BaseClass.java:

package com.sep.tesing.automationtesting;

import java.awt.Rectangle;

import java.awt.Robot;

import java.awt.Toolkit;

import java.awt.image.BufferedImage;

import java.io.File;

import javax.imageio.ImageIO;

import org.apache.commons.io.FileUtils;

import org.openqa.selenium.OutputType;

import org.openqa.selenium.TakesScreenshot;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.chrome.ChromeOptions;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.firefox.FirefoxOptions;

public class BaseClass {

public static WebDriver driver;

public static void LaunchBrowser(String BrowserType)

{

System.out.println("In Launch Browser method!!");

if(BrowserType.equalsIgnoreCase("Chrome"))

{

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

ChromeOptions options = new ChromeOptions();

options.addArguments("--start-maximized");

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

options.setExperimentalOption("useAutomationExtension", false);

options.setExperimentalOption("excludeSwitches", new String[] {"enable-automation"});

driver=new ChromeDriver(options);

}

else if(BrowserType.equalsIgnoreCase("Firefox"))

{

System.setProperty("webdriver.gecko.driver", "geckodriver.exe");

FirefoxOptions options = new FirefoxOptions();

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

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

driver=new FirefoxDriver(options);

}

}

public static void takeSnapshot(WebDriver driver,String filesuffix)

{

try {

TakesScreenshot scrShot =((TakesScreenshot)driver);

String filename = "src//test//resources//snapshots//"+filesuffix+".png";

File SrcFile=scrShot.getScreenshotAs(OutputType.FILE);

System.out.println("File name is : "+filesuffix);

File destFile = new File(filename);

System.out.println("Screenshot captured in path: "+filename);

FileUtils.copyFile(SrcFile, destFile);

System.out.println("Screenshot Captured ");

} catch (Exception e) {

e.printStackTrace();

}

}

public static String CaptureScreen(String filename)

{

try{

Robot robotClassObject = new Robot();

Rectangle screenSize = new Rectangle(Toolkit.getDefaultToolkit().getScreenSize());

BufferedImage tmp = robotClassObject.createScreenCapture(screenSize);

String filepath = "src//test//resources//snapshots//"+filename+".png";

ImageIO.write(tmp, "png",new File(filepath));

return filepath;

}catch(Exception e)

{

System.out.println("Some exception occured." + e.getMessage());

return "";

}

}

}

TestRunner.java:

package com.sep.tesing.automationtesting;

import org.junit.runner.RunWith;

import cucumber.api.CucumberOptions;

import cucumber.api.junit.Cucumber;

@RunWith(Cucumber.class)

@CucumberOptions

(

features = {"src/test/resources/features/print.feature"},

//tags={"@tag2"},

glue="com.sep.tesing.automationtesting",

format = {"pretty","html:target/destination"},

monochrome=true

)

public class TestRunner {

}

Print.feature:

@tag

Feature: Print values

print the index of BSE and NSE

@tag1

Scenario: Print the values of BSE and NSE

Given open browser

And navigates to url

When user performs required action

Then successful validation and close browser

@tag2

Scenario: Print the values of BSE and NSE

Given open browser

And navigates to url

When user prints href

Then successful validation and close browser

TestNG.xml:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >

<suite name=*"Suite"*>

<test name=*"Test"*>

<classes>

<class name=*"com.sep.tesing.automationtesting.AppTest"*>

<methods>

<include name=*"test1"* />

<include name=*"test2"* />

</methods>

</class>

</classes>

</test><!-- Test -->

</suite><!-- Suite -->

pom.xml:

<project xmlns=*"http://maven.apache.org/POM/4.0.0"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>Seltest</groupId>

<artifactId>Automation</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>Automation</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-simple</artifactId>

<version>1.7.7</version>

</dependency>

<dependency>

<groupId>com.codeborne</groupId>

<artifactId>selenide</artifactId>

<version>5.7.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.assertj</groupId>

<artifactId>assertj-core</artifactId>

<version>3.6.2</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.hamcrest</groupId>

