# ExcelDownload:

@RequestMapping(method=RequestMethod.***POST***,value="exportOrderData",consumes="application/json")

@ResponseBody **public** **void** exportOrderData(HttpServletResponse response) **throws** IOException, ParseException {

List<Map<String,Object>> partsData= objService.getOrderExportData();

SXSSFWorkbook workbook = **new** SXSSFWorkbook(1000);

Sheet sheet = workbook.createSheet("Order Data");

List<String> headerList = **new** ArrayList<>();

Font headerFont = setFont(workbook,FMSVariableConstants.***HEADER***);

Font dataFont = setFont(workbook,FMSVariableConstants.***DATA***);

CellStyle headerStyle = setStyle(workbook, FMSVariableConstants.***HEADER***);

CellStyle dateStyle = setStyle(workbook,FMSVariableConstants.***DATE***);

CellStyle dataStyle = setStyle(workbook, FMSVariableConstants.***DATA***);

Row row = sheet.createRow(0);

**int** cellNo = 0;

**for**(String key:partsData.get(0).keySet()){

headerList.add(key);

}

**for** (String header : headerList) {

Cell cell = row.createCell(cellNo);

headerStyle.setFont(headerFont);

cell.setCellValue(header);

cell.setCellStyle(headerStyle);

cellNo++;

}

**int** currRow = 0;

**for**(Map<String,Object> record: partsData){

dataStyle.setFont(dataFont);

dataStyle.setFont(dataFont);

currRow++;

row = sheet.createRow(currRow);

cellNo = 0;

**for** (Iterator<Entry<String, Object>> iterator = record.entrySet()

.iterator(); iterator.hasNext();) {

Entry<String, Object> entry = iterator.next();

cellSet(FMSVariableConstants.***STRING***,dateStyle,dataStyle,row,cellNo++,Utils.*getValidation*(entry.getValue()));

}

}

Utils.*downloadFile*("Order-Data.xlsx", workbook, response);

}

**public** Cell cellSet(String type, CellStyle dateStyle, CellStyle dataStyle, Row row, **int** cellNo, Object input){

**int** cellN = cellNo;

Cell dataC = row.createCell(cellN);

dataC.setCellStyle(dataStyle);

dataC.setCellValue((String)input);

**return** dataC;

}

**public** **static** **void** downloadFile(String fileName, SXSSFWorkbook workbook, HttpServletResponse response) {

**try**

{

ByteArrayOutputStream bos = **new** ByteArrayOutputStream();

workbook.write(bos);

bos.close();

**byte**[] bytes = bos.toByteArray();

response.setHeader(FMSVariableConstants.***CONTENTDISPOSITION***, "attachment; filename="+fileName);

response.getOutputStream().write(bytes);

response.flushBuffer();

}

**catch**(IOException e)

{

***log***.info(e);

}

}

**public** Font setFont(SXSSFWorkbook wb, String type){

Font font = wb.createFont();

font.setFontHeightInPoints((**short**) 10);

font.setFontName(FMSVariableConstants.***GE\_INSPIRA\_FONT***);

font.setItalic(**false**);

**if**(FMSVariableConstants.***HEADER***.equalsIgnoreCase(type))

{

font.setBoldweight(XSSFFont.***BOLDWEIGHT\_BOLD***);

}

**return** font;

}

**public** CellStyle setStyle(SXSSFWorkbook wb, String type){

CellStyle style = wb.createCellStyle();

**if**(FMSVariableConstants.DATE.equalsIgnoreCase(type)) {

CreationHelper createHelper = wb.getCreationHelper();

style.setDataFormat(createHelper.createDataFormat().getFormat("yyyy-MM-dd"));

}

style.setAlignment(CellStyle.ALIGN\_LEFT);

style.setBorderBottom(CellStyle.BORDER\_THIN);

style.setBorderLeft(CellStyle.BORDER\_THIN);

style.setBorderRight(CellStyle.BORDER\_THIN);

style.setBorderTop(CellStyle.BORDER\_THIN);

style.setBottomBorderColor(IndexedColors.BLACK.getIndex());

style.setLeftBorderColor(IndexedColors.BLACK.getIndex());

style.setRightBorderColor(IndexedColors.BLACK.getIndex());

style.setTopBorderColor(IndexedColors.BLACK.getIndex());

**return** style;

}

# Service Call With PathVariable and RequestBoby map

@RequestMapping(method=RequestMethod.***POST***, value="getIPMData/{ServiceType}")

@ResponseBody **public** List<Map<String, Object>> getIPMData(@PathVariable("ServiceType") String type,@RequestBody Map<String,Object> filterData){

**return** objDao.getIPMData(filterData,type);

}

**public** List<Map<String, Object>> getIPMData(Map<String,Object> filterData,String type) {

String query = **null**;

Object[] params = **new** Object[24];

params[0] = Utils.getValidation(filterData.get("businessSegment"));

**return** jdbc.query(query,params,**new** MiscRowMapper());

}

**public** **class** MiscRowMapper **implements** RowMapper<Map<String, Object>> {

@Override

**public** Map<String, Object> mapRow(ResultSet resultSet, **int** arg1)

**throws** SQLException {

ResultSetMetaData meta = resultSet.getMetaData();

Map<String, Object> results = **new** LinkedHashMap<>();

**for** (**int** index = 1; index <= meta.getColumnCount(); index++) {

results.put(meta.getColumnName(index), resultSet.getObject(index));

}

**return** results;

}

}

# Import/Export CSV to/from Table

import org.postgresql.copy.CopyManager;

import org.postgresql.core.BaseConnection;

@RequestMapping(value = "/importCSVToTable", headers = "Content-Type= multipart/form-data",method = RequestMethod.***POST***)

