Pom.xml file for tvs:

<?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>

<artifactId>sauce\_testng</artifactId>

<groupId>net.premieredigital</groupId>

<version>1.0</version>

<packaging>jar</packaging>

<name>PDS\_TEST</name>

<description>PDS Test scripts</description>

<dependencies>

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>6.1.1</version>

</dependency>

<dependency>

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

<artifactId>selenium-java</artifactId>

<version>LATEST</version>

</dependency>

<dependency>

<groupId>commons-lang</groupId>

<artifactId>commons-lang</artifactId>

<version>2.6</version>

</dependency>

<dependency>

<groupId>com.saucelabs</groupId>

<artifactId>sauce\_testng</artifactId>

<version>[1.0.0,)</version>

</dependency>

<dependency>

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

<artifactId>poi</artifactId>

<version>3.10.1</version>

</dependency>

<dependency>

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

<artifactId>poi-ooxml</artifactId>

<version>3.10.1</version>

<exclusions>

<exclusion>

<artifactId>xml-apis</artifactId>

<groupId>xml-apis</groupId>

</exclusion>

</exclusions>

</dependency>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.17</version>

</dependency>

<dependency>

<groupId>xml-apis</groupId>

<artifactId>xml-apis</artifactId>

<version>1.4.01</version>

</dependency>

<dependency>

<groupId>com.googlecode.json-simple</groupId>

<artifactId> json-simple</artifactId>

<version>1.1</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

<version>3.0</version>

<configuration>

<source>1.6</source>

<target>1.6</target>

</configuration>

</plugin>

<plugin>

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

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

<version>2.18</version>

<configuration>

<redirectTestOutputToFile>true</redirectTestOutputToFile>

<suiteXmlFiles>

<suiteXmlFile>src/test/resources/xml/testng.xml</suiteXmlFile>

</suiteXmlFiles>

<systemProperties>

<property>

<name>java.util.logging.config.file</name>

<value>src/main/resources/log4j.properties</value>

</property>

</systemProperties>

</configuration>

</plugin>

<plugin>

<groupId>ch.fortysix</groupId>

<artifactId>maven-postman-plugin</artifactId>

<version>0.1.6</version>

<executions>

<execution>

<id>send\_an\_email</id>

<phase>test</phase>

<goals>

<goal>send-mail</goal>

</goals>

<inherited>false</inherited>

<configuration>

<mailhost>192.168.0.6</mailhost>

<mailport>25</mailport>

<mailssl>false</mailssl>

<mailAltConfig>true</mailAltConfig>

<mailuser>hardik.gohil@server1.com</mailuser>

<mailpassword>tatva123</mailpassword>

<from>hardik.gohil@server1.com</from>

<receivers>

<receiver>vikas.thakkar@server1.com</receiver>

<receiver>rakesh.ramani@server1.com</receiver>

<receiver>tarkik.shah@server1.com</receiver>

<receiver>hardik.gohil@server1.com</receiver>

</receivers>

<subject>Test Automation Report </subject>

<failonerror>true</failonerror>

<htmlMessage>Please Find Attached Automation Test Report</htmlMessage>

<fileSets>

<fileSet>

<directory>${basedir}/target/surefire-reports</directory>

<includes>

<include>emailable-report.html</include>

</includes>

</fileSet>

</fileSets>

</configuration>

</execution>

</executions>

</plugin>

</plugins>

</build>

<repositories>

<repository>

<id>saucelabs-repository</id>

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

<releases>

<enabled>true</enabled>

</releases>

<snapshots>

<enabled>true</enabled>

</snapshots>

</repository>

</repositories>

</project>

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="admintestng.xml" />

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

</suite-files>

</suite>

Wiptestng.xml:

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

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

<suite name="WIP Test suites" verbose="2" parallel="tests"

thread-count="2">

<parameter name="username" value="smartuser1" />

<parameter name="key" value="2abf108b-33f2-4df7-a858-23ab4ea70f38" />

<parameter name="suitename" value="WIP Test suites"></parameter>

<parameter name="applicationURL" value="http://183.182.91.146/PDS/"></parameter>

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

<test name="WIP with firefox">

<parameter name="os" value="WIN8" />

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

<parameter name="browserVersion" value="34" />

<packages>

<package name="net.premieredigital.test.wip.register"></package>

<package name="net.premieredigital.test.wip.support"></package>

</packages>

</test>

</suite>

BaseTest.java:

package net.premieredigital;

import java.lang.annotation.Annotation;

import java.lang.reflect.Method;

import java.net.URL;

import org.apache.commons.lang.StringUtils;

import org.apache.log4j.Logger;

import org.apache.log4j.xml.DOMConfigurator;

import org.openqa.selenium.Capabilities;

