package com.cts.reusable;

//@AUTHOR = RUSHI

import java.io.File;

import java.io.IOException;

import java.util.Set;

import java.util.concurrent.TimeUnit;

import org.apache.commons.io.FileUtils;

import org.openqa.selenium.Alert;

import org.openqa.selenium.By;

import org.openqa.selenium.JavascriptExecutor;

import org.openqa.selenium.NoAlertPresentException;

import org.openqa.selenium.NoSuchElementException;

import org.openqa.selenium.OutputType;

import org.openqa.selenium.TakesScreenshot;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.interactions.Actions;

import org.openqa.selenium.support.ui.Select;

import org.testng.Reporter;

public class ActionDriver {

static WebDriver driver;

/\*\*

\* To launch browser

\*

\* //@AUTHOR = RUSHI

\*

\*/

public static void launchBrowser()

{

driver=new FirefoxDriver();

}

public static void windowMaximize(){

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

}

/\*\*

\* To launch URL

\* //@AUTHOR = RUSHI

\* @param url: url value want to launch

\*

\*/

public static void launchUrl(String url) {

driver.navigate().to(url);

}

/\*\*@AUTHOR = RUSHI

\*

\* @param locator: Action to be performed on element (Get it from Object repository)

\* @param locatorName: Meaningful name to the element (Ex:Login Button, SignIn Link etc..)

\* @return --boolean (true or false)

\* @throws Throwable

\*/

public static boolean click(By locator, String locatorName)

throws Throwable {

boolean flag = false;

try {

//ImplicitWait();

driver.findElement(locator).click();

return true;

} catch (Exception e) {

System.out.println(e.getMessage());

flag = true;

return false;

} finally {

if (flag) {

Reporter.log("Unable to click on :"+locatorName);

}

else

{

Reporter.log("able to click on :"+locatorName);

}

}

}

/\*\* @AUTHOR = RUSHI

\* This method returns check existence of element

\*

\* @param locator: Action to be performed on element (Get it from Object repository)

\* @param locatorName : Meaningful name to the element (Ex:Textbox, checkbox etc)

\* @return: Boolean value(True or False)

\* @throws NoSuchElementException

\*/

public static boolean isElementPresent(By by, String locatorName)

throws Throwable {

boolean flag = false;

try {

driver.findElement(by);

flag = true;

return true;

} catch (NoSuchElementException e) {

System.out.println(e.getMessage());

return false;

} finally {

if (flag) {

Reporter.log("check IsElementPresent" +locatorName);

} else {

Reporter.log("Element is not Present on the page" +locatorName);

}

}

}

/\*\* @AUTHOR = RUSHI

\* This method used type value in to text box or text area

\*

\* @param locator: Action to be performed on element (Get it from Object repository)

\*

\* @param testdata: Value wish to type in text box / text area

\*

\* @param locatorName: Meaningful name to the element (Ex:Textbox,Text Area etc..)

\*

\* @throws NoSuchElementException

\*/

public static boolean type(By locator, String testdata, String locatorName)

throws Throwable {

boolean flag = false;

try {

//ImplicitWait();

driver.findElement(locator).clear();

driver.findElement(locator).sendKeys(testdata);

flag = true;

return true;

} catch (NoSuchElementException e) {

e.printStackTrace();

return false;

} finally {

if (flag) {

Reporter.log("Data typing action is performed on:" +locatorName+" with data is "+testdata);

} else {

Reporter.log("Data typing action is not performed on:" +locatorName+" with data is "+testdata);

}

}

}

/\*\* @AUTHOR = RUSHI

\* Moves the mouse to the middle of the element. The element is scrolled

\* into view and its location is calculated using getBoundingClientRect.

\*

\* @param locator: Action to be performed on element (Get it from Object repository)

\*

\* @param locatorName: Meaningful name to the element (Ex:link,menus etc..)