<artifactId>hamcrest-all</artifactId>

<version>1.3</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>com.github.mkolisnyk</groupId>

<artifactId>cucumber-reports</artifactId>

<version>0.0.2</version>

</dependency>

<!-- https://mvnrepository.com/artifact/com.github.mkolisnyk/cucumber-runner -->

<dependency>

<groupId>com.github.mkolisnyk</groupId>

<artifactId>cucumber-runner</artifactId>

<version>1.0.8</version>

</dependency>

<!-- https://mvnrepository.com/artifact/info.cukes/cucumber-jvm -->

<dependency>

<groupId>info.cukes</groupId>

<artifactId>cucumber-jvm</artifactId>

<version>1.2.5</version>

<type>pom</type>

</dependency>

<!-- https://mvnrepository.com/artifact/com.jramoyo/indexed-file-reader-java6 -->

<dependency>

<groupId>com.jramoyo</groupId>

<artifactId>indexed-file-reader-java6</artifactId>

<version>1.0</version>

</dependency>

<dependency>

<groupId>info.cukes</groupId>

<artifactId>cucumber-picocontainer</artifactId>

<version>1.2.5</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>info.cukes</groupId>

<artifactId>cucumber-core</artifactId>

<version>1.2.5</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>info.cukes</groupId>

<artifactId>cucumber-java</artifactId>

<version>1.2.5</version>

</dependency>

<dependency>

<groupId>info.cukes</groupId>

<artifactId>cucumber-junit</artifactId>

<version>1.2.5</version>

</dependency>

<!-- These are the selenium-webdriver dependencies -->

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>3.9.1</version>

</dependency>

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-server</artifactId>

<version>3.6.0</version>

</dependency>

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-firefox-driver</artifactId>

<version>3.6.0</version>

</dependency>

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-remote-driver</artifactId>

<version>3.6.0</version>

</dependency>

<dependency>

<groupId>info.cukes</groupId>

<artifactId>gherkin</artifactId>

<version>2.12.2</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-htmlunit-driver</artifactId>

<version>2.20.0</version>

</dependency>

<dependency>

<groupId>org.yaml</groupId>

<artifactId>snakeyaml</artifactId>

<version>1.13</version>

</dependency>

<dependency>

<groupId>com.esotericsoftware.yamlbeans</groupId>

<artifactId>yamlbeans</artifactId>

<version>1.06</version>

</dependency>

<dependency>

<groupId>com.google.guava</groupId>

<artifactId>guava</artifactId>

<version>23.0</version>

</dependency>

<dependency>

<groupId>com.codoid.products</groupId>

<artifactId>fillo</artifactId>

<version>1.18</version>

</dependency>

<!-- https://mvnrepository.com/artifact/com.vimalselvam/cucumber-extentsreport -->

<dependency>

<groupId>com.vimalselvam</groupId>

<artifactId>cucumber-extentsreport</artifactId>

<version>3.0.2</version>

</dependency>

<!-- https://mvnrepository.com/artifact/com.aventstack/extentreports -->

<dependency>

<groupId>com.aventstack</groupId>

<artifactId>extentreports</artifactId>

<version>4.0.8</version>

</dependency>

<!-- https://mvnrepository.com/artifact/net.masterthought/cucumber-reporting -->

<dependency>

<groupId>net.masterthought</groupId>

<artifactId>cucumber-reporting</artifactId>

<version>3.5.1</version>

</dependency>

<dependency>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

<type>maven-plugin</type>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-surefire-plugin -->

<dependency>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.18.1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-surefire-report-plugin -->

<dependency>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-report-plugin</artifactId>

<version>2.20</version>

</dependency>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.17</version>

</dependency>

</dependencies>

</project>

AppPrint.java:

**package** com.sep.tesing.automationtesting;

**import** java.util.Set;

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

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

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

**import** com.app.objects.pageobjs;

**import** com.codoid.products.fillo.Connection;

**import** com.codoid.products.fillo.Fillo;

**import** com.codoid.products.fillo.Recordset;

**import** cucumber.api.java.en.Given;

**import** cucumber.api.java.en.Then;

**import** cucumber.api.java.en.When;

**import** junit.framework.~~Assert~~;

**public** **class** AppPrint **extends** BaseClass{

@Given("^open browser$")

