BaseTest.java:-

package net.premieredigital.portal;

import java.io.File;

import java.io.IOException;

import java.lang.annotation.Annotation;

import java.lang.reflect.Method;

import java.net.URL;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.concurrent.TimeUnit;

import net.premieredigital.portal.pom.LoginPage;

import net.premieredigital.portal.pom.PageConfiguration;

import org.apache.commons.io.FileUtils;

import org.apache.commons.lang.StringUtils;

import org.apache.log4j.Logger;

import org.apache.log4j.xml.DOMConfigurator;

import org.openqa.selenium.By;

import org.openqa.selenium.Capabilities;

import org.openqa.selenium.JavascriptExecutor;

import org.openqa.selenium.NoSuchElementException;

import org.openqa.selenium.NoSuchWindowException;

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.firefox.FirefoxDriver;

import org.openqa.selenium.htmlunit.HtmlUnitDriver;

import org.openqa.selenium.ie.InternetExplorerDriver;

import org.openqa.selenium.remote.CapabilityType;

import org.openqa.selenium.remote.DesiredCapabilities;

import org.openqa.selenium.remote.LocalFileDetector;

import org.openqa.selenium.remote.RemoteWebDriver;

import org.openqa.selenium.remote.UnreachableBrowserException;

import org.openqa.selenium.safari.SafariOptions;

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

import org.testng.SkipException;

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.Parameters;

import org.testng.annotations.Test;

import com.google.common.base.Predicate;

import com.saucelabs.common.SauceOnDemandAuthentication;

import com.saucelabs.common.SauceOnDemandSessionIdProvider;

import com.saucelabs.testng.SauceOnDemandAuthenticationProvider;

public class BaseTest implements SauceOnDemandSessionIdProvider, SauceOnDemandAuthenticationProvider {

public SauceOnDemandAuthentication authentication;

private String applicationUrl;

protected WebDriver driver;

protected static PageConfiguration pageConfig;

private static Logger log = Logger.getLogger(BaseTest.class.getName());

//private SessionId sessionId = null;

private String methodName = null;

protected static ThreadLocal<WebDriver> webDriver = new ThreadLocal<WebDriver>();

protected ThreadLocal<String> sessionId = new ThreadLocal<String>();

private DesiredCapabilities capabilities;

@BeforeClass

@Parameters({"userName", "key", "os", "browser", "browserVersion", "suiteName", "runPlatform", "applicationUrl"})

public void beforeSuite(String userName,

String key,

String os,

String browser,

String browserVersion,

String suiteName,

String runPlatform,

String applicationUrl

) throws Exception {

// one-time initialization code

capabilities = DesiredCapabilities.chrome();

// capabilities = DesiredCapabilities.phantomjs();

// capabilities.setJavascriptEnabled(true);

capabilities.setCapability(CapabilityType.ACCEPT\_SSL\_CERTS, true);

capabilities.setCapability("idle-timeout", 300);

capabilities.setCapability("command-timeout", 600);

capabilities.setCapability("sauce-advisor", true);

capabilities.setCapability("elementScrollBehavior", 1);

this.pageConfig = new PageConfiguration(applicationUrl);

this.pageConfig.setBrowserName(browser);

//webDriver.set(new ChromeDriver());

// webDriver.set(new PhantomJSDriver(capabilities));

capabilities.setCapability(CapabilityType.UNEXPECTED\_ALERT\_BEHAVIOUR,org.openqa.selenium.UnexpectedAlertBehaviour.ACCEPT);

webDriver.set(new RemoteWebDriver(new URL("http://52.8.100.59:4444/wd/hub"), capabilities));

webDriver.get().manage().window().maximize();

((RemoteWebDriver) webDriver.get()).setFileDetector(new LocalFileDetector());

webDriver.get().manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

try {

new LoginPage(webDriver.get(), this.pageConfig).login();

}catch(NoSuchWindowException e) {

e.printStackTrace();

webDriver.get().quit();

}catch (Exception ex) {

ex.printStackTrace();

webDriver.get().quit();

}

}

@SuppressWarnings("unchecked")

@BeforeMethod(alwaysRun=true)

public void setUp(Method method) throws Exception {

DOMConfigurator.configure("log4j.xml");

//log.info("Hello World!");

System.out.println("START TEST : "+getTestDescription(method));

webDriver.get().get(pageConfig.getRootURL());

try {

this.preCondition();

}

catch(NoSuchWindowException e) {

e.printStackTrace();

webDriver.get().quit();

}

catch (UnreachableBrowserException ex) {

ex.printStackTrace();

webDriver.get().quit();

}

}

@AfterMethod

public void afterTest(Method method) {

DateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy\_HH-mm");

Date date = new Date();

createScreenshotFolder(System.getProperty("user.dir"));

createFailedScreenshot(System.getProperty("user.dir")+"/screenshot");

try {

File scrFile = ((TakesScreenshot)webDriver.get()).getScreenshotAs(OutputType.FILE);

