Handling pagenation: with handling stale element exception

//store the pagenation in to the list of webelements

List<WebElement> pagination=dr.findElements(By.*xpath*("//\*[@id='ProjectsGrid']/div/ul/li"));

//verify the pagenation s presenet or not

**if**(pagination .size()>0){

System.***out***.println("pagination exists");

//get size of the pagenation

System.***out***.println(pagination.size());

//iterate the pagenation

**for**(**int** i2=2;i2<pagination.size();i2++){

//click pagenation links(1,2,3,4...)

pagination.get(i2).click();

Thread.*sleep*(2000);

//for stale element we need to initalize the pagenation

pagination=dr.findElements(By.*xpath*("//\*[@id='ProjectsGrid']/div/ul/li"));

}

}**else** {

System.***out***.println("pagination not exists");

}

Handling table:

Xpath for hanling table element dr.findElement(By.xpath("//td[contains(text(),22274)]/following-sibling::td[6]")).click();

package com.test.testexamples;

import java.util.Iterator;  
import java.util.List;  
import java.util.ListIterator;  
import java.util.concurrent.TimeUnit;  
import org.openqa.selenium.By;  
import org.openqa.selenium.JavascriptExecutor;  
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.WebElement;  
import org.openqa.selenium.chrome.ChromeDriver;  
public class testtable {  
    public static void main(String[] args) throws InterruptedException {  
    WebDriver dr;  
    String i="22274";  
     dr= new ChromeDriver();  
     dr.get("https://uat-gms.thebigword.com/SingleSignOn/");       
     dr.manage().window().maximize();  
    dr.findElement(By.xpath("//\*[@id='Username']")).sendKeys("Techwave.Client1");  
    dr.findElement(By.xpath("//\*[@id='Password']")).sendKeys("thebigUat11");  
    dr.findElement(By.xpath("//\*[@id='loginButton']/span[2]")).click();  
    Thread.sleep(3000);      
    dr.findElement(By.xpath("//\*[@id='Tile1']/span[2]")).click();  
    Thread.sleep(3000);  
    //JavascriptExecutor js = (JavascriptExecutor) dr;  
    //js.executeScript("window.scrollBy(0,1100)");  
    dr.manage().timeouts().implicitlyWait(50, TimeUnit.SECONDS);  
    //dr.findElement(By.xpath("//td[contains(text(),22274)]/following-sibling::td[6]")).click();

List<WebElement>  e1=dr.findElements(By.xpath("//\*[@id='ProjectsGrid']/table/tbody/tr/td[1]"));  
     
    Iterator<WebElement> i1 = e1.iterator();  
    while(i1.hasNext()) {  
        WebElement row = i1.next();  
        System.out.println(row.getText());  
          
        String s=row.getText();  
        if(s.equals(i)){  
            JavascriptExecutor js1 = (JavascriptExecutor) dr;  
            js1.executeScript("window.scrollBy(0,1200)");  
            dr.manage().timeouts().implicitlyWait(50, TimeUnit.SECONDS);  
            dr.findElement(By.xpath("//td[contains(text(),'20225')]/following-sibling::td[6]")).click();  
        }else{  
            WebElement n=dr.findElement(By.xpath("//\*[@id='ProjectsGrid']/div/a[3]/span"));  
              
              
            JavascriptExecutor js1 = (JavascriptExecutor) dr;  
            js1.executeScript("window.scrollBy(0,1100)");  
            //js1.executeScript("arguements[0].scrollIntoView()",n);  
            n.click();  
            dr.manage().timeouts().implicitlyWait(100, TimeUnit.SECONDS);  
            dr.findElement(By.xpath("//td[contains(text(),'20225')]/following-sibling::td[6]")).click();  
        }  
    }   
}}

[scrollingthepage](#scrollingthepage)

JavascriptExecutor js = (JavascriptExecutor) dr;

js.executeScript("window.scrollBy(0,1000)");