\*

\*/

public static boolean mouseover(By locator, String locatorName)

throws Throwable {

boolean flag = false;

try {

WebElement mo = driver.findElement(locator);

new Actions(driver).moveToElement(mo).build().perform();

return true;

} catch (NoSuchElementException e) {

flag = true;

return false;

} finally {

if (flag) {

Reporter.log("MouseOver action is not perform on:" + locatorName);

} else {

Reporter.log("MouseOver action is done perform on:" + locatorName);

}

}

}

/\*\* @AUTHOR = RUSHI

\* A convenience method that performs click-and-hold at the location of the

\* source element, moves by a given offset, then releases the mouse.

\*

\* @param source: Element to emulate button down at.

\*

\* @param xOffset: Horizontal move offset.

\*

\* @param yOffset: Vertical move offset.

\*

\*/

public static void draggable(By source, int x, int y, String locatorName)

throws Throwable {

boolean flag = false;

try {

WebElement dragitem = driver.findElement(source);

new Actions(driver).dragAndDropBy(dragitem, x, y).build().perform();

Thread.sleep(5000);

flag = true;

} catch (NoSuchElementException e) {

} finally {

if (flag) {

Reporter.log(" Draggable action is done on:" + locatorName);

} else {

Reporter.log(" Draggable action is not done on:" + locatorName);

}

}

}

/\*\* @AUTHOR = RUSHI

\* A convenience method that performs click-and-hold at the location of the

\* source element, moves to the location of the target element, then

\* releases the mouse.

\*

\* @param source: Element to emulate button down at.

\*

\* @param target: Element to move to and release the mouse at.

\*

\* @param locatorName: Meaningful name to the element (Ex:Button,image etc..)

\*

\*/

public static void draganddrop(By source, By target, String locatorName)

throws Throwable {

boolean flag = false;

try {

WebElement from = driver.findElement(source);

WebElement to = driver.findElement(target);

new Actions(driver).dragAndDrop(from, to).perform();

} catch (Exception e) {

flag = true;

} finally {

if (flag) {

Reporter.log(" DragAndDrop action is not perform on:" + locatorName);

} else {

Reporter.log(" DragAndDrop action is perform on:" + locatorName);

}

}

}

/\*\* @AUTHOR = RAMESH

\* To right click on an element

\*

\* @param by: Action to be performed on element (Get it from Object repository)

\*

\* @param locatorName: Meaningful name to the element (Ex:Login Button, SignIn Link etc..)

\*

\* @throws Throwable

\*/

public static void rightclick(By by, String locatorName) throws Throwable {

boolean flag = false;

try {

WebElement elementToRightClick = driver.findElement(by);

Actions clicker = new Actions(driver);

clicker.contextClick(elementToRightClick).perform();

// driver.findElement(by1).sendKeys(Keys.DOWN);

} catch (Exception e) {

flag = true;

} finally {

if (flag) {

Reporter.log(" RightClick action is not perform on:" + locatorName);

} else {

Reporter.log(" RightClick action is perform on:" + locatorName);

}

}

}

/\*\* @AUTHOR = RUSHI

\* Wait for an element

\*

\* @param locator: Action to be performed on element (Get it from Object repository)

\*

\*/

public static void waitForTitlePresent(By locator) throws Throwable

{

for (int i = 0; i < 200; i++)

{

if (driver.findElements(locator).size()>0)

{

break;

} else

{

Thread.sleep(50);

}

{

try {

driver.wait(5000);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

/\*\* @AUTHOR = RUSHI

\* Wait for an ElementPresent

\*

\* @param locator: Action to be performed on element (Get it from Object repository)

\*

\* @return Whether or not the element is displayed

\*/

public static void waitForElementPresent(By locator) throws Throwable {

for (int i = 0; i < 200; i++) {

if (driver.findElement(locator).isDisplayed()) {

break;

} else {

Thread.sleep(50);

}

{

/\*try {

driver.wait(5000);

} catch (InterruptedException e) {

e.printStackTrace();

}\*/

}

}

}

public static boolean selectByIndex(By locator, int index, String locatorName)

throws Throwable {

boolean flag = false;

try {

Select s = new Select(driver.findElement(locator));

s.selectByIndex(index);

return true;

} catch (Exception e) {

flag = true;

return false;

} finally {

if (flag) {

Reporter.log(" Select Option at index "+index+" is Not Select from the DropDown:" + locatorName);

} else {

Reporter.log(" Select Option at index "+index+" is Selected from the DropDown:" + locatorName);

}

}

}

/\*\* @AUTHOR = RUSHI

\* select value from DD by using value

\*

\* @param locator: Action to be performed on element (Get it from Object repository)

\*

\* @param value: Value wish to select from dropdown list.