import org.openqa.selenium.JavascriptExecutor;

import org.openqa.selenium.Platform;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.ie.InternetExplorerDriver;

import org.openqa.selenium.remote.DesiredCapabilities;

import org.openqa.selenium.remote.RemoteWebDriver;

import org.testng.annotations.AfterMethod;

import org.testng.annotations.AfterSuite;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.Parameters;

import org.testng.annotations.Test;

import com.saucelabs.common.SauceOnDemandAuthentication;

import com.saucelabs.common.SauceOnDemandSessionIdProvider;

import com.saucelabs.testng.SauceOnDemandAuthenticationProvider;

public class BaseTest implements SauceOnDemandSessionIdProvider, SauceOnDemandAuthenticationProvider {

public SauceOnDemandAuthentication authentication;

protected ThreadLocal<WebDriver> driver = new ThreadLocal<WebDriver>();

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

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

@SuppressWarnings("unused")

private String applicationURL ;

@Parameters({"username", "key", "os", "browser", "browserVersion", "suitename", "applicationURL", "runPlatform"})

@BeforeMethod(alwaysRun=true)

public void setUp

(String userName,

String key,

String os,

String browser,

String browserVersion,

String suiteName,

String applicationURL,

String runPlatform,

Method method) throws Exception {

DOMConfigurator.configure("src/main/resources/log4j.properties");

Log.info("current method name :"+method.getName());

DesiredCapabilities capabilities;

if(StringUtils.equals(runPlatform, "saucelabs"))

{

if (StringUtils.isNotEmpty(userName) && StringUtils.isNotEmpty(key)) {

authentication = new SauceOnDemandAuthentication(userName, key);

} else {

authentication = new SauceOnDemandAuthentication();

}

capabilities = new DesiredCapabilities();

capabilities.setBrowserName(browser);

capabilities.setCapability("version", browserVersion);

capabilities.setCapability("platform", Platform.valueOf(os));

capabilities.setCapability("name", browser+ ": "+ getTestDescription(method));

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

Log.info("current method name :"+method.getName());

driver.set(new RemoteWebDriver(

new URL("http://" + authentication.getUsername() + ":" + authentication.getAccessKey() + "@ondemand.saucelabs.com:80/wd/hub"), capabilities));

}

else

{

if(StringUtils.equals(browser, "firefox"))

{

driver.set(GetBrowser.getFirefoxDriver());

}

else if(StringUtils.equals(browser, "chrome"))

{

capabilities = DesiredCapabilities.chrome();

driver.set(new ChromeDriver(capabilities));

}

else

{

capabilities = DesiredCapabilities.internetExplorer();

driver.set(new InternetExplorerDriver(capabilities));

}

}

Capabilities cap = ((RemoteWebDriver) driver.get()).getCapabilities();

if(cap.getBrowserName().toLowerCase().equals("chrome") || cap.getBrowserName().toLowerCase().equals("firefox")) {

((JavascriptExecutor) driver.get()).executeScript("window.alert = function(msg) { return true; }");

((JavascriptExecutor) driver.get()).executeScript("window.confirm = function(msg) { return true; }");

((JavascriptExecutor) driver.get()).executeScript("window.prompt = function(msg) { return true; }");

}

sessionID.set(((RemoteWebDriver)driver.get()).getSessionId().toString());

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

preConditions();

}

@Override

public String getSessionId() {

return sessionID.get();

}

@AfterMethod(alwaysRun = true)

public void tearDown() throws Exception {

postConditions();

}

@AfterSuite

public void close() {

Log.info("Completion of the suite");

GetBrowser.getFirefoxDriver().quit();

}

@Override

public SauceOnDemandAuthentication getAuthentication() {

return authentication;

}

public void preConditions () {

}

public void postConditions() {

}

private 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();

}

}

TestDataProvider.java:

package net.premieredigital.utils;

import java.io.FileInputStream;

import java.io.IOException;

import java.lang.reflect.Field;

import org.apache.poi.hssf.usermodel.HSSFSheet;

import org.apache.poi.hssf.usermodel.HSSFWorkbook;

import org.apache.poi.ss.usermodel.Cell;

import org.apache.poi.ss.usermodel.Row;

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

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

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

public class TestDataProvider

{

public static XSSFSheet getSheet(String workbook, String sheetname) throws Exception

{

String url = TestDataProvider.class.getResource("/xls/testdata.xlsx").getPath();

FileInputStream fis = new FileInputStream(url);

XSSFWorkbook wb = new XSSFWorkbook(fis);

XSSFSheet sheet= wb.getSheet(sheetname);

return sheet;

}

public static Object syncPageData(Object obj, String workbook, String sheetname)

{

try {

XSSFSheet sheet = getSheet(workbook, sheetname);

