package com.maan.eway.service.impl;

import java.math.BigDecimal;

import java.text.DecimalFormat;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

import java.util.Map;

import java.util.Map.Entry;

import java.util.concurrent.Callable;

import java.util.concurrent.ConcurrentLinkedQueue;

import java.util.concurrent.ForkJoinPool;

import java.util.concurrent.Future;

import java.util.stream.Collectors;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

import org.dozer.DozerBeanMapper;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Lazy;

import org.springframework.stereotype.Service;

import com.maan.eway.bean.EserviceCustomerDetails;

import com.maan.eway.bean.EserviceMotorDetails;

import com.maan.eway.bean.MsCommonDetails;

import com.maan.eway.bean.MsCustomerDetails;

import com.maan.eway.bean.MsVehicleDetails;

import com.maan.eway.config.onetime.Thread\_OneTime;

import com.maan.eway.config.thread.MyTaskList;

import com.maan.eway.repository.EServiceMotorDetailsRepository;

import com.maan.eway.repository.EserviceCustomerDetailsRepository;

import com.maan.eway.repository.MsCommonDetailsRepository;

import com.maan.eway.repository.MsCustomerDetailsRepository;

import com.maan.eway.repository.MsVehicleDetailsRepository;

import com.maan.eway.req.OneTimeTableReq;

import com.maan.eway.res.MsCommonDetailsRes;

import com.maan.eway.res.MsCustomerDetailsRes;

import com.maan.eway.res.MsDetailsRes;

import com.maan.eway.res.MsVehicleDetailsRes;

import com.maan.eway.res.OneTimeTableRes;

import com.maan.eway.service.OneTimeService;

@Service

public class OneTimeServiceImpl implements OneTimeService {

@Lazy

@Autowired

private OneTimeService otSer;

private ForkJoinPool forkjoin = new ForkJoinPool(2);

@Autowired

private EserviceCustomerDetailsRepository eserviceCustomerRepo ;

@Autowired

private MsCustomerDetailsRepository msCustomerRepo ;

@Autowired

private EServiceMotorDetailsRepository eserviceMotorRepo ;

@Autowired

private MsVehicleDetailsRepository msVehicleRepo ;

@Autowired

private MsCommonDetailsRepository msCommonRepo ;

private Logger log = LogManager.getLogger(OneTimeTableServiceImpl.class);

@Override

public OneTimeTableRes call\_OT\_Insert(OneTimeTableReq req) {

OneTimeTableRes response = new OneTimeTableRes();

SimpleDateFormat sdf = new SimpleDateFormat("yyMMddhhmmssSS");

try {

List<Callable<Object>> queue = new ArrayList<Callable<Object>>();

MyTaskList taskList = new MyTaskList(queue);

Thread\_OneTime msVehicle = new Thread\_OneTime("MSVehicle", req, otSer);

Thread\_OneTime msCustomer = new Thread\_OneTime("MSCustomer", req, otSer);

queue.add(msVehicle);

queue.add(msCustomer);

ConcurrentLinkedQueue<Future<Object>> invoke = (ConcurrentLinkedQueue<Future<Object>>) forkjoin

.invoke(taskList);

int success = 0;

List<String> motorData = new ArrayList<String>();

String cdRefno = "";

String msCust=null;

String msVehDet = null;

for (Future<Object> callable : invoke) {

log.info(callable.getClass() + "," + callable.isDone());

if (callable.isDone()) {

Map<String, Object> map = (Map<String, Object>) callable.get();

for (Entry<String, Object> future : map.entrySet()) {

if ("MSVehicle".equalsIgnoreCase(future.getKey())) {

motorData = (List<String>) future.getValue();

} else if ("MSCustomer".equalsIgnoreCase(future.getKey())) {

//cdRefno = (String) future.getValue();

cdRefno = (String) future.getValue();

}

}

success++;

}

}

String vdRefno = motorData.get(0) ;

String sectionId = motorData.get(1) ;

String agencyCode = motorData.get(2) ;

String branchCode = motorData.get(3) ;

String productId = motorData.get(4) ;

String companyId = motorData.get(5) ;

String msRefNo = "" ;

String complete = (queue.size() == success) ? "yes" : "No";

log.info("call\_OTInsert--> complete: " + complete);

// Save Common Details

MsCommonDetails findData = msCommonRepo.findByCdRefnoAndVdRefno(Long.valueOf(cdRefno) ,Long.valueOf(vdRefno) ) ;

if( findData !=null) {

msRefNo = String.valueOf(findData.getMsRefno());

} else {

msRefNo = sdf.format(new Date()) ; ;

