Base:

package base;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.net.HttpURLConnection;

import java.net.URL;

import java.util.List;

import java.util.Set;

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

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

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

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

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import utilties.Xls\_Reader;

public class Base {

public String path;

public FileInputStream fis = null;

public FileOutputStream fileOut = null;

protected XSSFWorkbook workbook = null;

protected XSSFSheet sheet = null;

protected XSSFRow row = null;

protected XSSFCell cell = null;

public static WebDriver webDriver;

public Set<String> uniqueList = null;

public String url;

public HttpURLConnection httpConn;

public URL link;

public static Xls\_Reader excel;

public void intializeBrowser(String url) {

System.setProperty("webdriver.chrome.driver", System.getProperty("user.dir") + "\\src\\driver\\chromedriver.exe");

webDriver = new ChromeDriver();

webDriver.get(url);

try {

Thread.sleep(5000);

} catch (InterruptedException e) {

e.getMessage();

}

}

public List<WebElement> getObjects() {

return webDriver.findElements(By.tagName("a"));

}

}

Util:

Utilities:

package utilties;

import java.io.IOException;

import java.net.HttpURLConnection;

import java.net.URL;

import java.util.Iterator;

import java.util.List;

import java.util.Set;

import org.openqa.selenium.WebElement;

import base.Base;

public class Utilities extends Base {

public Set<String> getUniqueAndValidUrl(List<WebElement> objectList) {

for (int i = 0; i < objectList.size(); i++) {

try {

String url = objectList.get(i).getAttribute("href");

if (url != null && !url.contains("javascript") && !url.contains("file")) {

uniqueList.add(url);

}

} catch (Exception e) {

getObjects();

}

}

return uniqueList;

}

public void verifyLinkAndUpdateTracker(Set<String> uniqueList) throws IOException {

Iterator<String> itr = uniqueList.iterator();

excel = new Xls\_Reader(System.getProperty("user.dir")+"\\Report\\BrokenLink.xlsx");

int i=1;

while (itr.hasNext()) {

url = itr.next();

try {

URL link = new URL(url);

HttpURLConnection httpConn = (HttpURLConnection) link.openConnection();

httpConn.setConnectTimeout(5000);

httpConn.connect();

excel.setCellData("Report", 0, i, url);

excel.setCellData("Report", 1, i, httpConn.getResponseCode());

excel.setCellData("Report", 2, i, httpConn.getResponseMessage());

} catch (Exception e) {

httpConn.setConnectTimeout(5000);

httpConn.connect();

excel.setCellData("Report", 0, i, url);

excel.setCellData("Report", 1, i, e.getMessage());

excel.setCellData("Report", 2, i, "Configured TimeOut is 5 Seconds");

}

i++;

}

}

}

Xls\_Reader:

package utilties;

import java.io.FileInputStream;

import java.io.FileOutputStream;

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

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

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

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

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

import org.apache.poi.ss.util.CellRangeAddress;

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

import base.Base;

public class Xls\_Reader extends Base {

public Xls\_Reader(String path) {

this.path = path;

try {

fis = new FileInputStream(path);

workbook = new XSSFWorkbook(fis);

// sheet = workbook.getSheetAt(0);

fis.close();

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public int getRowCount(String sheetName) {

int index = workbook.getSheetIndex(sheetName);

if (index == -1) {

return 0;

} else {

sheet = workbook.getSheetAt(index);

int number = sheet.getLastRowNum() + 1;

return number;

}

}

public boolean setCellData(String sheetName, int colNum, int rowNum, String data) {

try {

fis = new FileInputStream(path);

workbook = new XSSFWorkbook(fis);

if (rowNum <= 0)

return false;

int index = workbook.getSheetIndex(sheetName);

if (index == -1)

return false;

sheet = workbook.getSheetAt(index);

row = sheet.getRow(0);

/\*

\* for(int i=0;i<row.getLastCellNum();i++){

\* //System.out.println(row.getCell(i).getStringCellValue().trim());

\* if(row.getCell(i).getStringCellValue().trim().equals(colName)) colNum=i; }

\*/

if (colNum == -1)

return false;

sheet.autoSizeColumn(colNum);

row = sheet.getRow(rowNum - 1);

if (row == null)

row = sheet.createRow(rowNum - 1);

cell = row.getCell(colNum);

if (cell == null)

cell = row.createCell(colNum);

cell.setCellValue(data);

fileOut = new FileOutputStream(path);

workbook.write(fileOut);

fileOut.close();

} catch (Exception e) {

e.printStackTrace();

return false;

}

return true;

}

public boolean setCellData(String sheetName, int colNum, int rowNum, int data) {

try {

fis = new FileInputStream(path);

workbook = new XSSFWorkbook(fis);

if (rowNum <= 0)

return false;

int index = workbook.getSheetIndex(sheetName);

if (index == -1)

return false;

sheet = workbook.getSheetAt(index);

row = sheet.getRow(0);

/\*

\* for(int i=0;i<row.getLastCellNum();i++){

\* //System.out.println(row.getCell(i).getStringCellValue().trim());

\* if(row.getCell(i).getStringCellValue().trim().equals(colName)) colNum=i; }

\*/

if (colNum == -1)

return false;

sheet.autoSizeColumn(colNum);

row = sheet.getRow(rowNum - 1);

if (row == null)

row = sheet.createRow(rowNum - 1);

cell = row.getCell(colNum);

if (cell == null)

cell = row.createCell(colNum);

cell.setCellValue(data);

fileOut = new FileOutputStream(path);

workbook.write(fileOut);

fileOut.close();

