**What is Selenium Framework? Types of Framework?**

* The Selenium Framework is a code structure that makes code maintenance easy and efficient.
* Without frameworks, users may place the “code” and “data” at the same location which is neither reusable nor readable.
* Frameworks produce beneficial outcomes like increased code reusability, higher portability, reduced cost of script maintenance, better code readability, etc.
* There are mainly three types of frameworks created by Selenium WebDriver to automate manual test cases

**Framework Types**

1. Data Driven (TDD)

2. Keyword Driven (BDD, Cucumber)🡪 when🡪Then

3. Hybrid

**1. Data Driven**

* Data Driven Framework in Selenium is a method of separating Test data from the test case.
* Once the Test data are separated from the test case, it can be easily modified for a specific functionality without changing the code.
* It is used to fetch Test Data from external files like Excel, .csv, .xml or some database tables, properties file

**POM with page factory**

**POM (page object module):**

It is a java design pattern used for design of classes in Test script.

Page Object model is an object design pattern in Selenium, where

* Web pages are represented as classes
* The various elements on the page are defined as variables in the class.

In this case we will use **Page Factory** to initialize web elements that are defined in web page classes or Page Objects.

POM Strictly follows encapsulation concept where

1. Data member should be declared globally with access level private using @findBy Annotation

2. Initialize within a constructor with access level public using pagefactory

3. Utilize within a method with access level public

Note:

* Number of D.M. that need to be created under a POM class will depends on number of elements that need to be handle in a webpage.
* POM class will not contain a main method, to run a POM class we require another class with main() ie. Test class
* Test class will contain all the navigation steps to test an application

**Pagefactory**:

* It is a class which contains static method like initElements.
* To initialize D.M./variable in PageFactory we need to use initElements method within the constructor.

Syntax: PageFactory.initElements(driver, this);

* initElements will initialize D.M by identifying each component present in a webpage by using @findBy annotation, which takes locator type as an input.

Syntax: @FindBy(locator Type ="locator value/exression") private WebElement D.M. ;

**@FindBy(xpath=”pin”) private WebElement Pin;**

**Working of PageFactory:**

* While executing Test Script initElement method will convert all the data members @findBy annotation to findElement(), this process is known as basic/early initialization.-->after creating object of test class
* @FindBy(xpath="//span[text()='KV']") private WebElement PN; ---->
* private WebElement PN = driver.findElement(By.xpath("//span[text()='KV']"));
* To perform action on component we need to call a methods.
* Before performing each action, initElement method will identifies component present or not, then it will do complete initialization this process is known as late/lazy initialization.

**Disadvantage of POM (without PageFactory):**

1. POM will initialize all the component before performing actions, but sometimes application may contain few components which will be hidden & displayed once we perform action on components, that hidden component will not be displayed while pom initializing, so it throws "No such element" exception.
2. To overcome drawback of pom, we need to use "PageFactory" which is an extension of pom.

**Difference between POM & PageFactory**

|  |  |
| --- | --- |
| **POM** | **Page factory** |
| It will initialize all the D.M. present in class completely before performing action on components | It will initialize the D.M. present in a class before  performing each action. |
| It will use if webpage is not containing hidden elements | It will be used if webpage is containing hidden elements. |

**Advantages of POM:**

1) Makes the code reusable

2) It is useful in reducing code duplication

3) It makes ease in maintaining the code (flow in the UI is separated from verification)

4) Makes code readable (Methods get more realistic names)

5) The Code becomes less and optimized

**POM class:**

* POM class depends on webpage present in an application.
* For each webpage pom class will be created, no of POM class depends on no of webpages present in an application.
* In each POM class D.M./variable are created in encapsulation concept by using pagefactory.
* No of D.M. created in POM class will depend on no of elements present in a webpage.
* Each declared D.M. should initialized & utilized in POM class.

**Test class:**

* Test class depends on no of Test cases written by manual Test engineer.
* Test class will contains navigation steps & inputs that need to be given to the components/elements.
* In test class data/inputs that can be given directly or through external source like Excel sheet.

Eg1

Loginpage🡪

**package** KitePOM;

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

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

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

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

**public** **class** KiteLoginPage {

@FindBy(xpath = "//input[@id='userid']") **private** WebElement UserID;

@FindBy(xpath = "//input[@id='password']") **private** WebElement PWD;

@FindBy(xpath = "//button[@class='button-orange wide']") **private** WebElement LoginButton;

**public** KiteLoginPage (WebDriver driver)

{

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

}

**public** **void** EnterUserId()

{

UserID.sendKeys("DV1510");

}

**public** **void** EnterPassword()

{

PWD.sendKeys("Year@123");

}

**public** **void** ClickOnLoginButton()

{

LoginButton.click();

}

}

PinPage🡪

**package** KitePOM;

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

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

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

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

**public** **class** KitePinPage {

@FindBy(xpath = "//input[@id='pin']") **private** WebElement Pin;

@FindBy(xpath = "//button[@class='button-orange wide']") **private** WebElement ContinueButton;

**public** KitePinPage(WebDriver driver)

{

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

}

**public** **void** EnterPin()

{

Pin.sendKeys("959594");

}

**public** **void** ClickOnContinue()

{

ContinueButton.click();

}

}

KiteHomePage

**package** KitePOM;

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

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

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

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

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

**public** **class** KiteHomePage {

@FindBy(xpath = "//span[@class='user-id']") **private** WebElement Userid;

**public** KiteHomePage(WebDriver driver) {

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

}

**public** **void** UserIdValidation()

{

String ActualUserid = Userid.getText();

