**NOTE APP MIGRATION**

1. hpt.has.shinhan.filenet:
2. Migration:

//

**public** **static** **void** Handler(String \_url\_filenet, String \_user\_filenet, String \_pass\_filenet, String \_optional\_filenet, String \_obj\_filenet, String \_class\_doc, String \_root\_filenet, String \_fol\_filenet, String \_url\_oracle, String \_port\_oracle, String \_data\_name, String \_user\_oracle, String \_pass\_oracle, String \_driver\_oracle, String \_url\_get\_file, String \_url\_DB2\_Log, String \_user\_log, String \_pass\_log, String \_driver\_db2, **int** \_volume, String from, String to, **int** rownum) {

**try** {

//Get directory từ thư mục user bằng file dklog.log

String currentDirectory = System.*getProperty*("user.dir");

//Tạo ra một connection đến server oracle database

Connection conPro = ConnectionOracle.*CreateConnection*(\_url\_oracle, \_port\_oracle, \_data\_name, \_user\_oracle, \_pass\_oracle, \_driver\_oracle);

//Tạo ra câu truy vấn sql theo điều kiện thời gian bắt đầu và kết thúc

String sqlInfo = QueryOracle.*QueryStringOracle1*(from, to, rownum);

//Truy vấn câu lệnh sql để lấy ra được danh sách metadata và path get file

List<MetaData> lstPro = QueryOracle.*ExcuteQuerySelect*(conPro, sqlInfo);

//Chia nhỏ danh sách lớn thành nhiều danh sách nhỏ để chạy mutithread

List<List<MetaData>> lst = DivideList.*distributeList2*(lstPro, \_volume);

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

//Khởi tạo từng thread để upload lên filenet

SegmentMigrate seg = **new** SegmentMigrate(lst.get(i), \_url\_filenet, \_user\_filenet, \_pass\_filenet, \_optional\_filenet, \_obj\_filenet, \_class\_doc, \_root\_filenet, \_fol\_filenet, \_url\_get\_file, \_driver\_db2, \_url\_DB2\_Log, \_user\_log, \_pass\_log, currentDirectory, (i + 1)); //run thread

seg.start();

}

//Đóng connection oracle lại

ConnectionOracle.*CloseConnection*(conPro);

}

**catch**(Exception ex) {

System.***out***.println("hpt.has.shinhan.filenet.Migration.Handler.Error Handler: " + ex);

}

}

1. hpt.has.shinhan.oracle:
2. CreateConnection:

//Tạo ra connection từ app đến server database oracle

// host server: localhost

//port: 1521

//data: database name

//username: userId

//password: password

//driver: jdbcClassName = "oracle.jdbc.driver.OracleDriver"

**public** **static** Connection CreateConnection(String host, String port, String data, String username, String password, String driver) {

Connection con = **null**;

**try** {

Class.*forName*(driver);

String conectionString = "jdbc:oracle:thin:@" + host + ":" + port + ":" + data;

con = DriverManager.*getConnection*(

conectionString, username, password);

**if**(con != **null**) {

System.***out***.println("hpt.has.shinhan.oracle.ConnectionOracle.CreateConnection.Infor Connection oracle db successful");

}

**else** {

System.***out***.println("hpt.has.shinhan.oracle.ConnectionOracle.CreateConnection.Infor Connection oracle db fail");

}

}

**catch**(Exception e) {

System.***out***.println("hpt.has.shinhan.oracle.ConnectionOracle.CreateConnection.Error: " + e);

}

**return** con;

}

1. QueryStringOracle1:

//from: ngày bắt đầu theo định dạng dd-mm-yyyy

//to: ngày kết thúc theo định dạng dd-mm-yyyy

//rownum: là giới hạn số dòng đc lấy

**public** **static** String QueryStringOracle1(String from, String to, **int** rownum) {

String sql = "SELECT boxid, appfrmno, apprefno, cif, create\_dt, custid, custname, docdesc, docid, docrefid, filename, filepath, item, loanagrno, mimetype, scanneddate, update\_dt, uploadchannel FROM icm\_document WHERE";

**try** {

sql += " (create\_dt >= to\_date('" + from + "', 'DD-MM-YYYY') AND create\_dt <= to\_date('" + to + "', 'DD-MM-YYYY')";

**if**(rownum >= 0) {

sql += " and ROWNUM <= " + rownum + "";

}

}

**catch**(Exception e) {

System.***out***.println("hpt.has.shinhan.oracle.QueryOracle.QueryStringOracle.Error: " + e);

