), clickonRegister(), loginButton()…these are methods/funtions for that pages)

**Test Layer-2:**

* Here we have to create Test Layer, for each & every Class we create a Test Layer(writing Test Cases) like LoginPageTest.Java, HomePageTest.java…etc, this Test Layer will be written with the help of **TestNG**, here we will be using all the **TestNG Annotations**, in which we will be writing all the Test Cases

**BaseClass Layer-3:**

* Before these 2 Layers we need to create an **Base Class** like **TestBase.java**(most imp class), this is **Parent** Class of all Classes, All **Page Layer Classes** & **Test Layer Classes** are **Child Classes** which are extending **Parent Class(Base Class)**
* In Base Class whatever the prerequisites are there like initializing driver(WebDriver driver= new ChromeDriver();) will initialize in TestBase.java & properties file, amx window, timeouts(implicit & explicit), delete cookies, getUrl(), driver.quit() all these will be defined here(it’s a common location) here use the concept of Inheritance(parent & child relationship) & then access all these methods directly from the Base Class), its an one time activity, we will initialize all this only once, bcoz to reuse & to save time & for code maintenance & it is a systematic Approach, bcoz we cant write every time for all Classes

**EnvironmentVariables/Configuration Layer-4:**

* Apart from this I will create one more layer **config.properties**, I will define all my environment variables over here(url, uname, pwd, browser…etc)

**TestData Layer-5:**

* We will store all our data in Excel files(read data by using Apache POI API), I can create different Excel Sheets

**Utility Layer-6:**

* I will create one TestUtil.java, in which all utilities will be stored like screenShots(), sendMail(), commonUtil(), These are all Generic Functions which is common for all Classes

**Reporting Layer-7:**

* We have to use some reports for project, it can be HTML Reports, TestNG Reports(extent Reports), XML Reports(Test Reports), this is the most imp thing, we have to share after the execution like how many Test Cases passed/Failed

**Packages:**

1. All Page Layer Classes will be created under com.qa.pages Package
2. All Test Layer Classes will be created under com.qa.tests Package
3. Base Class will be under com.qa.base Package
4. TestData class will be under com.qa.testdata Package
5. All Utility Classes will be under com.qa.util Package
6. Reporting Classes will be under Test-Output/reports Package
7. Env Variables Class will be under com.qa.config Package

In POM, if click on Login Page it will redirect to Home page(it is the landing Page of Login Page), in HP if click on search link it will redirect to Search Page…(all the pages are interconnected to each other by clicking some links/buttons ,I can go from lp to hp, hpt to sp…etc), so it is some times called **Page Chaining Model**(there is a chain, bcoz all the pages are interconnected)

**Technologies Used:**

1. **Java : It is used to write thecode**
2. **Selenium Webdriver** :
3. **TestNG**  : For writing Test cases
4. **Maven**  : For creating the Build
5. **Apache POI API** : to read the data from Excel
6. **Extent Report/TestNG** : For reporting
7. **Log4J API** : To Generate the Logs(instead of s.o.p, we will use this)
8. **Jenkins** : Continuos Integration to Trigger the Build
9. **GIT-Repo** : To maintain the code, where we will push the code(for check-in & check-out code)
10. **Selenium Grid** : For Parallel testing
11. **Browsers** : Chrome/Firefox/IE
12. **Platforms** : MC/WIN/LINUX
13. **Virtual Machines** : Sauce labs/Browser Stack/Locally in our systems

