@Service

@Transactional

public class MailMasterServiceImpl implements MailMasterService {

@PersistenceContext

private EntityManager em;

@Autowired

private MailMasterRepository mailRepo;

Gson json = new Gson();

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

public Long getMasterTableCount() {

Long data = 0L;

try {

List<Long> list = new ArrayList<Long>();

// Find Latest Record

CriteriaBuilder cb = em.getCriteriaBuilder();

CriteriaQuery<Long> query = cb.createQuery(Long.class);

// Find All

Root<MailMaster> b = query.from(MailMaster.class);

// Select

query.multiselect(cb.count(b));

// Effective Date Max Filter

Subquery<Long> effectiveDate = query.subquery(Long.class);

Root<MailMaster> ocpm1 = effectiveDate.from(MailMaster.class);

effectiveDate.select(cb.max(ocpm1.get("effectiveDateStart")));

Predicate a1 = cb.equal(ocpm1.get("companyId"), b.get("companyId"));

effectiveDate.where(a1);

Predicate n1 = cb.equal(b.get("effectiveDateStart"), effectiveDate);

query.where(n1);

// Get Result

TypedQuery<Long> result = em.createQuery(query);

list = result.getResultList();

data = list.get(0);

} catch (Exception e) {

e.printStackTrace();

log.info(e.getMessage());

}

return data;

}

// Company Id Exist Details validation

public List<MailMaster> getSnoDetails(String companyId) {

List<MailMaster> list = new ArrayList<MailMaster>();

try {

// Find Latest Record

CriteriaBuilder cb = em.getCriteriaBuilder();

CriteriaQuery<MailMaster> query = cb.createQuery(MailMaster.class);

// Find All

Root<MailMaster> b = query.from(MailMaster.class);

// Select

query.select(b);

// Effective Date Max Filter

Subquery<Long> effectiveDate = query.subquery(Long.class);

Root<MailMaster> ocpm1 = effectiveDate.from(MailMaster.class);

effectiveDate.select(cb.max(ocpm1.get("effectiveDateStart")));

Predicate a1 = cb.equal(ocpm1.get("companyId"), b.get("companyId"));

effectiveDate.where(a1);

Predicate n1 = cb.equal(b.get("effectiveDateStart"), effectiveDate);

Predicate n2 = cb.equal(b.get("companyId"), companyId);

query.where(n1, n2);

// Get Result

TypedQuery<MailMaster> result = em.createQuery(query);

list = result.getResultList();

} catch (Exception e) {

e.printStackTrace();

log.info(e.getMessage());

}

return list;

}

@Override

public List<Error> validatemailmaster(MailMasterSaveReq req) {

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

try {

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

errorList.add(new Error("01", "Company Id", "Please Enter Company Id"));

} else if (req.getCompanyId().length() > 20) {

errorList.add(new Error("01", "Company Id", "Please Enter Company Id within 20 Characters"));

} else if (StringUtils.isBlank(req.getSNo())) {

List<MailMaster> mailList = getSnoDetails(req.getCompanyId());

if (mailList.size() > 0) {

errorList.add(new Error("01", "S No", "Please Enter Your Sno"));

}

} else {

List<MailMaster> mailList = getCompanyIdExistDetails(req.getCompanyId());

if (mailList.size() > 0 && (!req.getSNo().equalsIgnoreCase(mailList.get(0).getSNo().toString()))) {

errorList.add(new Error("01", "Company Id", "This Company Id Already Exist "));

}

}

// Date Validation

Calendar cal = new GregorianCalendar();

Date today = new Date();

cal.setTime(today);

cal.add(Calendar.DAY\_OF\_MONTH, -1);

cal.set(Calendar.HOUR\_OF\_DAY, 23);

cal.set(Calendar.MINUTE, 50);

today = cal.getTime();

if (req.getEffectiveDateStart() == null) {

errorList.add(new Error("02", "EffectiveDateStart", "Please Enter Effective Date Start "));

} else if (req.getEffectiveDateStart().before(today)) {

errorList

.add(new Error("02", "EffectiveDateStart", "Please Enter Effective Date Start as Future Date"));

} else if (req.getEffectiveDateEnd() == null) {

errorList.add(new Error("03", "EffectiveDateEnd", "Please Enter Effective Date End "));

} else if (req.getEffectiveDateEnd().before(req.getEffectiveDateStart())

|| req.getEffectiveDateEnd().equals(req.getEffectiveDateStart())) {

errorList.add(new Error("03", "EffectiveDateEnd",

"Please Enter Effective Date End is After Effective Date Start"));

}

// Status Validation

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

errorList.add(new Error("04", "Status", "Please Enter Status"));

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

errorList.add(new Error("04", "Status", "Status 1 Character Only"));

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

errorList.add(new Error("04", "Status", "Enter Status Y or N Only"));

}

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

errorList.add(new Error("05", "CreatedBy", "Please Enter CreatedBy"));

} else if (req.getCreatedBy().length() > 100) {

errorList.add(new Error("05", "CreatedBy", "Please Enter CreatedBy within 100 Characters"));

}

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

errorList.add(new Error("06", "Remarks", "Please Enter Remarks"));

} else if (req.getRemarks().length() > 300) {

errorList.add(new Error("06", "Remarks", "Please Enter Remarks within 100 Characters"));

}

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

errorList.add(new Error("07", "Smtp Host", "Please Enter Smtp Host"));

} else if (req.getSmtpHost().length() > 300) {

errorList.add(new Error("07", "Smtp Host", "Please Enter Smtp Host within 300 Characters"));

}

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

errorList.add(new Error("08", "Smtp User", "Please Enter Smtp User"));