\*

\* @param locatorName: Meaningful name to the element (Ex:Year Dropdown, items Listbox etc..)

\*/

public void selectByValue(By locator, String value, String locatorName)

throws Throwable {

boolean flag = false;

try {

Select s = new Select(driver.findElement(locator));

s.selectByValue(value);

} catch (Exception e) {

flag = true;

} finally {

if (flag) {

Reporter.log(" Select Option with value attribute "+value+" is Not Select from the DropDown:" + locatorName);

} else {

Reporter.log(" Select Option with value attribute "+value+" is Selected from the DropDown:" + locatorName);

}

}

}

/\*\* @AUTHOR = RUSHI

\* select value from DropDown by using selectByVisibleText

\*

\* @param locator: Action to be performed on element (Get it from Object repository)

\*

\* @param visibletext: VisibleText wish to select from dropdown list.

\*

\* @param locatorName: Meaningful name to the element (Ex:Year Dropdown, items Listbox etc..)

\*/

public void selectByVisibleText(By locator, String visibletext,

String locatorName) throws Throwable {

boolean flag = false;

try {

Select s = new Select(driver.findElement(locator));

s.selectByVisibleText(visibletext);

} catch (Exception e) {

flag = true;

} finally {

if (flag) {

Reporter.log("Select "+visibletext+" is Not Select from the DropDown:" + locatorName);

} else {

Reporter.log("Select "+visibletext+" is Selected from the DropDown:" + locatorName);

}

}

}

/\*\* @AUTHOR = RUSHI

\* SWITCH TO WINDOW BY USING TITLE

\*

\* @param windowTitle: Title of window wish to switch

\*

\* @param count: Selenium launched Window id (integer no)

\*

\* @return: Boolean value(true or false)

\*

\*/

//

public Boolean switchWindowByTitle(String windowTitle, int count)

throws Throwable {

boolean flag = false;

try {

Set<String> windowList = driver.getWindowHandles();

int windowCount = windowList.size();

// Calendar calendar = new GregorianCalendar();

// int second = calendar.get(Calendar.SECOND); // /to get current time

// int timeout = second + 40;

/\* while (windowCount != count && second < timeout) {

Thread.sleep(500);

windowList = driver.getWindowHandles();

windowCount = windowList.size();

}\*/

String[] array = windowList.toArray(new String[0]);

for (int i = 0; i <=windowCount; i++) {

driver.switchTo().window(array[count-1]);

if (driver.getTitle().contains(windowTitle))

return true;

}

return false;

} catch (Exception e) {

flag = true;

return false;

} finally {

if (flag) {

Reporter.log("SelectWindow The Window with title:" + windowTitle

+ "is not Selected");

} else {

Reporter.log("SelectWindow The Window with title:" + windowTitle

+ "is Selected");

}

}

}

/\*\* @AUTHOR = RUSHI

\* Verify alert present or not

\*

\* @return: Boolean (True: If alert preset, False: If no alert)

\*

\*/

public boolean Alert() throws Throwable {

boolean presentFlag = false;

Alert alert = null;

try {

// Check the presence of alert

alert = driver.switchTo().alert();

// Alert present; set the flag

presentFlag = true;

// if present consume the alert

alert.accept();

} catch (NoAlertPresentException ex) {

// Alert not present

ex.printStackTrace();

} finally {

if (presentFlag) {

Reporter.log("There was alert to handle");

} else {

Reporter.log("There was no alert to handle");

}

}

return presentFlag;

}

/\*\* @AUTHOR = RUSHI

\* This method verify check box is checked or not

\*

\* @param locator: Action to be performed on element (Get it from Object repository)

\*

\* @param locatorName: Meaningful name to the element (Ex:sign in Checkbox etc..)

