/\*

\* Copyright (c) 2019. All right reserved

\* Created on 2022-03-11 ( Date ISO 2022-03-11 - Time 16:18:37 )

\* Generated by Telosys Tools Generator ( version 3.3.0 )

\*/

package com.maan.crm.service.impl;

import java.text.DecimalFormat;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.Collections;

import java.util.Date;

import java.util.List;

import org.apache.commons.lang3.StringUtils;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

import org.modelmapper.ModelMapper;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.google.gson.Gson;

import com.maan.crm.bean.CrmStateMaster;

import com.maan.crm.error.Error;

import com.maan.crm.repository.CrmStateMasterRepository;

import com.maan.crm.req.CrmStateMasterSaveReq;

import com.maan.crm.req.StateDropDownReq;

import com.maan.crm.res.CommonCrmRes;

import com.maan.crm.res.CrmStateMasterRes;

import com.maan.crm.res.CrmStateSuccessRes;

import com.maan.crm.res.DropDownRes;

import com.maan.crm.res.SuccessRes;

import com.maan.crm.service.CrmStateMasterService;

/\*\*

\* <h2>CrmStateMasterServiceimpl</h2>

\*/

@Service

@Transactional

public class CrmStateMasterServiceImpl implements CrmStateMasterService {

@Autowired

private CrmStateMasterRepository repository;

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

Gson json = new Gson();

///Date Format

SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");

/////Decimal to String Convert/////

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

DecimalFormat decimalFormat = new DecimalFormat(pattern);

/\*

\* public CrmStateMasterServiceImpl(CrmStateMasterRepository repo) {

\* this.repository = repo; }

\*

\*/

@Override

public CrmStateMaster create(CrmStateMaster d) {

CrmStateMaster entity;

try {

entity = repository.save(d);

} catch (Exception ex) {

log.error(ex);

return null;

}

return entity;

}

@Override

public CrmStateMaster update(CrmStateMaster d) {

CrmStateMaster c;

try {

c = repository.saveAndFlush(d);

} catch (Exception ex) {

log.error(ex);

return null;

}

return c;

}

/\*

\* @Override public CrmStateMaster getOne(long id) { CrmStateMaster t;

\*

\* try { t = repository.findById(id).orElse(null);

\*

\* } catch (Exception ex) { log.error(ex); return null; } return t; }

\*

\*/

@Override

public List<CrmStateMaster> getAll() {

List<CrmStateMaster> lst;

try {

lst = repository.findAll();

} catch (Exception ex) {

log.error(ex);

return Collections.emptyList();

}

return lst;

}

@Override

public long getTotal() {

long total;

try {

total = repository.count();

} catch (Exception ex) {

log.error(ex);

return 0;

}

return total;

}

@Override

public CommonCrmRes getStateDropDown(StateDropDownReq req) {

CommonCrmRes commonRes = new CommonCrmRes();

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

try {

List<CrmStateMaster> lst = repository.OrderByStatenameAsc();

for (CrmStateMaster data : lst) {

DropDownRes res = new DropDownRes();

res.setCode(data.getStatecode().toString());

res.setCodedesc(data.getStatename());

resList.add(res);

}

// Response

commonRes.setCommonResponse(resList);

commonRes.setIsError(false);

commonRes.setMessage("Success");

} catch (Exception e) {

e.printStackTrace();

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

commonRes.setCommonResponse(null);

commonRes.setIsError(true);

commonRes.setMessage("Failed");

return commonRes;

}

return commonRes;

}

/\*

\* @Override public boolean delete(long id) { try { repository.deleteById(id);

\* return true;

\*

\* } catch (Exception ex) { log.error(ex); return false; } }

\*

\*/

////// Insert

@Override

public List<Error> validateCrmStateMaster(CrmStateMasterSaveReq req) {

// TODO Auto-generated method stub

List<Error> errors = new ArrayList<Error>();

//// validation

try {

if (StringUtils.isNotBlank(req.getStatecode())) {

if (!StringUtils.isNumeric(req.getStatecode())) {

errors.add(new Error("01", "State Code", "Please Enter Valid State Code "));

}

}

if (req.getStatename() == null || StringUtils.isBlank(req.getStatename())) {

errors.add(new Error("02", "State Name", "Please Enter State Name"));