FileUtils.copyFile(scrFile, new File(System.getProperty("user.dir")+"/screenshot/"+getTestDescription(method).replaceAll(" ", "\_")+"\_"+dateFormat.format(date)+".png"));

} catch (IOException e1) {

e1.printStackTrace();

} catch (NoSuchWindowException ex) {

ex.printStackTrace();

}

System.out.println("END TEST : "+getTestDescription(method));

}

public void preCondition() throws Exception {

}

public void postCondition(){

}

public void createScreenshotFolder(String path){

File dir = new File("screenshot");

dir.mkdir();

}

public void createFailedScreenshot(String path){

File dir = new File("failed");

dir.mkdir();

}

public void takeScreenshot(String name){

/\* DateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy\_HH-mm");

Date date = new Date();

try {

File scrFile = ((TakesScreenshot)webDriver.get()).getScreenshotAs(OutputType.FILE);

FileUtils.copyFile(scrFile, new File(System.getProperty("user.dir")+"/screenshot/"+name+"\_"+dateFormat.format(date)+".png"));

} catch (IOException e1) {

e1.printStackTrace();

} catch (NoSuchWindowException ex) {

ex.printStackTrace();

}\*/

}

public void adminUserLogin(){

new LoginPage(webDriver.get(), this.pageConfig).login();

}

public void adminUserLogout(){

new LoginPage(webDriver.get(), this.pageConfig).logout();

}

public void fluentlyWait(FluentWait<By> fluentWait) {

fluentWait.pollingEvery(100, TimeUnit.MILLISECONDS);

fluentWait.withTimeout(150000, TimeUnit.MILLISECONDS);

fluentWait.until(new Predicate<By>() {

public boolean apply(By by) {

try {

return webDriver.get().findElement(by)!=null;

} catch (NoSuchElementException ex) {

return false;

}

}

});

}

public void setImplicitWait(int seconds) {

}

@Override

public SauceOnDemandAuthentication getAuthentication() {

return authentication;

}

protected String getTestDescription(Method m) {

Annotation[] annotations = m.getDeclaredAnnotations();

for (Annotation annotation : annotations) {

if(annotation instanceof Test){

Test test = (Test) annotation;

if (!test.description().isEmpty()) {

return test.description();

}

}

}

return m.getName();

}

@Override

public String getSessionId() {

// TODO Auto-generated method stub

return sessionId.get();

}

@AfterClass

public void afterSuite() {

try {

new LoginPage(webDriver.get(), this.pageConfig).logout();

webDriver.get().quit();

}

catch(Exception ex) {

log.error(this.getClass().getName());

log.error(ex.getMessage());

ex.printStackTrace();

}

}

public WebDriver getWebDriver() {

// TODO Auto-generated method stub

return webDriver.get();

}

public PageConfiguration getPageConfiguration() {

// TODO Auto-generated method stub

return this.pageConfig;

}

}

Pom.xml:-

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

<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>net.premieredigital.portal</groupId>

<artifactId>pds-portal-test</artifactId>

<version>1.0</version>

<dependencies>

<dependency>

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

<artifactId>selenium-java</artifactId>

<version>2.43.1</version>

</dependency>

<dependency>

<groupId>com.opera</groupId>

<artifactId>operadriver</artifactId>

</dependency>

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>6.8.8</version>

</dependency>

<dependency>

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

<artifactId>guava</artifactId>

<version>18.0</version>

</dependency>

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi</artifactId>

<version>3.11-beta2</version>

</dependency>

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

<version>3.11-beta2</version>

<exclusions>

<exclusion>

<artifactId>xml-apis</artifactId>

<groupId>xml-apis</groupId>

</exclusion>

</exclusions>

</dependency>

<dependency>

<groupId>net.sourceforge.jexcelapi</groupId>

<artifactId>jxl</artifactId>

<version>2.6.12</version>

</dependency>

<dependency>

<groupId>com.saucelabs</groupId>

<artifactId>sauce\_testng</artifactId>

<version>2.1.10</version>

</dependency>

<dependency>

<groupId>xml-apis</groupId>

<artifactId>xml-apis</artifactId>

<version>1.4.01</version>

</dependency>

<dependency>

<groupId>com.github.detro</groupId>

<artifactId>phantomjsdriver</artifactId>

<version>1.2.0</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<version>2.17</version>

<configuration>

<suiteXmlFiles>

<suiteXmlFile>${testNg}</suiteXmlFile>

</suiteXmlFiles>

</configuration>

</plugin>

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>build-helper-maven-plugin</artifactId>