**public** **void** open\_browser() {

**try** {

*LaunchBrowser*("Chrome");

} **catch** (Exception e) {

e.printStackTrace();

}

}

@Given("^navigates to url$")

**public** **void** navigates\_to\_url() {

**try** {

*driver*.get("https://www.rediff.com");

} **catch** (Exception e) {

e.printStackTrace();

}

}

@SuppressWarnings("deprecation")

@When("^user performs required action$")

**public** **void** user\_performs\_required\_action() {

**try** {

PageFactory.*initElements*(*driver*, pageobjs.**class**);

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

// WebDriverWait wait = new WebDriverWait(driver,20);

// wait.until(ExpectedConditions.visibilityOf(pageobjs.bse));

*driver*.switchTo().frame(pageobjs.*frame*);

Thread.*sleep*(3000);;

String bsevalue=pageobjs.*bse*.getText();

System.***out***.println("The index value of BSE is : "+bsevalue);

String nsevalue=pageobjs.*nse*.getText();

System.***out***.println("The index value of NSE is : "+nsevalue);

String pwin = *driver*.getWindowHandle();

System.***out***.println("Rediff win handle"+pwin);

pageobjs.*company*.sendKeys("HDFC Bank Ltd.");

pageobjs.*search*.click();

Thread.*sleep*(5000);

Set<String> windows = *driver*.getWindowHandles();

System.***out***.println(windows);

System.***out***.println("a1l window");

**for** (String window : windows)

{

*driver*.switchTo().window(window);

}

String htitle = *driver*.getTitle().substring(0, 14);;

System.***out***.println("The HDFC bank title is : "+htitle);

~~Assert~~.~~assertEquals~~("HDFC BANK LTD.", htitle);

System.***out***.println("The HDFC bank window title asserted");

*driver*.switchTo().window(pwin);

String ptitle=*driver*.getTitle().substring(0, 11);

~~Assert~~.~~assertEquals~~("Rediff.com:", ptitle);

System.***out***.println("The rediff window title is : "+ptitle);

System.***out***.println("The rediff window title asserted");

} **catch** (Exception e) {

e.printStackTrace();

}

}

@When("^user prints href$")

**public** **void** user\_prints\_href() {

**try** {

Fillo fillo = **new** Fillo();

Connection connection = fillo.getConnection("src/test/resources/TestData/testdata.xlsx");

System.***out***.println("Connection Established");

String strQuery = "select \* from href";

Recordset recordset = connection.executeQuery(strQuery);

**int** count = recordset.getCount();

System.***out***.println("The total count of rows is : "+count);

Thread.*sleep*(3000);

**for**(**int** i=1;i<=count;i++)

{

recordset.moveNext();

String tcid=recordset.getField("TopStories");

System.***out***.println("TC\_ID is : "+tcid);

Thread.*sleep*(2000);

String href = *driver*.findElement(By.*xpath*("//div[@id='topdiv\_0']/h2["+i+"]/following::a[1]")).getAttribute("href");

System.***out***.println("Iteration "+i+" :"+href);

String updateQuery = "Update href Set Href='"+href+"' where TopStories='"+tcid+"'";

System.***out***.println(updateQuery);

connection.executeUpdate(updateQuery);

System.***out***.println("Excel sheet updated");

}

}

**catch** (Exception e) {

e.printStackTrace();

}

}

@Then("^successful validation and close browser$")

**public** **void** successful\_validation\_and\_close\_browser() {

**try** {

Thread.*sleep*(3000);

*driver*.quit();

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

AppTest.java:

package com.sep.tesing.automationtesting;

import java.awt.Robot;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.Point;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.interactions.Action;

import org.openqa.selenium.interactions.Actions;

import org.testng.Assert;

import org.testng.annotations.AfterClass;

import org.testng.annotations.AfterMethod;

import org.testng.annotations.AfterSuite;

import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.BeforeSuite;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

import com.codoid.products.fillo.Connection;

import com.codoid.products.fillo.Fillo;

import com.codoid.products.fillo.Recordset;

public class AppTest extends BaseClass

{

@Test

public static void test1()

