package pagefactory.facebook;

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 LoginPageNewPageFactory {

WebDriver driver;

public LoginPageNewPageFactory(WebDriver driver1)

{

this.driver= driver1;

}

@FindBy(id="email")

@CacheLookup

WebElement username;

@FindBy(how=How.ID,using="pass")

@CacheLookup

WebElement password;

@FindBy(how=How.XPATH,using="//\*[@id=\'u\_0\_b\']")

@CacheLookup

WebElement loginbutton; //Submit\_button

@FindBy(how=How.LINK\_TEXT,using="Forgot account?")

@CacheLookup

WebElement forgotAccount;

public void login\_FaceBook(String un, String pwd) {

username.sendKeys(un);

password.sendKeys(pwd);

loginbutton.click();

forgotAccount.click();

}

}

\* 1) One good feature about page Factory , you can store all the WebElements in a Cache memory

Say @CacheLookup. So it will not check this WEbElements[forgotAccount,password,username]

each and every time on a web page. It will simply check in the cache memory and it will start processing.

2) So whatever WebElement you want to keep in cache memory, you can use this separate Annotation

3) So Now We have given One Instruction these four WebElement keep and cache. So Performance will be high.

4) It's a just small test cases, so you will get the difference but if you go for long run, you will notice performance

will be high because it will not check this webElements on the web page again. it will take from the cache memory

and it will start processing.

5) Pls be care full while using this because you will use only when you are pretty sure that this

Web Element will not changed.

\*/

/\* 1) This is the which I am creating. This is nothing but pageObject.we will store all the Locators.

\* This is another way to locate Web element. This only work with the PageFactory. Specify the WebElement

\* This is Just one line to identify webElement.So in This is Statement where We are saying find an Element where ID="email".

\* So this is one way .

\* \*

\* 2) The Standard way to use FindBy is using 'HOW'. Now You can specify id here name, Xpath ,CSS ,anything.

\* but you want to take dynamically using the library you have to use one 'How'.

\* It will ask you like which locator you want to use. So 'How' is a Class here, 'Using' is a another String variable,

\* and you need to specify the Exact Value which you want to use.

\* So you need to specify the Exact Id. and

\* 3) Apart from this you need a constructor as well. Constructor having the same name as class name.

\* and This constructor will accepts WebDriver as an argument.So Whatever Web driver we will receive, we will directly pass here.

\* So this Driver will come from PageFactory.So Whatever driver will come from PAgeFActory, We will initialize to the Local Driver

\* So Whatever WEb Driver we will receiver . We will directly pass here. driver1 from PageFactory and driver from Local Driver.

\*

\*

\*/

// This is How we can design PAGE OBJECT MODEL using Page Factory

**package** pomFactoryTestCases;

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

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

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

**import** HelperORUtility.BrowserFactory;

**import** pagefactory.facebook.LoginPageNewPageFactory;

**public** **class** VerifyValidLogin {

@Test

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

{

// This will launch browser and Specific URL

WebDriver driver =BrowserFactory.*startBrowser*("Chrome", "https://www.facebook.com/");

//Created Page object using page Factory

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

//Call the method

login\_page.login\_FaceBook("afsheen", "afsheen123");

}

}

/\* 1) We need to start application, we need to Enter UN and Pwd and we need to verify the title

\* 2) Simple type BrowserFactory. then I am getting the method that I created and you see the argument

\* , it is taking BrowserName, Url and it will return you the webDriver reference.

\* So I will pass the Browser name = Chrome and URl

\* This is meaningful like I have one Browser factory where I am starting a browser with this URL and

\* It will return me the driver.

\*

\* 3) Now my task is to start test cases that is nothing but . we need to initialize our web page.

\* We need to initialize all these web Element . So we have created Four(4) Web Element and

\* I will use three(3).

\* SO THIS IS OUR TEST.

\*

\* \*\*\*Now you need to call PAGEFACTORY CLASS. It is a class . it has so many methods and I will

\* Stick to the LastOne and It is used to INSTANTIATE the page object and it accepts the TWO arguments

\* a) Pass the Driver and b) The object with webElement and List<WebElement> field.

\* 4) PageFactory.initElements(driver, page); it will ask the page which you want to initialize.

\* So our page that we created "LoginPageNewPageFactory" .So I want to Initialize this page.

\* And then you need to use the BINARY class. it will return Byte code of this .

\* and Then it will return you PAGE OBJECT of that particular page ->LoginPageNewPageFactory login\_page.

\* So if you pass LoginPageNewPageFactory , it will return you object of the same page object and you can

\* use it ->login\_page.

\*

\* 5) In case if you are using some other pages . Like I want to initialize LoginPageNewxyz page.

\* So I will say LoginPageNewxyz =PageFactory.initElements(driver, LoginPageNewxyz.class);

\* So this will return page object of the same cllogin\_pageoldass like that i typed in No 5.

\* So Only the point which you need to remember if i pass login\_new then

\* it will return the page object of the same class.

\*

\* 6) Now We need to call one Final method, using this that I created now [login\_page] ,

\* call the method that I created [login\_FaceBook]. it will ask you UN, Pwd. So UN I will

\* give afsheen and pass -afsheen123.

\*

\*

\*

\*

\*

\*/

**package** HelperORUtility;

**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** BrowserFactory { //5

**static** WebDriver *driver*; //5

**public** **static** WebDriver startBrowser(String browserName, String url){ // 4 //5

System.*setProperty*("webdriver.driver.chromedriver", "chromedriver.exe");

**if**(browserName.equalsIgnoreCase("chrome")) {

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

}

**else** **if** (browserName.equalsIgnoreCase("firefox")) {

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

}

**else** **if** (browserName.equalsIgnoreCase("IE")) {

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

}

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

*driver*.get(url);

**return** *driver*;

}

}

/\* TO START A BROWSER, WE CREATED THE BROWSER FACTORY

\* 1) Create a separate Utility then code redundancy will be less.

\* 2) Helper is the Package that I am creating ,you can REname your Utility or Helper and

\* Class name I can give Like Browser FActory.

\* 1) The BrowserFactory will Accept the browser name in a parameter. and

\* Based on your parameter it will trigger the respective browser.

\* So I will simply say-- Public void startBrowser ()and

\* I will accept Browser name as in parameter

\*

\* 2) and I will use one If condition ,This is very important.

\* So I will check if browser equal to "Chrome" then Start Chrome Maximize and Pass the URL

\* 3) Crate Local WebDRiver Variable

\* 4) I will take one more argument from here ,I will say accept the URL as well.

\* So whatever user will pass , it will directly pass here.So it becomes very customize method now.Just

\* pass the url and pass the browser and it will start your application

\* 5) we have created this method as non-static, so let me make this static method .

\* So we need to create webdriver driver as static method

\*/