**javascript helper:**

package com.test.helpers;

import org.apache.log4j.Logger;

import org.openqa.selenium.JavascriptExecutor;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

public class Javascripthelper {

public WebDriver dr;

private Logger log=Logger.getLogger(Javascripthelper.class);

public Javascripthelper(WebDriver dr){

this.dr=dr;

log.debug("Javascripthelper"+this.dr.hashCode());

}

public Object excutescript(String script){

JavascriptExecutor exe=(JavascriptExecutor)dr;

log.info(script);

return exe.executeScript(script);

}

public Object excutescript(String script,Object...args){

JavascriptExecutor exe=(JavascriptExecutor)dr;

log.info(script);

return exe.executeScript(script,args);

}

public void scrolltoelement(WebElement element){

excutescript("window.scrollTo(arguement[0],arguement[1])",element.getLocation().x,element.getLocation().y);

log.info(element);

}

public void scrolltoelementandclick(WebElement element){

scrolltoelement(element);

element.click();

log.info(element);

}

public void scrollIntoview(WebElement element){

excutescript("arguements[0].scrollIntoView()",element);

log.info(element);

}

public void scrollIntoviewandclick(WebElement element){

scrollIntoview(element);

element.click();

log.info(element);

}

public void scrollDownvertically(){

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

}

public void scrollupvertically(){

excutescript("window.scrollTo(0,-document.body.scrollHeight)");

}

public void scrolldownByPixcel(){

excutescript("window.scrollBy(0,1500)");

}

public void scrollupByPixcel(){

excutescript("window.scrollBy(0,-1500)");

}

public void zoomByPercentage(){

excutescript("document.body.style.zoom='40%'");

}

public void zoomBy100Percentage(){

excutescript("document.body.style.zoom='100%'");

}

}

**Browser helper:**

package com.test.helpers;

import java.awt.Window;

import java.util.LinkedList;

import java.util.Set;

import org.openqa.selenium.WebDriver;

public class Browserhelper {

public WebDriver dr;

public Browserhelper(WebDriver dr){

this.dr=dr;

}

public void goback(){

dr.navigate().back();

}

public void gofarwad(){

dr.navigate().forward();

}

public void Refresh(){

dr.navigate().refresh();

}

public Set<String> getwinowhandles(){

return dr.getWindowHandles();

}

public void switchtowindow(int index){

LinkedList<String>windowid=new LinkedList<String>(getwinowhandles());

if(index< 0|| index>windowid.size())

throw new IllegalArgumentException("invalid index"+index);

dr.switchTo().window(windowid.get(index));

//log.info(index);

}

public void switchtoparentwindow(){

LinkedList<String>windowid=new LinkedList<String>(getwinowhandles());

dr.switchTo().window(windowid.get(0));

//log.info("");

}

public void switchtoparentwithchildclose(){

LinkedList<String>windowid=new LinkedList<String>(getwinowhandles());

for(int i=1;i<windowid.size();i++){

//log.info(windowid.get(i));

dr.switchTo().window(windowid.get(i));

dr.close();

}switchtoparentwindow();

}

public void switchtoframe(String nameorid){

dr.switchTo().frame(nameorid);

}

}

**date helper:**

package com.test.helpers;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.util.Calendar;

public class Datehelper {

public static String getcurrentDatetime(){

DateFormat datefarmat=new SimpleDateFormat("\_yyyy\_MM\_dd\_HH-mm-ss");

Calendar cal=Calendar.getInstance();

String time=""+datefarmat.format(cal.getTime());

return time;

}

public static String getcurrentate(){

return getcurrentDatetime().substring(0,11);

}

public static void main(String args){

Datehelper d=new Datehelper();

System.out.println(Datehelper.getcurrentate());

}

}

**Dropdownhelper:**

package com.test.helpers;

import java.util.LinkedList;

import java.util.List;

import org.apache.log4j.Logger;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