{

try {

driver.findElement(By.xpath("//a[.='MOVIES']")).click();

driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);

takeSnapshot(driver,"MOVIES");

String headlines = driver.findElement(By.xpath("//a[.='Headlines']")).getAttribute("href");

System.out.println("The href of headlines is : "+headlines );

String images = driver.findElement(By.xpath("//a[.='Images']")).getAttribute("href");

System.out.println("The href of images is : "+images );

String interviews = driver.findElement(By.xpath("//a[.='Interviews']")).getAttribute("href");

System.out.println("The href of images is : "+interviews );

String reviews = driver.findElement(By.xpath("//a[.='Reviews']")).getAttribute("href");

System.out.println("The href of reviews is : "+reviews );

String webseries = driver.findElement(By.xpath("//a[.='Web Series']")).getAttribute("href");

System.out.println("The href of webseries is : "+webseries );

String television = driver.findElement(By.xpath("//a[.='Television']")).getAttribute("href");

System.out.println("The href of television is : "+television );

String videos = driver.findElement(By.xpath("//a[.='Videos']")).getAttribute("href");

System.out.println("The href of videos is : "+videos );

String southcinema = driver.findElement(By.xpath("//a[.='South Cinema']")).getAttribute("href");

System.out.println("The href of southcinema is : "+southcinema );

WebElement holly = driver.findElement(By.xpath("//div[@class='subnavbar movies']"));

String hollywood = holly.getAttribute("href");

System.out.println("The href of hollywood is : "+hollywood );

String starsspotted = driver.findElement(By.xpath("//a[.='Stars Spotted']")).getAttribute("href");

System.out.println("The href of starsspotted is : "+starsspotted );

String expText="Cricket headlines";

WebElement act = driver.findElement(By.xpath("//a[.='CRICKET']"));

String actText = act.getAttribute("title");

Assert.assertEquals(actText, expText);

System.out.println("Actual Text is equal to Expected Text!!!");

Thread.sleep(10000);

WebElement tooltip = driver.findElement(By.xpath("//a[.='CRICKET']"));

Robot robot;

robot = new Robot();

Point point;

point = tooltip.getLocation();

int x = point.getX();

int y = point.getY();

System.out.println(x+","+y);

robot.mouseMove(561,285);

Actions builder = new Actions(driver);

Action acti = builder.moveToElement(tooltip).build();

acti.perform();

Thread.sleep(5000);

CaptureScreen("Cricket Headlines tool tip");

acti = builder.release(tooltip).build();

acti.perform();

Thread.sleep(4000);

driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);

Thread.sleep(3000);

String bgr = holly.getCssValue("background");

System.out.println("The Background color is : "+bgr);

String bgrsub = bgr.substring(0, 16);

String expbgr = "rgb(199, 49, 89)";

Assert.assertEquals(bgrsub, expbgr);

{

System.out.println("The color is 'Old Rose'");

}

String bgrclr = holly.getCssValue("background-color");

System.out.println("The Background color is : "+bgrclr);

String color = holly.getCssValue("color");

System.out.println("The color is "+color);

} catch (Exception e) {

e.printStackTrace();

}

}

@Test

public void test2()

{

try {

driver.findElement(By.xpath("//a[.='MOVIES']")).click();

driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);

WebElement head = driver.findElement(By.xpath("//a[.='Headlines']"));

String actheadlines = head.getAttribute("href");

WebElement img = driver.findElement(By.xpath("//a[.='Images']"));

String actimages = img.getAttribute("href");

WebElement inter = driver.findElement(By.xpath("//a[.='Interviews']"));

String actinterviews = inter.getAttribute("href")+".html";

WebElement rev = driver.findElement(By.xpath("//a[.='Reviews']"));