}

**return** sql;

}

1. ExcuteQuerySelect

//con: connection từ app đến server của database oracle

//sql: là câu lên truy vấn của oracle

//List<MetaData>: lấy ra được danh sách thông tin metadata được query từ câu truy vấn sql

**public** **static** List<MetaData> ExcuteQuerySelect(Connection con, String sql){

List<MetaData> lst = **new** ArrayList<MetaData>();

**try** {

Statement st = con.createStatement(ResultSet.***TYPE\_SCROLL\_SENSITIVE***, ResultSet.***CONCUR\_READ\_ONLY***);

ResultSet rset = st.executeQuery(sql);

**while**(rset.next()) {

System.***out***.println("hpt.has.shinhan.oracle.QueryOracle.ExcuteQuerySelect.Info DOCREFID: " + rset.getString("DOCREFID"));

//Tạo ra object thông tin metadata của 1 record được truy vấn ra

MetaData m = **new** MetaData(rset.getString("DOCREFID"), rset.getString("APPFRMNO"), rset.getString("LOANAGRNO"), rset.getString("DOCDESC"), rset.getString("CUSTID"), rset.getString("CUSTNAME"), rset.getDate("SCANNEDDATE"), rset.getString("BOXID"), rset.getString("MIMETYPE"), rset.getString("FILEPATH"), rset.getString("CIF"), rset.getString("DOCID"), rset.getString("ITEM"), rset.getString("UPLOADCHANNEL"), rset.getString("APPREFNO"), rset.getString("FILENAME"), rset.getDate("CREATE\_DT"), rset.getDate("UPDATE\_DT"));

m.setList();

lst.add(m);

}

}

**catch**(Exception e) {

System.***out***.println("hpt.has.shinhan.oracle.QueryOracle.ExcuteQuerySelect.Error: " + e);

}

**return** lst;

}

1. hpt.has.shinhan.thread
2. DivideList:

//mainList: list object lớn bị chia ra thành nhiều list nhỏ

//int subSize: số lượng list nhỏ được chia

//List<List<M>>: danh sách list nhỏ được tạo ra từ list object lớn

**public** **static** <M> List< List<M> > distributeList2(List<M> mainList, **int** subSize)

{

List< List<M> > listArray = **new** ArrayList<>();

**try** {

**int** divineNumber = mainList.size() / subSize;

**int** from = 0;

**int** to = from + divineNumber;

**if**(mainList.size() > 1)

{

**for**(**int** i = 0; i < subSize; i++)

{

List<M> subListArray = mainList.subList(from, to);

from = to;

to = to + divineNumber;

**if**(i + 1 == subSize - 1)

{

to = mainList.size();

}

listArray.add(subListArray);

}

}

**else**

{

listArray.add(mainList);

}

}

**catch**(Exception e) {

System.***out***.println("hpt.has.shinhan.thread.DivideList.distributeList.Error distributeList: " + e);

}

**return** listArray;

}

1. SegmentMigrate

//Contructor lớp SegmentMigrate extents từ lớp Thread

**public** SegmentMigrate(List<MetaData> \_lst\_meta\_data, String \_url\_filenet, String \_user\_filenet, String \_pass\_filenet, String \_optional\_filenet, String \_obj\_filenet, String \_class\_doc, String \_root, String \_folder, String \_url\_get\_file, String \_driver, String \_url\_Log, String \_user\_log, String \_pass\_log, String \_current\_dir, **int** \_volume) {

**this**.LIST\_META\_DATA = \_lst\_meta\_data;

**this**.OBJECT\_TYPE = \_class\_doc;

**this**.ROOT\_FOLDER = \_root;

**this**.FOLDER\_ID = \_folder;

**this**.URL\_FILENET = \_url\_filenet;

**this**.HOST\_GET\_FILE = \_url\_get\_file;

**this**.CURRENT\_DIRECTORY = \_current\_dir;

**this**.VOLUME = \_volume;

**this**.URL\_FILENET = \_url\_filenet;

**this**.USER\_NAME\_FILENET = \_user\_filenet;

**this**.PASS\_WORD\_FILENET = \_pass\_filenet;

**this**.OPTIONAL\_FILENET = \_optional\_filenet;

**this**.OBJECT\_STORE = \_obj\_filenet;

**this**.DRIVER\_DB2 = \_driver;

**this**.URL\_LOG = \_url\_Log;

**this**.USER\_NAME\_LOG = \_user\_log;

**this**.PASS\_WORD\_LOG = \_pass\_log;

}

1. Override function run():

@Override

**public** **void** run() {

**try** {

// Tạo connection từ app đến hệ thông filenet

ObjectStore object\_store = ConnectFilenet.*OpenConnection*(URL\_FILENET, USER\_NAME\_FILENET, PASS\_WORD\_FILENET, OPTIONAL\_FILENET, OBJECT\_STORE);