String ExpecxtedUserid="DV1510";

**if**(ActualUserid.equals(ExpecxtedUserid))

{

System.***out***.println("TC passed user id is matching");

}

**else** {

System.***out***.println("TC failed user id is not-matching");

}

}

}

Test Class

**package** TestKite;

//import java.io.FileInputStream;

**import** java.io.IOException;

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

**import** org.apache.poi.EncryptedDocumentException;

**import** org.apache.poi.ss.usermodel.Sheet;

**import** org.apache.poi.ss.usermodel.WorkbookFactory;

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

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** org.openqa.selenium.chrome.ChromeOptions;

**import** KitePOM.KiteHomePage;

**import** KitePOM.KiteLoginPage;

**import** KitePOM.KitePinPage;

//import KitePOM.KiteLoginPage1;

**public** **class** KiteTest {

**public** **static** **void** main(String[] args) **throws** EncryptedDocumentException, IOException {

ChromeOptions options=**new** ChromeOptions();

//options.addArguments("--disable-notifications");

options.addArguments("headless");

System.*setProperty*("webdriver.chrome.driver", "F:\\Velocity\\May-2021 Class\\test2\\chromedriver\_win32 (1)\\chromedriver.exe");

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

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

driver.get("https://kite.zerodha.com/");

KiteLoginPage login= **new** KiteLoginPage(driver);

login.EnterUserId();

login.EnterPassword();

login.ClickOnLoginButton();

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

KitePinPage Login2= **new** KitePinPage(driver);

Login2.EnterPin();

Login2.ClickOnContinue();

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

KiteHomePage Home=**new** KiteHomePage(driver);

Home.UserIdValidation();

}

}

Reading data from Excel

package KitePOM;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.support.FindBy;

import org.openqa.selenium.support.PageFactory;

public class KiteLoginPage {

@FindBy(xpath = "//input[@id='userid']") private WebElement UserID;

@FindBy(xpath = "//input[@id='password']") private WebElement PWD;

@FindBy(xpath = "//button[@class='button-orange wide']") private WebElement LoginButton;

public KiteLoginPage (WebDriver driver)

{

PageFactory.initElements(driver, this);

}

public void EnterUserId(String UserName)

{

UserID.sendKeys(UserName);

}

public void EnterPassword(String Password)

{

PWD.sendKeys(Password);

}

public void ClickOnLoginButton()

{

LoginButton.click();

}

}

package KitePOM;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.support.FindBy;

import org.openqa.selenium.support.PageFactory;

public class KitePinPage {

@FindBy(xpath = "//input[@id='pin']") private WebElement Pin;

@FindBy(xpath = "//button[@class='button-orange wide']") private WebElement ContinueButton;

public KitePinPage(WebDriver driver)

{

PageFactory.initElements(driver, this);

}

public void EnterPin(String PinValue)

{

Pin.sendKeys(PinValue);

}

public void ClickOnContinue()

{

ContinueButton.click();

}

}

=====

package KitePOM;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.support.FindBy;

import org.openqa.selenium.support.PageFactory;

public class KiteHomePage {

@FindBy(xpath = "//span[@class='user-id']") private WebElement Userid;

public KiteHomePage(WebDriver driver) {

PageFactory.initElements(driver, this);

}

public void UserIdValidation(String UserName)

{

String ActualUserid = Userid.getText();

String ExpecxtedUserid=UserName;

if(ActualUserid.equals(ExpecxtedUserid))

{

System.out.println("TC passed user id is matching");

}

else {

System.out.println("TC failed user id is not-matching");

}

}

}

Test Class🡪

**package** TestKite;

**import** java.io.FileInputStream;

//import java.io.FileInputStream;

**import** java.io.IOException;

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

**import** org.apache.poi.EncryptedDocumentException;

**import** org.apache.poi.ss.usermodel.Sheet;

**import** org.apache.poi.ss.usermodel.WorkbookFactory;

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

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** org.openqa.selenium.chrome.ChromeOptions;

**import** KitePOM.KiteHomePage;

**import** KitePOM.KiteLoginPage;

**import** KitePOM.KitePinPage;

//import KitePOM.KiteLoginPage1;

**public** **class** KiteTest {

**public** **static** **void** main(String[] args) **throws** EncryptedDocumentException, IOException {

FileInputStream Myfile= **new** FileInputStream("C:\\Users\\user\\Desktop\\mytesting.xlsx");

Sheet Mysheet = WorkbookFactory.*create*(Myfile).getSheet("Sheet4");

String Userid = Mysheet.getRow(0).getCell(0).getStringCellValue();

String Password = Mysheet.getRow(0).getCell(1).getStringCellValue();

**double** PinValue = Mysheet.getRow(0).getCell(2).getNumericCellValue();

String PinValues=Double.*toString*(PinValue);

ChromeOptions options=**new** ChromeOptions();

options.addArguments("--disable-notifications");

//options.addArguments("headless");

System.*setProperty*("webdriver.chrome.driver", "F:\\Velocity\\May-2021 Class\\test2\\chromedriver\_win32 (1)\\chromedriver.exe");

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

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

driver.get("https://kite.zerodha.com/");

KiteLoginPage login= **new** KiteLoginPage(driver);

login.EnterUserId(Userid);

login.EnterPassword(Password);

login.ClickOnLoginButton();

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

KitePinPage Login2= **new** KitePinPage(driver);

Login2.EnterPin(PinValues);

Login2.ClickOnContinue();

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

KiteHomePage Home=**new** KiteHomePage(driver);

Home.UserIdValidation(Userid);

}

}