MsCommonDetails saveData = new MsCommonDetails();

saveData.setAgencyCode(agencyCode);

saveData.setBranchCode(branchCode);

saveData.setInsuranceId(companyId);

saveData.setProductId(Integer.valueOf(productId)) ;

saveData.setRequestreferenceno(req.getRequestReferenceNo());

saveData.setCdRefno(Long.valueOf(cdRefno));

saveData.setEntryDate(new Date());

saveData.setVdRefno(Long.valueOf(vdRefno));

saveData.setMsRefno(Long.valueOf(msRefNo));

saveData.setStatus("Y");

saveData.setSectionId(Integer.valueOf(sectionId));

msCommonRepo.save(saveData);

}

//Response

response.setCdRefNo(cdRefno);

response.setVdRefNo(vdRefno);

response.setMsRefNo(msRefNo);

response.setRequestReferenceNo(req.getRequestReferenceNo());

return response;

} catch (Exception e) {

e.printStackTrace();

log.info("Exception is --> " + e.getMessage());

return null;

}

}

@Override

public String call\_MSCustomer(OneTimeTableReq request) {

String cdRefNo = "" ;

DozerBeanMapper dozerMapper = new DozerBeanMapper();

SimpleDateFormat sdf = new SimpleDateFormat("yyMMddhhmmssSS");

try {

EserviceCustomerDetails findData = eserviceCustomerRepo.findByRequestReferenceNo(request.getRequestReferenceNo() );

List<MsCustomerDetails> oldDatas = msCustomerRepo.findByIdNumber(findData.getIdNumber());

List<MsCustomerDetails> filterData = oldDatas.stream().filter( o -> o.getPolicyHolderTypeid().equals(findData.getPolicyHolderTypeid()) &&

o.getIdType().equals(findData.getIdType()) && o.getAge().equals(findData.getAge()) && o.getGender().equals(findData.getGender()) &&

o.getOccupation().equals(findData.getOccupation()) && o.getBusinessType().equals(findData.getBusinessType()) && o.getRegionCode().equals(findData.getRegionCode())

&& o.getCityCode().equals(findData.getCityCode()) && o.getTaxExemptedId().equals(findData.getTaxExemptedId()) && o.getStatus().equals("Y") ).collect(Collectors.toList());

if (filterData.size() > 0 ) {

cdRefNo =String.valueOf( filterData.get(0).getCdRefno()) ;

} else {

cdRefNo = sdf.format(new Date()) ;

MsCustomerDetails saveNewEntry = new MsCustomerDetails();

dozerMapper.map(findData, saveNewEntry);

saveNewEntry.setCdRefno(Long.valueOf(cdRefNo) );

saveNewEntry.setEntryDate(new Date());

msCustomerRepo.save(saveNewEntry);

}

}catch (Exception e) {

e.printStackTrace();

log.error("Exception is ---> " + e.getMessage());

return null ;

}

return cdRefNo;

}

@Override

public List<String> call\_MSVehicle(OneTimeTableReq request) {

List<String> resList = new ArrayList<String>();

String vdRefNo = "" ;

String sectionId = "" ;

String agencyCode = "" ;

String branchCode = "" ;

String productId = "" ;

String companyId = "" ;

SimpleDateFormat sdf = new SimpleDateFormat("yyMMddhhmmssSS");

String pattern = "#####0";

DecimalFormat df = new DecimalFormat(pattern);

DozerBeanMapper dozerMapper = new DozerBeanMapper();

try {

List<EserviceMotorDetails> motorDatas = eserviceMotorRepo.findByRequestReferenceNo(request.getRequestReferenceNo());

boolean newEntry = false ;

sectionId = motorDatas.get(0).getSectionId() ;

branchCode = motorDatas.get(0).getBranchCode();

agencyCode = motorDatas.get(0).getAgencyCode();

productId = motorDatas.get(0).getProductId();

companyId = motorDatas.get(0).getCompanyId();

for (EserviceMotorDetails data : motorDatas) {

List<MsVehicleDetails> vehicleDatas = msVehicleRepo.findByChassisNumber(data.getChassisNumber() );

List<MsVehicleDetails> filterData = new ArrayList<MsVehicleDetails>();

List<MsVehicleDetails> filterData2 = new ArrayList<MsVehicleDetails>();