String actreviews = rev.getAttribute("href")+".html";

Fillo fillo = new Fillo();

Connection connection = fillo.getConnection("src/test/resources/TestData/testdata.xlsx");

String strquery = "Select \* from data";

Recordset recordset = connection.executeQuery(strquery);

int count = recordset.getCount();

System.out.println("Count ="+count);

for (int i=0;i<count;i++)

{

recordset.moveNext();

String Menu = recordset.getField("Menu");

String SubMenu = recordset.getField("SubMenu");

String Href = recordset.getField("Href");

System.out.println("Iteration"+i);

System.out.println("Menu : "+Menu+"\n"+"SubMenu : "+SubMenu);

if(Menu.equalsIgnoreCase("Movies") && SubMenu.equalsIgnoreCase("Headline"))

{

System.out.println("Inside condition 1");

System.out.println("Actual : "+actheadlines+"\n"+"Expected :"+Href);

Assert.assertEquals(actheadlines, Href);

System.out.println("The href of headlines submenu is same as in the excel");

head.click();

Thread.sleep(2000);

takeSnapshot(driver,"Question2\_Headline");

driver.navigate().back();

Thread.sleep(10000);

}

else if(Menu.equalsIgnoreCase("Movies") && SubMenu.equalsIgnoreCase("Images"))

{

System.out.println("Inside condition 2");

System.out.println("Actual : "+actimages+"\n"+"Expected :"+Href);

Assert.assertEquals(actimages, Href);

System.out.println("The href of images submenu is same as in the excel");

driver.findElement(By.xpath("//a[.='Images']")).click();

Thread.sleep(2000);

takeSnapshot(driver,"Question2\_Images");

driver.navigate().back();

}

else if(Menu.equalsIgnoreCase("Movies") && SubMenu.equalsIgnoreCase("Interviews"))

{

System.out.println("Inside condition 3");

System.out.println("Actual : "+actinterviews+"\n"+"Expected :"+Href);

Assert.assertEquals(actinterviews, Href);

System.out.println("The href of interviews submenu is same as in the excel");

driver.findElement(By.xpath("//a[.='Interviews']")).click();

Thread.sleep(2000);

takeSnapshot(driver,"Question2\_Interviews");

driver.navigate().back();

}

else if(Menu.equalsIgnoreCase("Movies") && SubMenu.equalsIgnoreCase("Reviews"))

{

System.out.println("Inside condition 4");

System.out.println("Actual : "+actreviews+"\n"+"Expected :"+Href);

Assert.assertEquals(actreviews, Href);

System.out.println("The href of reviews submenu is same as in the excel");

driver.findElement(By.xpath("//a[.='Reviews']")).click();

Thread.sleep(2000);

takeSnapshot(driver,"Question2\_Reviews");

}

}

} catch (Exception e) {

e.printStackTrace();

}

}

@BeforeSuite

public void befSuite()

{

try {

LaunchBrowser("Chrome");

} catch (Exception e) {

e.printStackTrace();

}

}

@BeforeTest

public void befTest()

{

try {

System.out.println("Before Test");

} catch (Exception e) {

e.printStackTrace();

}

}

@BeforeClass

public void befClass()

{

try {

System.out.println("Before class");

} catch (Exception e) {

e.printStackTrace();

}

}

@BeforeMethod

public void befMethod()

{

try {

driver.get("https://www.rediff.com");

System.out.println("Before method");

} catch (Exception e) {

e.printStackTrace();

}

}

@AfterMethod

public void aftMethod()

{

try {

//driver.close();

System.out.println("After method");

} catch (Exception e) {

e.printStackTrace();

}

}

@AfterClass

public void aftClass()

{

try {

System.out.println("After Class");

} catch (Exception e) {

e.printStackTrace();

}

}

@AfterTest

public void aftTest()

{

try {

System.out.println("After Test");

} catch (Exception e) {

e.printStackTrace();

}

}

@AfterSuite

public void aftSuite()

{

try {

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

driver.quit();

} catch (Exception e) {

e.printStackTrace();

}

}

}