} else if (req.getStatename().length() > 200) {

errors.add(new Error("02", "State Name", "Please Enter State Name within 200 Characters"));

}

if (req.getStatus() == null || StringUtils.isBlank(req.getStatus())) {

errors.add(new Error("03", "Status", "Please Enter Status"));

} else if (req.getStatus().length() > 1) {

errors.add(new Error("03", "Status", "Please Enter Status within 1 Character"));

} else if (!(req.getStatus().equals("Y") || req.getStatus().equals("N"))) {

errors.add(new Error("03", "Status", "Please Enter Status Properly Yes-Y or No-N"));

}

Calendar cal = Calendar.getInstance();

cal.add(Calendar.DATE, -1);

Date yesterday = cal.getTime();

SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");

Date a = sdf.parse(req.getEffectivedate());

if (req.getEffectivedate() == null || StringUtils.isBlank(req.getEffectivedate().toString())) {

errors.add(new Error("04", "Effective Date", "Please Enter Effective Date"));

} else if (a.before(yesterday)) {

errors.add(new Error("04", "EffectiveDate", "Please Enter Future Date as EffectiveDate"));

} else if (!req.getEffectivedate().matches("([0-9]{2})/([0-9]{2})/([0-9]{4})")) {

errors.add(new Error("04", "EffectiveDate",

"Effective Date format should be dd/MM/yyyy only allowed . Example :- 15/12/2020"));

}

Date endDate = sdf.parse(req.getEnddate());

Date effectiveDate = sdf.parse(req.getEffectivedate());

if (req.getEnddate() == null || StringUtils.isBlank(req.getEnddate().toString())) {

errors.add(new Error("05", "End Date", "Please Enter End Date"));

} else if (endDate.before(effectiveDate)) {

errors.add(new Error("05", "End Date", "End Date not before Effective Date"));

} else if (!req.getEnddate().matches("([0-9]{2})/([0-9]{2})/([0-9]{4})")) {

errors.add(new Error("05", "End Date",

"End Date format should be dd/MM/yyyy only allowed . Example :- 15/12/2020"));

}

/\*

\* if(req.getCoreappcode()== null || StringUtils.isBlank(req.getCoreappcode()))

\* { errors.add(new Error("06", "Core App Code", "Please Enter Core App Code"));

\* }

\*/

if (req.getCoreappcode().length() > 50) {

errors.add(new Error("06", "Core App Code", "Please Enter Core App Code within 50 Characters"));

}

/\*

\* Long coreCount =

\* repository.countByCoreappcodeOrderByEffectiveDateDesc(req.getCoreappcode());

\* if(coreCount>0) { errors.add(new Error("06", "Core App Code",

\* "Please Enter New Core App Code "+req.getCoreappcode()+" is exist already"));

\* }

\*/

if (StringUtils.isBlank(req.getStatecode()) && StringUtils.isNotBlank(req.getCoreappcode())) {

Long coreCount = repository.countByCoreappcode(req.getCoreappcode());

if (coreCount > 0)

errors.add(new Error("06", "Core App Code",

"Please Enter New Core App Code " + req.getCoreappcode() + " is exist already"));

} else if (StringUtils.isNotBlank(req.getStatecode()) && StringUtils.isNotBlank(req.getCoreappcode())) {

Long stateandCoreCount = repository

.countByStatecodeNotAndCoreappcode(Double.valueOf(req.getStatecode()), req.getCoreappcode());

if (stateandCoreCount > 0)

errors.add(new Error("06", "Core App Code",

"Please Enter New Core App Code " + req.getCoreappcode() + " is exist already"));

}

/\*

\* if(req.getRemarks()== null || StringUtils.isBlank(req.getRemarks())) {

\* errors.add(new Error("07", "Remarks", "Please Enter Remarks")); }

\*/

if (req.getRemarks().length() > 200) {

errors.add(new Error("07", "Remarks", "Please Enter Remarks within 200 Characters"));

}

} catch (Exception e) {

e.printStackTrace();

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

errors.add(new Error("08", " Common", e.getMessage()));

return errors;

}

return errors;

}

///////////Save

@Override

@Transactional

public CrmStateSuccessRes saveCrmStateMaster(CrmStateMasterSaveReq req) {

CrmStateSuccessRes res = new CrmStateSuccessRes();

