package tek.sdet.framework.utilities;

import java.time.Duration;

import java.util.Set;

import org.openqa.selenium.By;

import org.openqa.selenium.JavascriptExecutor;

import org.openqa.selenium.Keys;

import org.openqa.selenium.NoSuchElementException;

import org.openqa.selenium.OutputType;

import org.openqa.selenium.StaleElementReferenceException;

import org.openqa.selenium.TakesScreenshot;

import org.openqa.selenium.TimeoutException;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.interactions.Actions;

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

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

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

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

import tek.sdet.framework.base.BaseSetup;

public class CommonUtility extends BaseSetup {

public WebDriverWait getWait() {

return new WebDriverWait(getDriver(), Duration.ofSeconds(20));

}

public WebElement waitTillClickable(WebElement element) {

return this.getWait().until(ExpectedConditions.elementToBeClickable(element));

}

public WebElement waitTillClickable(By by) {

return this.getWait().until(ExpectedConditions.elementToBeClickable(by));

}

public WebElement waitTillPresence(WebElement element) {

return this.getWait().until(ExpectedConditions.visibilityOf(element));

}

public WebElement waitTillPresence(By by) {

return this.getWait().until(ExpectedConditions.visibilityOfElementLocated(by));

}

public void click(WebElement element) {

this.waitTillClickable(element).click();

}

public void sendText(WebElement element, String value) {

this.waitTillPresence(element).sendKeys(value);

}

public String getElementText(WebElement element) {

return this.waitTillPresence(element).getText();

}

public byte[] takeScreenShotAsBytes() {

return ((TakesScreenshot) getDriver()).getScreenshotAs(OutputType.BYTES);

}

public void click(By by) {

waitTillClickable(by).click();

}

public void sendText(By by, String value) {

this.waitTillPresence(by).sendKeys(value);

}

public String getElementText(By by) {

return this.waitTillPresence(by).getText();

}

public String getTitle() {

String title = getDriver().getTitle();

return title;

}

public void sendValueUsingJS(WebElement ele, String value) {

JavascriptExecutor executor = ((JavascriptExecutor) getDriver());

executor.executeScript("arguments[0].value='" + value + "';", ele);

}

public void clearTextUsingSendKeys(WebElement toClear) {

toClear.sendKeys(Keys.CONTROL + "a");

toClear.sendKeys(Keys.DELETE);

}

public void selectByIndex(WebElement ele, int index) {

Select select = new Select(ele);

select.selectByIndex(index);

}

public void selectByValue(WebElement ele, String value) {

Select select = new Select(ele);

select.selectByValue(value);

}

public void selectByVisibleText(WebElement ele, String visibleText) {

Select select = new Select(ele);

select.selectByVisibleText(visibleText);

}

public void deselectByIndex(WebElement ele, int index) {

Select deselect = new Select(ele);

deselect.deselectByIndex(index);

}

public void deselectByValue(WebElement ele, String value) {

Select deselect = new Select(ele);

deselect.deselectByValue(value);

}

public void deselectByVisibleText(WebElement ele, String visibleText) {

Select deselect = new Select(ele);

deselect.deselectByVisibleText(visibleText);

}

public String getAttribute(WebElement ele, String value) {

String attribute = ele.getAttribute(value);

return attribute;

}

public String getTagname(WebElement ele) {

return ele.getTagName();

}

public String getText(WebElement ele) {

String text = ele.getText();

return text;

}

public void HighlightElement(WebElement ele) {

JavascriptExecutor js = (JavascriptExecutor) getDriver();

js.executeScript("arguments[0].style.border='3px solid red'", ele);

js.executeScript("arguments[0].style.border='1px white'", ele);

}

public void dragAndDropAction(WebElement elementToHover, WebElement elementToClick) {

Actions action = new Actions(getDriver());

action.dragAndDrop(elementToHover, elementToClick).build().perform();

}

public boolean isElementDisplayed(WebElement ele) {

if (ele.isDisplayed()) {

return true;

} else {

return false;

}

}

public boolean isElementEnabled(WebElement ele) {

if (ele.isEnabled()) {

return true;

} else {

return false;

}

}

public boolean isElementSelected(WebElement ele) {

if (ele.isSelected()) {

return true;

} else {

return false;

}

}

public void moveToElementAction(WebElement ele) {

Actions actions = new Actions(getDriver());

actions.moveToElement(ele);

actions.build().perform();

}

public WebElement fluientWaitforElement(WebElement element, int timoutSec, int pollingSec) {

FluentWait<WebDriver> fWait = new FluentWait<WebDriver>(getDriver()).withTimeout(Duration.ofSeconds(30))

.pollingEvery(Duration.ofSeconds(30)).ignoring(NoSuchElementException.class, TimeoutException.class)

.ignoring(StaleElementReferenceException.class);

for (int i = 0; i < 2; i++)

fWait.until(ExpectedConditions.visibilityOf(element));

return element;

}

public void switchwindow(String pageTitle) {

String currentWindow = getDriver().getWindowHandle();

Set<String> handles = getDriver().getWindowHandles();

for (String winHandle : handles) {

String currentWindowTitle = getDriver().switchTo().window(winHandle).getTitle();

if (currentWindowTitle.equals(pageTitle)) {

break;

} else {

getDriver().switchTo().window(currentWindow);

}

}

}

public void selectCalendarDateWithJS(String date, WebElement element) {

JavascriptExecutor js = ((JavascriptExecutor) getDriver());

js.executeScript("arguments[0].setAttribute('value','" + date + "');", element);

}

public void clickElementWithJS(WebElement element) {

JavascriptExecutor js = ((JavascriptExecutor) getDriver());

js.executeScript("arguments[0].click();", element);

}

public void scrollPageDownWithJS() {

JavascriptExecutor js = ((JavascriptExecutor) getDriver());

js.executeScript("window.scrollTo(0,document.body.scrollHeight)");

}

}