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

public class Dropdownhelper {

public WebDriver dr;

private Logger log=Logger.getLogger(Dropdownhelper.class);

public Dropdownhelper(WebDriver dr){

this.dr=dr;

log.debug("Dropdownhelper"+this.dr.hashCode());

}

public void selectusingvisiblvalue(WebElement element,String visibletext){

Select se=new Select(element);

se.selectByVisibleText(visibletext);

log.info("locator:"+element+ "value:"+visibletext);

}

public String getselectedvalue(WebElement element){

String value=new Select(element).getFirstSelectedOption().getText();

log.info("Webelement"+element+"value:"+value);

return value;

}

public void selectusingIndex(WebElement element,int index){

Select se=new Select(element);

se.selectByIndex(index);

log.info("locator:"+element+ "value:"+index);

}

public List<String> getAlldropdownvalues(WebElement locator){

Select se=new Select(locator);

List<WebElement>elementlist=se.getAllSelectedOptions();

List<String>valuelist=new LinkedList<String>();

for(WebElement element:elementlist){

log.info(element.getText());

valuelist.add(element.getText());

}

return valuelist;

}

}

**ExtentreportTEstng**

package com.test.helpers;

import java.util.Calendar;

import java.util.Date;

import java.util.List;

import java.util.Map;

import org.testng.IReporter;

import org.testng.IResultMap;

import org.testng.ISuite;

import org.testng.ISuiteResult;

import org.testng.ITestContext;

import org.testng.ITestResult;

import org.testng.xml.XmlSuite;

import com.aventstack.extentreports.ExtentReports;

import com.aventstack.extentreports.ExtentTest;

import com.aventstack.extentreports.Status;

import com.aventstack.extentreports.reporter.ExtentHtmlReporter;

public class ExtentReporterNG implements IReporter {

private ExtentReports extent;

ExtentHtmlReporter htmlReporter;

public void generateReport(List<XmlSuite> xmlSuites, List<ISuite> suites, String outputDirectory) {

htmlReporter = new ExtentHtmlReporter(System.getProperty("user.dir")+"\\Reports\\htmlreport.html");

extent = new ExtentReports();

extent.attachReporter(htmlReporter);

for (ISuite suite : suites) {

Map<String, ISuiteResult> result = suite.getResults();

for (ISuiteResult r : result.values()) {

ITestContext context = r.getTestContext();

buildTestNodes(context.getPassedTests(), Status.PASS);

buildTestNodes(context.getFailedTests(), Status.FAIL);

buildTestNodes(context.getSkippedTests(), Status.SKIP);

}

}

extent.flush();

}

private void buildTestNodes(IResultMap tests, Status status) {

ExtentTest test;

if (tests.size() > 0) {

for (ITestResult result : tests.getAllResults()) {

test = extent.createTest(result.getMethod().getMethodName());

/\*test.getTest(). = getTime(result.getStartMillis());

test.getTest().endedTime = getTime(result.getEndMillis());\*/

for (String group : result.getMethod().getGroups())

test.assignCategory(group);

String message = "Test " + status.toString().toLowerCase() + "ed";

if (result.getThrowable() != null)

message = result.getThrowable().getMessage();

test.log(status, message);

// extent.endTest(test);

}

}

}

private Date getTime(long millis) {

Calendar calendar = Calendar.getInstance();

calendar.setTimeInMillis(millis);

return calendar.getTime();

}

}

**Loghelper:**

package com.test.helpers;

import org.apache.log4j.Logger;

import org.apache.log4j.PropertyConfigurator;

public class Loghelper {

public static boolean root=false;

public static Logger getlogger(Class clas){

if(root){

return Logger.getLogger(clas);

}

// PropertyConfigurator.configure(Resourcehelper.getResourcePath("/src/resource/log4j.properties"));

root=true;

return Logger.getLogger(clas);