//Tạo connection từ app đến database db2

Connection conLog = ConnectionDB2.*ConnectDB2*(DRIVER\_DB2, URL\_LOG, USER\_NAME\_LOG, PASS\_WORD\_LOG);

// **TODO** Auto-generated method stub

**for**(**int** i = 0; i < LIST\_META\_DATA.size(); i++) {

//Tạo ra thư mục từ root thư mục và folder object type (Sau này sẽ ko dùng hàm khi thiết kế lại cấu trúc thư mục)

Boolean create = CreateFolder.*createFolder*(object\_store, ROOT\_FOLDER, LIST\_META\_DATA.get(i).getProperty("CUSTID"), FOLDER\_ID);

//Get được full path: url + path

String pathFull = HOST\_GET\_FILE + LIST\_META\_DATA.get(i).getProperty("FILEPATH");

//Tạo file server xuống từ full path và lưu trữ tại dir

String dir = DownloadFileURL.*DownloadFile1*(pathFull, CURRENT\_DIRECTORY, LIST\_META\_DATA.get(i).getProperty("MIMETYPE"));

**int** del = 1;

File file = **null**;

//Tạo id log input và output theo guid

String input = GenerateUniqueIDUsingGUID.*generateUniqueID*();

String out = GenerateUniqueIDUsingGUID.*generateUniqueID*();

**if**(dir == **null**) {

//Nếu file server tải về lỗi hay ko tồn tại sẽ ko upload lên filenet mà sẽ ghi log lại để sau này truy vấn

del = 0;

String insert1 = QueryDataDB2.*QueryStringInsertInput*(input, LIST\_META\_DATA.get(i).getProperty("ITEM"), LIST\_META\_DATA.get(i).getProperty("DOCREFID"), LIST\_META\_DATA.get(i).getProperty("CUSTID"), pathFull, 0, del, 0, "false");

QueryDataDB2.*ExcuteQueryInsert*(conLog, insert1);

String insert2 = QueryDataDB2.*QueryStringInsertOutput*(out, **null**, input, create);

QueryDataDB2.*ExcuteQueryInsert*(conLog, insert2);

}

**else** {

// file = new File(dir);

// String output = CURRENT\_DIRECTORY + "\\" + LIST\_META\_DATA.get(i).getProperty("FILENAME");

// String encodestring = ConvertBase64.encodeFileToBase64Binary(file);

// dir = ConvertBase64.writeBase64ToFile(encodestring, output, LIST\_META\_DATA.get(i).getProperty("MIMETYPE"));

// file.delete();

//Nếu file server được tải thành công

file = **new** File(dir);

//Sẽ ghi log input lại

String insert1 = QueryDataDB2.*QueryStringInsertInput*(input, LIST\_META\_DATA.get(i).getProperty("ITEM"), LIST\_META\_DATA.get(i).getProperty("DOCREFID"), LIST\_META\_DATA.get(i).getProperty("CUSTID"), pathFull, file.length(), del, 3, "true");

QueryDataDB2.*ExcuteQueryInsert*(conLog, insert1);

//Thực hiện thao tác upload file đó lên filenet cùng thông tin property lấy từ metadata của file đó

String id = UploadFilenet.*UploadDocument2*(object\_store, file, OBJECT\_TYPE, LIST\_META\_DATA.get(i).getProperty("FILENAME"), ROOT\_FOLDER + "/" + LIST\_META\_DATA.get(i).getProperty("CUSTID"), LIST\_META\_DATA.get(i));

//Ghi log output lại

String insert2 = QueryDataDB2.*QueryStringInsertOutput*(out, id, input, create);

QueryDataDB2.*ExcuteQueryInsert*(conLog, insert2);

file.delete();

}

}

//Đóng connection filenet và db2

ConnectFilenet.*CloseConnection*();

ConnectionDB2.*CloseBD2*(conLog);

//ConnectionDB2.CloseBD2(conPath);

}

**catch**(Exception e) {

System.***out***.println("hpt.has.shinhan.thread.SegmentMigrate.run.Error run: " + e);

}

}