} catch (Exception e) {

e.printStackTrace();

return false;

}

return true;

}

public void setConditionalFormatting(String sheetName) {

// get row count

int rowCount;

int index = workbook.getSheetIndex(sheetName);

if (index == -1) {

rowCount = 0;

} else {

sheet = workbook.getSheetAt(index);

rowCount = sheet.getLastRowNum() + 1;

}

// Conditional formatting

SheetConditionalFormatting conditional = sheet.getSheetConditionalFormatting();

// Create rule

ConditionalFormattingRule rule = conditional.createConditionalFormattingRule(ComparisonOperator.NOT\_EQUAL,

"200");

// Change background color

PatternFormatting background = rule.createPatternFormatting();

background.setFillBackgroundColor(IndexedColors.RED.getIndex());

// Set cell range

CellRangeAddress[] range = { CellRangeAddress.valueOf("B2:B" + rowCount) };

conditional.addConditionalFormatting(range, rule);

try {

fileOut = new FileOutputStream(path);

workbook.write(fileOut);

fileOut.close();

workbook.close();

} catch (Exception e) {

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

}

}

}

Runner:

package utilties;

import java.io.FileInputStream;

import java.io.FileOutputStream;

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

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

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

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

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

import org.apache.poi.ss.util.CellRangeAddress;

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

import base.Base;

public class Xls\_Reader extends Base {

public Xls\_Reader(String path) {

this.path = path;

try {

fis = new FileInputStream(path);

workbook = new XSSFWorkbook(fis);

// sheet = workbook.getSheetAt(0);

fis.close();

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public int getRowCount(String sheetName) {

int index = workbook.getSheetIndex(sheetName);

if (index == -1) {

return 0;

} else {

sheet = workbook.getSheetAt(index);

int number = sheet.getLastRowNum() + 1;

return number;

}

}

public boolean setCellData(String sheetName, int colNum, int rowNum, String data) {

try {

fis = new FileInputStream(path);

workbook = new XSSFWorkbook(fis);

if (rowNum <= 0)

return false;

int index = workbook.getSheetIndex(sheetName);

if (index == -1)

return false;

sheet = workbook.getSheetAt(index);

row = sheet.getRow(0);

/\*

\* for(int i=0;i<row.getLastCellNum();i++){

\* //System.out.println(row.getCell(i).getStringCellValue().trim());

\* if(row.getCell(i).getStringCellValue().trim().equals(colName)) colNum=i; }

\*/

if (colNum == -1)

return false;

sheet.autoSizeColumn(colNum);

row = sheet.getRow(rowNum - 1);

if (row == null)

row = sheet.createRow(rowNum - 1);

cell = row.getCell(colNum);

if (cell == null)

cell = row.createCell(colNum);

cell.setCellValue(data);

fileOut = new FileOutputStream(path);

workbook.write(fileOut);

fileOut.close();

} catch (Exception e) {

e.printStackTrace();

return false;

}

return true;

}

public boolean setCellData(String sheetName, int colNum, int rowNum, int data) {

try {

fis = new FileInputStream(path);

workbook = new XSSFWorkbook(fis);

if (rowNum <= 0)

return false;

int index = workbook.getSheetIndex(sheetName);

if (index == -1)

return false;

sheet = workbook.getSheetAt(index);

row = sheet.getRow(0);

/\*

\* for(int i=0;i<row.getLastCellNum();i++){

\* //System.out.println(row.getCell(i).getStringCellValue().trim());

\* if(row.getCell(i).getStringCellValue().trim().equals(colName)) colNum=i; }

\*/

if (colNum == -1)

return false;

sheet.autoSizeColumn(colNum);

row = sheet.getRow(rowNum - 1);

if (row == null)

row = sheet.createRow(rowNum - 1);

cell = row.getCell(colNum);

if (cell == null)

cell = row.createCell(colNum);

cell.setCellValue(data);

fileOut = new FileOutputStream(path);

workbook.write(fileOut);

fileOut.close();

} catch (Exception e) {

e.printStackTrace();

return false;

}

return true;

}

public void setConditionalFormatting(String sheetName) {

// get row count

int rowCount;

int index = workbook.getSheetIndex(sheetName);

if (index == -1) {

rowCount = 0;

} else {

sheet = workbook.getSheetAt(index);

rowCount = sheet.getLastRowNum() + 1;

}

// Conditional formatting

SheetConditionalFormatting conditional = sheet.getSheetConditionalFormatting();

// Create rule

ConditionalFormattingRule rule = conditional.createConditionalFormattingRule(ComparisonOperator.NOT\_EQUAL,

"200");

// Change background color

PatternFormatting background = rule.createPatternFormatting();

background.setFillBackgroundColor(IndexedColors.RED.getIndex());

// Set cell range

CellRangeAddress[] range = { CellRangeAddress.valueOf("B2:B" + rowCount) };

conditional.addConditionalFormatting(range, rule);

try {

fileOut = new FileOutputStream(path);

workbook.write(fileOut);

fileOut.close();

workbook.close();

} catch (Exception e) {

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

}

}

}