package com.citi;

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.OutputStream;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Date;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import org.apache.poi.openxml4j.exceptions.InvalidFormatException;

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

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

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

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

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

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

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

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

import org.apache.poi.xssf.streaming.SXSSFWorkbook;

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

public class DataTruncation\_Phase2\_Styling {

static List<Map<Integer, Object[]>> dataList = null;

static Sheet sheet = null;

static File destFile = null;

static SXSSFWorkbook workbook = null;

static Map<Integer, Object[]> data = null;

static CellStyle existingHeaderStyle = null;

public static void main(String[] args)

throws InvalidFormatException, IOException {

destFile = new File("G://testing/dest.xlsx");

long startTime = System.currentTimeMillis();

File folder = new File("G://CitiFolder\_100");

File[] fileList = folder.listFiles();

dataList = new ArrayList<Map<Integer, Object[]>>();

for (File srcFile : fileList) {

data = new HashMap<Integer, Object[]>();

data = readDataFromFile(data, srcFile);

dataList.add(data);

System.out.println("Data list size:" + dataList.size());

}

System.out.println("File read successful!");

writeDataToFile(dataList);

long endTime = System.currentTimeMillis();

System.out.println("Total time:" + (endTime - startTime));

}

/\*\*

\* This method reads data from source excel file

\*

\* @param data

\* Map<Integer, Object[]>

\* @param srcFile

\* File

\* @return Map<Integer, Object[]>

\* @throws InvalidFormatException

\* @throws IOException

\*/

private static Map<Integer, Object[]> readDataFromFile(

Map<Integer, Object[]> data, File srcFile)

throws InvalidFormatException, IOException {

System.out.println("src file name:" + srcFile.getName());

Workbook srcWorkbook = new XSSFWorkbook(srcFile);

Sheet srcSheet = srcWorkbook.getSheetAt(0);

existingHeaderStyle = srcSheet.getRow(0).getCell(0).getCellStyle();

long srcRowCount = srcSheet.getPhysicalNumberOfRows();

int cellCount = srcSheet.getRow(0).getPhysicalNumberOfCells();

DataFormatter formatter = new DataFormatter();

for (int rowNum = 0; rowNum < srcRowCount; rowNum++) {

Object[] obj = new Object[cellCount];

Row row = srcSheet.getRow(rowNum);

if (row != null) {

for (int cellNum = 0; cellNum < cellCount; cellNum++) {

Cell cell = row.getCell(cellNum);

if (cell != null) {

obj[cellNum] = formatter.formatCellValue(cell);

}

}

data.put(rowNum, obj);

}

}

srcWorkbook.close();

return data;

}

/\*\*

\* This method writes data to excel

\*

\* @param dataList

\* List<Map<Integer, Object[]>>

\* @return boolean

\*/

private static boolean writeDataToFile(

List<Map<Integer, Object[]>> dataList) {

workbook = new SXSSFWorkbook(500);

sheet = workbook.createSheet();

int rowNum = 0;

//Creating style for header row

CellStyle newHeaderStyle = workbook.createCellStyle();

newHeaderStyle.cloneStyleFrom(existingHeaderStyle);

//Creating style for target cells

CellStyle targetCellStyle = workbook.createCellStyle();

targetCellStyle.setFillForegroundColor(

IndexedColors.GREY\_25\_PERCENT.getIndex());

targetCellStyle.setFillPattern(FillPatternType.SOLID\_FOREGROUND);

CellStyle styleToUse = null;

//Code to iterate the datalist, retrieve each datalist element and map to new cell

// in destination workbook

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

// System.out.println("Row num after file:" + rowNum);

Map<Integer, Object[]> tempMap = dataList.get(i);

System.out.println("Getting file data:" + i);

//Retrieving object array from single key

for (Map.Entry<Integer, Object[]> entry : tempMap.entrySet()) {

Row row = sheet.createRow(rowNum++);

int cellNum = 0;

Object[] objArr = entry.getValue();

List<Object> tempList = Arrays.asList(objArr);

//Code to set cell styles

if (tempList.contains("RESULTRECORDID")

&& !tempList.contains("COL\_SRC")

&& !tempList.contains("SRC")

&& !tempList.contains("TGT")) {

styleToUse = newHeaderStyle;

} else if (tempList.contains("TGT")

&& !tempList.contains("RESULTRECORDID")

&& !tempList.contains("SRC")

&& !tempList.contains("COL\_SRC")) {

styleToUse = targetCellStyle;

} else if (tempList.contains("SRC") && !tempList.contains("TGT")

&& !tempList.contains("RESULTRECORDID")

&& !tempList.contains("COL\_SRC")) {

styleToUse = null;

} else {

styleToUse = null;

}

//Setting individual object array elements to new cell values

for (Object obj : objArr) {

Cell cell = row.createCell(cellNum++);

if (obj instanceof String) {

cell.setCellValue((String) obj);

cell.setCellStyle(styleToUse);

} else if (obj instanceof Integer) {

cell.setCellValue((Integer) obj);

cell.setCellStyle(styleToUse);

} else if (obj instanceof Date) {

cell.setCellValue((Date) obj);

cell.setCellStyle(styleToUse);

} else if (obj instanceof Double) {

cell.setCellValue((Double) obj);

cell.setCellStyle(styleToUse);

} else if (obj instanceof Long) {

cell.setCellValue((Long) obj);

cell.setCellStyle(styleToUse);

}

}

}

}

//Writing data to destination workbook

OutputStream stream = null;

try {

System.out.println("Writing to file...");

stream = new FileOutputStream(destFile);

if (null != workbook && null != stream) {

workbook.write(stream);

stream.close();

workbook.close();

System.out.println("Writing over!");

}

} catch (Exception e) {

e.printStackTrace();

}

return true;

}

}