\*

\* @return: boolean value(True: if it is checked, False: if not checked)

\*

\*/

public static boolean isChecked(By locator, String locatorName) throws Throwable {

boolean value = false;

try {

if (driver.findElement(locator).isSelected()) {

value = true;

}

return value;

} catch (NoSuchElementException e) {

return value;

} finally {

if (value) {

Reporter.log("IsChecked "+locatorName+ " is Selected");

} else {

Reporter.log("IsChecked "+locatorName+ " is not Selected");

}

}

}

/\*\* @AUTHOR = RUSHI

\* Element is enable or not

\*

\* @param locator: Action to be performed on element (Get it from Object repository)

\*

\* @param locatorName: Meaningful name to the element (Ex:Login Button, UserName Textbox etc..)

\*

\* @return: boolean value (True: if the element is enabled, false: if it not enabled).

\*

\*/

public static Boolean isEnabled(By locator, String locatorName) throws Throwable {

Boolean value = false;

try {

if (driver.findElement(locator).isEnabled()) {

value = true;

}

return value;

} catch (NoSuchElementException e) {

return value;

} finally {

if (value) {

Reporter.log("IsEnabled"+locatorName+ ": is Enabled");

} else {

Reporter.log("IsEnabled"+locatorName+ ": is not Enabled");

}

}

}

/\*\* @AUTHOR = RUSHI

\* Element visible or not

\*

\* @param locator: Action to be performed on element (Get it from Object repository)

\*

\* @param locatorName: Meaningful name to the element (Ex:Login Button, SignIn Link etc..)

\*

\* @return: boolean value(True: if the element is visible, false: If element not visible)

\*

\*/

public Boolean isVisible(By locator, String locatorName) throws Throwable {

Boolean value = false;

try {

value = driver.findElement(locator).isDisplayed();

return value;

} catch (NoSuchElementException e) {

return value;

} finally {

if (value) {

Reporter.log("IsVisible" +locatorName+ ": Element is visible");

} else {

Reporter.log("IsVisible" +locatorName+ ": Element is not visible");

}

}

}

/\*\* @AUTHOR = RUSHI

\* @return: return title of current page.

\*

\* @throws Throwable

\*/

public static String getTitle() throws Throwable {

String text = driver.getTitle();

Reporter.log("Title of the page is:" + text);

return text;

}

/\*\*

\* Verify Title of the page.

\*

\* @param title: Expected title of the page.

\*

\*/

public static boolean verifyTitle(String title) throws Throwable {

if(driver.getTitle().equals(title))

{

Reporter.log("Page title is verified with "+title);

return true;

}

else

{

Reporter.log("Page title is not matched with "+title);

return false;

}

}

/\*\* @AUTHOR = RUSHI

\* Verify text present or not

\*

\* @param text: Text wish to verify on the current page.

\*

\*/

public static void verifyTextPresent(String text) throws Throwable

{

if((driver.getPageSource()).contains(text))

{

Reporter.log(text+"is present in the page");

}

else

{

Reporter.log(text+"is not present in the page");

}

}

/\*\* @AUTHOR = RUSHI

\* Verify the 404 error or broken links

\*

\* @param text: Text expect to display when 404 error / broken link.

\*

\*/

public static boolean verify404(String text) throws Throwable

{

if (driver.getPageSource().contains("404")) {

Reporter.log(text+"is present in the page");

return true;

}

else

{

Reporter.log(text+"is not present in the page");

return false;

}

}

/\*\* @AUTHOR = RUSHI

\* Capture Screenshot

\*

\* @param fileName: FileName screenshot save in local directory

\*

\*/

public static void screenShot(String fileName) {

File scrFile = ((TakesScreenshot) driver)

.getScreenshotAs(OutputType.FILE);

try {

// Now you can do whatever you need to do with it, for example copy

// somewhere

FileUtils.copyFile(scrFile, new File(fileName));

} catch (IOException e) {

e.printStackTrace();

}

}

/\*\* @AUTHOR = RUSHI

\* Click on the Link

\*

\* @param locator: Action to be performed on element (Get it from Object repository)

\*

\* @param locatorName: Meaningful name to the element (Ex:SignIn Link, menu's etc..)