}

}

**Resourcehelper:**

package com.test.helpers;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.InputStream;

public class Resourcehelper {

public static String getResourcepath(String resource){

String path=getBaseResourcepath()+resource;

return path;

}

public static String getBaseResourcepath(){

String path=System.getProperty("user.dir");

System.out.println(path);

return path;

}

public static InputStream getResourcepathinputstream(String path) throws FileNotFoundException {

return new FileInputStream(Resourcehelper.getResourcepath(path));

}}

**Verification helper:**

**package** com.test.helpers;

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

**public** **class** verificationhelper {

**public** **static** **synchronized** **boolean** verifyElementPresent(WebElement element){

//log.info("element not displayed"+e)

**boolean** isDisplayed=**false**;

**try**{

isDisplayed=element.isDisplayed();

element.getText();

}**catch**(Exception e){

//log.error("element not found"+e)

}**return** isDisplayed;

}

**public** **static** **synchronized** **boolean** verifyElementNOtPresent(WebElement element){

**boolean** isDisplayed=**false**;

**try**{

element.isDisplayed();

//log.info("element.getText()+"is displaye");

element.getText();

}

**catch**(Exception e){

//log.error("element not found"+e)

isDisplayed=**true**;

}**return** isDisplayed;

}

**public** **static** **synchronized** **boolean** verifytextEqual(WebElement element,String expectedtext){

**boolean** flag=**false**;

**try**{

String actvaltext=element.getText();

**if**(actvaltext.equals(expectedtext)){

//log.info(actvaltext is"+actvalte+"expected text is "+expectedtext);

**return** flag=**true**;

}**else**{

//log.error(actvaltext is"+actvalte+"expected text is "+expectedtext);

}}

**catch**(Exception e){

//log.error(actvaltext is"+actvalte+"expected text is "+expectedtext);

//log.info(""text is not matching"+e);

}**return** flag;

}

}

**Wait helper:**

package com.test.helpers;

import java.util.NoSuchElementException;

import java.util.concurrent.TimeUnit;

import org.apache.log4j.Logger;

import org.openqa.selenium.ElementNotVisibleException;

import org.openqa.selenium.NoSuchFrameException;

import org.openqa.selenium.StaleElementReferenceException;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

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

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

public class Waithelper {

public WebDriver dr;

private Logger log=Logger.getLogger(Waithelper.class);

public Waithelper(WebDriver dr){

this.dr=dr;

log.debug("Waithelper"+this.dr.hashCode());

}

public void setimplicitwait(long timeout,TimeUnit unit){

log.info(timeout);

dr.manage().timeouts().implicitlyWait(timeout, unit==null ? TimeUnit.SECONDS:unit);

}

public WebDriverWait getwait(long timeoutinmilliseconds,int pollingEveryInmilliseconds){

log.debug("");

WebDriverWait wait=new WebDriverWait(dr,timeoutinmilliseconds);

wait.pollingEvery(pollingEveryInmilliseconds,TimeUnit.MILLISECONDS);

wait.ignoring(NoSuchElementException.class);

wait.ignoring(ElementNotVisibleException.class);

wait.ignoring(StaleElementReferenceException.class);

wait.ignoring(NoSuchFrameException.class);

return wait;

}

public void waitforElementvisible(WebElement locator,long timeoutinmilliseconds,int pollingeveryinmilliseconds){

log.info(locator);

WebDriverWait wait=getwait(timeoutinmilliseconds, pollingeveryinmilliseconds);

wait.until(ExpectedConditions.visibilityOf(locator));

}

public void waitforElement(WebDriver dr,WebElement element,long timeout){

WebDriverWait wait=new WebDriverWait(dr, timeout);

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

log.info("element found"+element.getText());

}

public WebElement waitforElement(WebDriver dr,long timeout,WebElement element){

WebDriverWait wait=new WebDriverWait(dr, timeout);

return wait.until(ExpectedConditions.elementToBeClickable(element));

}

}