<version>1.8</version>

<executions>

<execution>

<id>add-gen-source</id>

<phase>generate-sources</phase>

<goals>

<goal>add-source</goal>

</goals>

<configuration>

<sources>

<source>src</source>

</sources>

</configuration>

</execution>

<execution>

<id>add-extra-source</id>

<phase>generate-sources</phase>

<goals>

<goal>add-source</goal>

</goals>

<configuration>

<sources>

<source>src</source>

</sources>

</configuration>

</execution>

</executions>

</plugin>

<plugin>

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

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

<version>3.0</version>

<configuration>

<source>1.7</source>

<target>1.7</target>

<debug>false</debug>

<executable>${JAVA\_HOME}/bin/javac</executable>

</configuration>

</plugin>

</plugins>

</build>

<dependencyManagement>

<dependencies>

<dependency>

<groupId>com.opera</groupId>

<artifactId>operadriver</artifactId>

<version>1.5</version>

<exclusions>

<exclusion>

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

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

</exclusion>

</exclusions>

</dependency>

</dependencies>

</dependencyManagement>

<repositories>

<repository>

<id>saucelabs-repository</id>

<url>https://repository-saucelabs.forge.cloudbees.com/release</url>

<releases>

<enabled>true</enabled>

</releases>

<snapshots>

<enabled>true</enabled>

</snapshots>

</repository>

</repositories>

</project>

Root\_testng.xml:-

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

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

<suite name="PDS Suites">

<suite-files>

<suite-file path="admin.xml" />

<suite-file path="dashboard.xml" />

</suite-files>

</suite>

Admin.xml:-

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

<suite name="Master branch PDS Tests" parallel="classes" thread-count="8">

<parameter name="userName" value="test999"/>

<parameter name="key" value="ff3d72b4-1298-48cf-96af-9373b8564e5e"/>

<parameter name="suiteName" value="PDS Idle time max test"/>

<parameter name="applicationUrl" value="https://masterbranch.premieredigital.net" />

<parameter name="runPlatform" value="local"/>

<parameter name="os" value="OS X 10.10"/>

<parameter name="browser" value="chrome"/>

<parameter name="browserVersion" value="41"/>

<test name="ChromeTests" preserve-order="true">

<packages>

<package name="net.premieredigital.portal.admin.\*"/>

<package name="net.premieredigital.portal.ordersview.\*"/>

<package name="net.premieredigital.portal.orderitemsview.\*"/>

<package name="net.premieredigital.portal.metadataview.\*"/>

<package name="net.premieredigital.portal.tasksview.\*"/>

<package name="net.premieredigital.portal.masteringview.\*"/>

</packages>

</test>

</suite>

BaseTestConfig.java:-

package net.premieredigital.portal.pom;

import java.io.FileInputStream;

import java.lang.reflect.Field;

import org.apache.poi.xssf.usermodel.XSSFSheet;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public abstract class BaseTestConfig {

public XSSFSheet getSheet(String sheetname) throws Exception

{

FileInputStream fis=new FileInputStream(System.getProperty("user.dir")+"//resources//TestData.xlsx");

XSSFWorkbook wb = new XSSFWorkbook(fis);

XSSFSheet sheet= wb.getSheet(sheetname);

return sheet;

}

public void setField(Object entity, String fieldName, String value)

throws NoSuchFieldException, IllegalAccessException {

Field field = entity.getClass().getDeclaredField(fieldName);

field.set(entity, value);

}

}

PageConfig.java:-

package net.premieredigital.portal.pom;

import java.lang.reflect.Field;

import java.util.Random;

import net.premieredigital.portal.BaseTest;

import org.apache.poi.xssf.usermodel.XSSFRow;

import org.apache.poi.xssf.usermodel.XSSFSheet;

public class PageConfiguration extends BaseTestConfig{

private String username;

private String password;

private String rootURL;

private String providername;

private String username1;

private String browserName;

private String client\_user\_name;

private Random random = new Random();

private XSSFSheet sheet;

public PageConfiguration(String applicationUrl) throws Exception {

this.sheet = getSheet("PageConfiguration");

XSSFRow header\_row = sheet.getRow(0);

XSSFRow value\_row = sheet.getRow(1);

for (int i = 0; header\_row.getCell(i) != null; i++) {

Field field = getClass().getDeclaredField(header\_row.getCell(i).getStringCellValue());

field.set(this, value\_row.getCell(i).getStringCellValue());

}

this.setProviderName();

this.setRootURL(applicationUrl);

}

public void setRootURL(String applicationUrl){

rootURL = applicationUrl;

}

public String getBrowserName() {

return browserName;

}

public void setBrowserName(String browserName) {

this.browserName = browserName;

}

public String getUsername() {

return username;

}

public String getPassword() {

return password;

}

public String getRootURL() {

return rootURL;

}

public String getProviderName() {

return providername;

}

public void setPassword(String password) {

this.password = password;

}

public String getUsername1() {

return username1;

}

public void setProviderName() {

this.providername = providername + Integer.toString(random.nextInt(100000) + 1);

}

public String getClientUserName() {

return client\_user\_name;

}

public void setClientUserName(String clientUserName) {

this.client\_user\_name = clientUserName;

}

}