List<MsVehicleDetails> filterData3 = new ArrayList<MsVehicleDetails>();

if(vehicleDatas.size()>0) {

// Uk 1

filterData = vehicleDatas.stream().filter( o -> o.getVehicleId().equals(String.valueOf( data.getVehicleId())) &&

o.getAccidentYn().equals(data.getAccident()) && o.getWindScreenCoverRequeired().equals(data.getWindScreenCoverRequired()) &&

o.getInsuranceType().equals(data.getInsuranceType()) && o.getInsuranceClass().equals(data.getInsuranceClass()) &&

o.getOwnerCategory().equals(data.getOwnerCategory()) && o.getChassisNumber().equals(data.getChassisNumber()) &&

o.getVehicleMake().equals(data.getVehicleMake()) && o.getVehcileModel().equals(data.getVehcileModel()) &&

o.getVehicleBodyType().equals(data.getVehicleType()) && o.getVehicleWeight().equals(Integer.valueOf(df.format( data.getGrossWeight())))

&& o.getStatus().equals("Y") ).collect(Collectors.toList());

// Uk2

filterData2 = vehicleDatas.stream().filter( o -> o.getFuelType().equals(data.getFuelType()) &&

o.getManufactureYear().equals(data.getManufactureYear()) &&

o.getManufactureAge().equals(data.getManufactureAge()) && o.getRegistrationAge().equals(data.getRegistrationAge()) &&

o.getRegistrationYear().equals(data.getRegistrationYear()) && o.getSeatingCapacity().equals(String.valueOf(data.getSeatingCapacity())) &&

o.getPeriodOfInsurance().equals(data.getPeriodOfInsurance()) && o.getWindScreenSumInsured().equals(data.getWindScreenSumInsured()) &&

o.getAcccessoriesSumInsured().equals(data.getAcccessoriesSumInsured()) && o.getSumInsured().equals(data.getSumInsured()) &&

o.getNcdyears().equals(data.getNcdYears()) && o.getNcdYn().equals(data.getNcdYn()) && o.getNoOfClaims().equals(data.getNoOfClaims())

&& o.getStatus().equals("Y") ).collect(Collectors.toList());

filterData3.addAll(filterData) ;

filterData3.addAll(filterData2) ;

}

if (filterData3.size() <= 0 ) {

newEntry = true ;

break ;

} else {

vdRefNo = String.valueOf(filterData.get(0).getVdRefno());

}

}

if(newEntry==true ) {

vdRefNo = sdf.format(new Date()) ;

for ( EserviceMotorDetails data : motorDatas) {

MsVehicleDetails saveVehicle = new MsVehicleDetails();

dozerMapper.map(data, saveVehicle);

saveVehicle.setVdRefno(Long.valueOf(vdRefNo) );

saveVehicle.setEntryDate(new Date());

saveVehicle.setAccidentYn(data.getAccident());

saveVehicle.setStatus("Y");

saveVehicle.setNcdyears(data.getNcdYears());

saveVehicle.setInsuranceClass(data.getInsuranceClass());

saveVehicle.setNcdYn(data.getNcdYn());

saveVehicle.setOwnerCategory(data.getOwnerCategory());

saveVehicle.setRegistrationYear(data.getRegistrationYear());

saveVehicle.setVehicleBodyType(data.getVehicleType());

saveVehicle.setWindScreenCoverRequeired(data.getWindScreenCoverRequired());

Integer grossWeight =Integer.valueOf(df.format(data.getGrossWeight()));

saveVehicle.setVehicleWeight(grossWeight);

msVehicleRepo.save(saveVehicle);

}

}

//

resList.add(vdRefNo);

resList.add(sectionId);

resList.add(agencyCode);

resList.add(branchCode);

resList.add(productId);

resList.add(companyId);

}catch (Exception e) {

e.printStackTrace();

log.error("Exception is ---> " + e.getMessage());

return null ;

}

return resList;

}

public MsDetailsRes msDetails (Long msrefno) {

MsDetailsRes res = new MsDetailsRes();

DozerBeanMapper mapper = new DozerBeanMapper();

try {

MsCommonDetails data = msCommonRepo.findByMsRefNo(msrefno);

MsCustomerDetails cusdetails = msCustomerRepo.findByCdRefno(Long.valueOf(data.getCdRefno()));

MsVehicleDetails msvehdetails = msVehicleRepo.findByVdRefno(Long.valueOf(data.getVdRefno()));

res.setMsCommonDetails(mapper.map(data,MsCommonDetailsRes.class));

res.setMsCustomerDetails(mapper.map(cusdetails,MsCustomerDetailsRes.class));

res.setMsVehicleDetails(mapper.map(msvehdetails, MsVehicleDetailsRes.class));

}

catch (Exception e) {

e.printStackTrace();

log.error("Exception is ---> " + e.getMessage());

return null ;

}

return res;

}

}