XSSFRow headerRow = sheet.getRow(0);

XSSFRow dataRow = sheet.getRow(1);

for (int i = 0; headerRow.getCell(i) != null; i++)

{

Field field = obj.getClass().getDeclaredField(headerRow.getCell(i).getStringCellValue());

field.setAccessible(true);

Cell cell = dataRow.getCell(i);

switch(cell.getCellType()) {

case Cell.CELL\_TYPE\_BOOLEAN:

field.set(obj, dataRow.getCell(i).getBooleanCellValue());

break;

case Cell.CELL\_TYPE\_NUMERIC:

field.set(obj, dataRow.getCell(i).getNumericCellValue());

break;

case Cell.CELL\_TYPE\_STRING:

field.set(obj, dataRow.getCell(i).getStringCellValue());

break;

case Cell.CELL\_TYPE\_BLANK:

field.set(obj, "") ;

break;

}

}

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return obj;

}

/\*@DataProvider(name="loginparameters")

public static Object[][] readLoginParameters() throws IOException

{

//String url = TestDataProvider.class.getResource("/xls/testdata.xlsx").getPath();

String url = TestDataProvider.class.getResource("/xls/pds\_login.xlsx").getPath();

return ExcelUtils.getExcelData(url);

}

\*/

public static String[][] readTestData() throws IOException

{

String[][] returnData = null;

// InputStream stream = TestDataProvider.class.getClassLoader().getResourceAsStream("xls/testdata.xlsx");

String url = TestDataProvider.class.getResource("/xls/testdata.xls").getPath();

FileInputStream fis = new FileInputStream(url);

HSSFWorkbook workbook = new HSSFWorkbook(fis);

HSSFSheet sheet = workbook.getSheetAt(0);

int totalRows = sheet.getLastRowNum();

int totalCols = 3;

returnData = new String[totalRows][totalCols];

// Iterator<Row> itr = sheet.iterator();

int c = 0;

int r = 0;

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

{

if(i > 0)

{

Row row = sheet.getRow(i);

for(int j = 0; j < totalCols; j++)

{

Cell cell = row.getCell(j);

try

{

returnData[r][c] = cell.getStringCellValue();

}

catch(NullPointerException e)

{

returnData[r][c] = "";

}

c++;

}

c = 0;

r++;

}

}

fis.close();

return returnData;

}

public static void main(String[] args) {

try {

readTestData();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

SundanceConst.java:

package net.premieredigital.utils;

public interface SundanceConst {

public static final String LOGIN\_SHEET = "successlogin" ;

public static final String WIP\_LOGIN\_SHEET = "wip-login" ;

public static final String PDS\_APP\_NAME = "TESTPDS" ;

public static final String WIP\_APP\_NAME = "PDSWIP" ;

}

ExcelUtils.java:

package net.premieredigital.utils;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import org.apache.commons.lang.StringUtils;

import org.apache.poi.ss.usermodel.Cell;

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

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

public class ExcelUtils {

static FileInputStream file = null;

//Get the workbook instance for XLS file

static XSSFWorkbook workbook = null;

//Get first sheet from the workbook

static XSSFSheet sheet = null;

static Object[][] dataArray = null;

public static Object[][] getExcelData(String excelName, String sheetName) {

try {

file = new FileInputStream(new File(excelName));

workbook = new XSSFWorkbook (file);

if(StringUtils.isNotEmpty(sheetName))

{

sheet = workbook.getSheet(sheetName);

}

else

{

sheet = workbook.getSheetAt(0);

}

int totalRows = sheet.getPhysicalNumberOfRows()-1;

int totalCols = sheet.getRow(0).getPhysicalNumberOfCells();

dataArray = new Object[totalRows][totalCols];

for( int i =1 ; i <= totalRows ; i++ ) {

for( int j=0 ; j < totalCols ; j++ ) {

try {

dataArray[i-1][j]=getCellData(i , j);

} catch (Exception e) {

e.printStackTrace();

}

}

}

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

return dataArray;

}

private static Object getCellData(int RowNum, int ColNum) throws Exception{

try{

Cell cell= sheet.getRow(RowNum).getCell(ColNum);

Object cellData = null ;

switch(cell.getCellType()) {

case Cell.CELL\_TYPE\_BOOLEAN:

cellData = cell.getBooleanCellValue();

break;

case Cell.CELL\_TYPE\_NUMERIC:

cellData = cell.getNumericCellValue();

break;

case Cell.CELL\_TYPE\_STRING:

cellData = cell.getStringCellValue();

break;

case Cell.CELL\_TYPE\_BLANK:

cellData = "" ;

break;

}

return cellData;

}catch (Exception e){

return "";

}

}

}