\*/

public static void mouseHoverByJavaScript(By by, String locatorName)

throws Throwable {

boolean flag = false;

try {

driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);

WebElement mo = driver.findElement(by);

String javaScript = "var evObj = document.createEvent('MouseEvents');"

+ "evObj.initMouseEvent(\"mouseover\",true, false, window, 0, 0, 0, 0, 0, false, false, false, false, 0, null);"

+ "arguments[0].dispatchEvent(evObj);";

JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript(javaScript, mo);

}

catch (Exception e) {

e.printStackTrace();

flag = true;

} finally {

if (flag) {

Reporter.log("MouseOver action is not perform on:" + locatorName);

} else {

Reporter.log("MouseOver action is performed on:" + locatorName);

}

}

}

/\*\* @AUTHOR = RUSHI

\* This method switch the focus to selected frame using frame index

\*

\* @param index: Index of frame wish to switch

\*

\*/

public static void switchToFrameByIndex(int index) throws Throwable

{ boolean flag=false;

try

{

driver.switchTo().frame(index);

flag=true;

}

catch(Exception e)

{

flag=false;

}

finally

{

if(flag)

{

Reporter.log("SelectFrame with index "+index+"is selected");

}

else

{

Reporter.log("SelectFrame with index "+index+"is not selected");

}

}

}

/\*\* @AUTHOR = RUSHI

\* This method switch the to frame using frame ID.

\*

\* @param idValue: Frame ID wish to switch

\*

\*/

public static void switchToFrameById(String idValue)throws Throwable

{ boolean flag=false;

try

{

driver.switchTo().frame(idValue);

flag=true;

}

catch(Exception e)

{

}

finally

{

if(flag)

{

Reporter.log("SelectFrame with ID "+idValue+"is selected");

}

else

{

Reporter.log("SelectFrame with ID "+idValue+"is not selected");

}

}

}

/\*\* @AUTHOR = RUSHI

\* This method switch the to frame using frame Name.

\*

\* @param nameValue: Frame Name wish to switch

\*

\*/

public static void switchToFrameByName(String nameValue)throws Throwable

{ boolean flag=false;

try

{

driver.switchTo().frame(nameValue);

flag=true;

}

catch(Exception e)

{

}

finally

{

if(flag)

{

Reporter.log("SelectFrame with Name "+nameValue+"is selected");

}

else

{

Reporter.log("SelectFrame with Name "+nameValue+"is not selected");

}

}

}

/\*\* @AUTHOR = RUSHI

\* This method switch the to frame using frame Name.

\*

\* @param nameValue: Frame Name wish to switch

\*

\* @param locatorName : Meaningful name to the element (Ex:SignIn Link, login button etc..)

\*

\*

\*/

public static void switchToFrameByLocator(By locacator,String locatorName)throws Throwable

{ boolean flag=false;

try

{

driver.switchTo().frame(driver.findElement(locacator));

flag=true;

}

catch(Exception e)

{

e.printStackTrace();

}

finally

{

if(flag)

{

Reporter.log("SelectFrame with Name "+locatorName+"is selected");

}

else

{

Reporter.log("SelectFrame with Name "+locatorName+"is not selected");

}

}

}

/\*\* @AUTHOR = RUSHI

\* This method wait selenium until element present on web page.

\*/

public static void ImplicitWait(){

driver.manage().timeouts().implicitlyWait(40, TimeUnit.SECONDS);

}

public static void switchToFormByLocator(By locacator,String locatorName)throws Throwable

{ boolean flag=false;

try

{

driver.findElement(locacator);

flag=true;

}

catch(Exception e)

{

e.printStackTrace();

}

finally

{

if(flag)

{

Reporter.log("SelectFrame with Name "+locatorName+"is selected");

}

else

{

Reporter.log("SelectFrame with Name "+locatorName+"is not selected");

}

}

}

}