@Consumes("multipart/form-data")

@ResponseBody **public** String importCSVToTable(@RequestParam("file") MultipartFile file) **throws** IOException{

System.***out***.println("calling importCSVToTable");

**return** objService.importCSVToTable(file.getInputStream());

}

**public** String importCSVToTable(InputStream inputStream) {

String response="";

**try** {

Connection con = DriverManager.*getConnection*(url,user,pass);

CopyManager copyManager = **new** CopyManager((BaseConnection) con);

jdbc.execute("truncate table fms\_ipm\_csv\_parts\_import");

StringBuilder sql = **new** StringBuilder();

sql.append("COPY ");

sql.append("fms\_ipm\_csv\_parts\_import(concatenate,line\_id)");

sql.append(" FROM STDIN WITH (");

sql.append(" FORMAT CSV ");

sql.append(", DELIMITER ','");

sql.append(", NULL ''");

sql.append(", HEADER TRUE");

sql.append(", QUOTE '\"'");

sql.append(", ESCAPE '\"' ");

sql.append(", ENCODING 'UTF8'");

sql.append(")");

log.info("Import started...");

copyManager.copyIn(sql.toString(), inputStream );

log.info("Import Completed!");

log.info("Records updation started...");

jdbc.update("update fms\_ipm\_parts\_edit\_fields AS fed set p\_rev\_rec= tmp.p\_rev\_rec FROM (Select p\_rev\_rec, concatenate from fms\_ipm\_csv\_parts\_import) AS tmp WHERE tmp.concatenate=fed.p\_concatenate");

log.info("Records updated!");

System.***out***.println("File imported!!");

response = FMSVariableConstants.***SUCCESS***;

}**catch**(Exception e){

e.printStackTrace();

response = FMSVariableConstants.***FAILURE***;

log.info(e);

}

**return** response;

}

**public** **void** exportCSVFromTable(HttpServletResponse responses) {

**try** {

Connection con = DriverManager.*getConnection*(url,user,pass);

CopyManager copyManager = **new** CopyManager((BaseConnection) con);

ByteArrayOutputStream out = **new** ByteArrayOutputStream();

StringBuilder sql = **new** StringBuilder();

sql.append("COPY ");

sql.append("(" + FMSQueryConstants.***PARTS\_DATA\_EXPT*** + ")");

sql.append(" TO STDOUT WITH (");

sql.append(" FORMAT CSV ");

sql.append(", DELIMITER ','");

sql.append(", NULL ''");

sql.append(", HEADER TRUE");

sql.append(", QUOTE '\"'");

sql.append(", ESCAPE '\"' ");

sql.append(", ENCODING 'UTF8'");

sql.append(")");

log.info("Export Started");

copyManager.copyOut(sql.toString(), out);

**byte**[] bytes = out.toByteArray();

responses.setHeader(FMSVariableConstants.***CONTENTDISPOSITION***, "attachment; filename=IPM-Parts-Data.csv");

responses.getOutputStream().write(bytes);

responses.flushBuffer();

}**catch**(Exception e){

log.info(e);

}

}

# Any file store in db and download from db instead of file system.

Controller:

@RequestMapping(value = "/uploadDocuments", headers = "Content-Type= multipart/form-data",method = RequestMethod.POST)

@Consumes("multipart/form-data")

@ResponseBody public String uploadDocuments(@RequestParam("file") MultipartFile file, @RequestParam("fileDescription") String fileDescription, @RequestParam("userSSO") String userSSO, @RequestParam("fileAccess") String fileAccess) throws IOException{

return objService.uploadDocuments(file.getInputStream(), file.getOriginalFilename(),fileDescription, userSSO, fileAccess);

}

Dao:

@Override

public String uploadDocuments(InputStream inputStream, String fileName,String fileDescription,String userSSO,String fileAccess) {

String response = FMSVariableConstants.FAILURE;

try {

if(fileAccess !="" && !fileAccess.isEmpty()){

Object[] params = {fileDescription,fileName, IOUtils.toByteArray(inputStream),userSSO, roleID};

int[] types = {Types.VARCHAR, Types.VARCHAR, Types.BINARY, Types.VARCHAR, Types.VARCHAR};

int id = jdbc.queryForObject(FMSQueryConstants.INSERT\_DOCUMENT\_QUERY, params, types, Integer.class);

String fileAccessArr[] = fileAccess.split(",");

List<Object[]> inputList = new ArrayList<Object[]>();

for (String roleId : fileAccessArr) {

Object[] tmp = {id, Integer.valueOf(roleId)};

inputList.add(tmp);

System.out.println("RoleID:" + roleId);

}

jdbc.batchUpdate("insert into fms\_documents\_role(doc\_id,role\_id) values(?,?)", inputList);

response = FMSVariableConstants.SUCCESS;

}

}catch(Exception e) {

e.printStackTrace();

response = FMSVariableConstants.FAILURE;

}

return response;

}

@Override

public void downloadDocuments(HttpServletResponse responses, Map<String, Object> data) {

try {

if(Utils.getIntegerValidation(data.get(FMSVariableConstants.ID)) != 0){

Map<String, Object> fileData = jdbc.queryForMap(FMSQueryConstants.SELECT\_DOCUMENT\_QUERY, new Object[] {Utils.getIntegerValidation(data.get(FMSVariableConstants.ID))});

String filename = fileData.get("filename").toString();

byte[] bytes = (byte[]) fileData.get("content");

responses.setHeader(FMSVariableConstants.CONTENTDISPOSITION, "attachment; filename="+filename);

responses.getOutputStream().write(bytes);

responses.flushBuffer();

}

}catch(Exception e){}

}