CrmStateMaster entity = new CrmStateMaster();

SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");

ModelMapper mapper = new ModelMapper();

Date entryDate = null;

Double stateCode = 0D;

try {

if (StringUtils.isBlank(req.getStatecode())) {

CrmStateMaster crmstatemaster = repository.findTop1ByOrderByStatecodeDesc();

stateCode = Double.valueOf(crmstatemaster.getStatecode() + 1);

entryDate = new Date();

mapper.map(req,CrmStateSuccessRes.class);

res.setResponse("Saved Successfully " );

res.setStatecode(" Your State Code Number is = " + decimalFormat.format(stateCode));

} else {

// Update

stateCode = Double.valueOf(req.getStatecode());

CrmStateMaster data = repository.findTop1ByStatecodeOrderByEffectiveDateDesc(stateCode);

entryDate = data.getEntryDate();

res.setResponse("Updated Successfully");

res.setStatecode(" Your State Code Number is = " + decimalFormat.format(stateCode));

}

// Primary

entity.setStatecode(Double.valueOf(stateCode));

entity.setCoreappcode(req.getCoreappcode());

entity.setEffectiveDate(sdf.parse(req.getEffectivedate()));

entity.setStatename(req.getStatename());

entity.setEndDate(sdf.parse(req.getEnddate()));

entity.setRemarks(req.getRemarks());

entity.setStatus(req.getStatus());

entity.setEntryDate(entryDate);

repository.save(entity);

log.info("Saved Details is ---> " + json.toJson(entity));

} catch (Exception ex) {

log.error(ex);

return null;

}

return res;

}

//////// Get All

@Override

public List<CrmStateMasterRes> getAllCrmStateMaster() {

{

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

// ModelMapper mapper = new ModelMapper();

try {

List<CrmStateMaster> crmstatemaster = repository.OrderByStatecodeDesc();

for (CrmStateMaster data : crmstatemaster) {

CrmStateMasterRes res = new CrmStateMasterRes();

res.setCoreappcode(data.getCoreappcode());

res.setEffectivedate(sdf.format(data.getEffectiveDate()));

res.setEnddate(sdf.format(data.getEndDate()));

res.setEntrydate(sdf.format(data.getEntryDate()));

res.setRemarks(data.getRemarks());

res.setStatename(data.getStatename());

res.setStatus(data.getStatus());

res.setStatecode(decimalFormat.format(data.getStatecode()));

resList.add(res);

}

} catch (Exception e) {

e.printStackTrace();

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

return null;

}

return resList;

}

}

/////////// Get By State Code

@Override

public CrmStateMasterRes getCrmStateMaster(String statecode) {

CrmStateMasterRes res = new CrmStateMasterRes();

ModelMapper mapper = new ModelMapper();

try {

// Map

CrmStateMaster data = repository.findByStatecodeOrderByStatecodeDesc(Double.valueOf(statecode));

res = mapper.map(data, CrmStateMasterRes.class);

res.setEffectivedate(sdf.format(data.getEffectiveDate()));

res.setEnddate(sdf.format(data.getEffectiveDate()));

res.setEntrydate(sdf.format(data.getEntryDate()));

res.setStatecode(decimalFormat.format(data.getStatecode()));

} catch (Exception e) {

e.printStackTrace();

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

return null;

}

return res;

}

////// Get By Active States

@Override

public List<CrmStateMasterRes> getCrmStateMasteractive() {

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

ModelMapper mapper = new ModelMapper();

try {

// Map

List<CrmStateMaster> crmstatemaster = repository.findByStatusOrderByStatecodeDesc("Y");

for (CrmStateMaster data : crmstatemaster) {

CrmStateMasterRes res = new CrmStateMasterRes();

res = mapper.map(data, CrmStateMasterRes.class);

res.setEffectivedate(sdf.format(data.getEffectiveDate()));

res.setEnddate(sdf.format(data.getEffectiveDate()));

res.setEntrydate(sdf.format(data.getEntryDate()));

res.setStatecode(decimalFormat.format(data.getStatecode()));

resList.add(res);

}

} catch (Exception e) {

e.printStackTrace();

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

return null;

}

return resList;

}

}