} else if (req.getSmtpUser().length() > 300) {

errorList.add(new Error("08", "Smtp User", "Please Enter Smtp User within 300 Characters"));

}

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

errorList.add(new Error("09", "Smtp Pwd", "Please Enter Smtp Pwd"));

} else if (req.getSmtpPwd().length() > 300) {

errorList.add(new Error("09", "Smtp Pwd", "Please Enter Smtp Pwd within 300 Characters"));

}

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

errorList.add(new Error("10", "Smtp Port", "Please Enter Smtp Port"));

}

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

errorList.add(new Error("11", "Address", "Please Enter Address"));

} else if (req.getAddress().length() > 300) {

errorList.add(new Error("11", "Address", "Please Enter Address within 300 Characters"));

}

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

errorList.add(new Error("12", "Core App Code", "Please Enter Core App Code"));

} else if (req.getCoreAppCode().length() > 20) {

errorList.add(new Error("12", "Core App Code", "Please Enter CreatedBy within 20 Characters"));

}

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

errorList.add(new Error("13", "Tira Code", "Please Enter Tira Code"));

} else if (req.getTiraCode().length() > 20) {

errorList.add(new Error("13", "Tira Code", "Please Enter Tira Code within 20 Characters"));

}

} catch (Exception e) {

log.error(e);

e.printStackTrace();

}

return errorList;

}

// Company Id Exist Details validation

private List<MailMaster> getCompanyIdExistDetails(String companyId) {

List<MailMaster> list = new ArrayList<MailMaster>();

try {

// Find Latest Record

CriteriaBuilder cb = em.getCriteriaBuilder();

CriteriaQuery<MailMaster> query = cb.createQuery(MailMaster.class);

// Find All

Root<MailMaster> b = query.from(MailMaster.class);

// Select

query.select(b);

// Effective Date Max Filter

Subquery<Long> effectiveDate = query.subquery(Long.class);

Root<MailMaster> ocpm1 = effectiveDate.from(MailMaster.class);

effectiveDate.select(cb.max(ocpm1.get("effectiveDateStart")));

Predicate a1 = cb.equal(ocpm1.get("sNo"), b.get("sNo"));

Predicate a2 = cb.equal(ocpm1.get("companyId"), b.get("companyId"));

effectiveDate.where(a1, a2);

Predicate n1 = cb.equal(b.get("effectiveDateStart"), effectiveDate);

Predicate n2 = cb.equal(b.get("companyId"), companyId);

query.where(n1, n2);

// Get Result

TypedQuery<MailMaster> result = em.createQuery(query);

list = result.getResultList();

} catch (Exception e) {

e.printStackTrace();

log.info(e.getMessage());

}

return list;

}

@Override

public SuccessRes insertmailmaster(MailMasterSaveReq req) {

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

SuccessRes res = new SuccessRes();

MailMaster saveData = new MailMaster();

List<MailMaster> list = new ArrayList<MailMaster>();

DozerBeanMapper dozermapper = new DozerBeanMapper();

try {

Integer amendId = 0;

Calendar cal = new GregorianCalendar();

cal.setTime(req.getEffectiveDateStart());

cal.set(Calendar.HOUR\_OF\_DAY, 23);

cal.set(Calendar.MINUTE, 59);

Date startDate = cal.getTime();

Date today = new Date();

cal.setTime(req.getEffectiveDateStart());

cal.add(Calendar.DAY\_OF\_MONTH, -1);

cal.set(Calendar.HOUR\_OF\_DAY, today.getHours());

cal.set(Calendar.MINUTE, today.getMinutes());

Date oldEndDate = cal.getTime();

cal.setTime(req.getEffectiveDateStart());

cal.set(Calendar.HOUR\_OF\_DAY, today.getHours());

cal.set(Calendar.MINUTE, today.getMinutes());

Date effDate = cal.getTime();

Date endDate = req.getEffectiveDateEnd();

String sno = "";

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

// Save

Long totalCount = getMasterTableCount();

sno = Long.valueOf(totalCount + 1).toString();

res.setResponse("Saved Successfully");

res.setSuccessId(sno);

} else {

// Update

sno = req.getSNo();

CriteriaBuilder cb = em.getCriteriaBuilder();

CriteriaQuery<MailMaster> query = cb.createQuery(MailMaster.class);

// Find all

Root<MailMaster> b = query.from(MailMaster.class);

// Select

query.select(b);

// Effective Date Max Filter

Subquery<Long> effectiveDate = query.subquery(Long.class);

Root<MailMaster> ocpm1 = effectiveDate.from(MailMaster.class);

effectiveDate.select(cb.max(ocpm1.get("effectiveDateStart")));

Predicate a1 = cb.lessThanOrEqualTo(ocpm1.get("effectiveDateStart"), startDate);

effectiveDate.where(a1);

// Where

Predicate n1 = cb.equal(b.get("status"), "Y");

Predicate n2 = cb.equal(b.get("effectiveDateStart"), effectiveDate);

Predicate n3 = cb.equal(b.get("sNo"), req.getSNo());

query.where(n1, n2, n3);

// Get Result

TypedQuery<MailMaster> result = em.createQuery(query);

list = result.getResultList();

if (list.size() > 0) {

mailRepo.delete(list.get(0));

// Amend Id

if (list.get(0).getEffectiveDateStart().before(startDate)) {

String startDatewithoutTime = sdformat.format(startDate);

String oldDateWithoutTime = sdformat.format(list.get(0).getEffectiveDateStart());

if (startDatewithoutTime.equalsIgnoreCase(oldDateWithoutTime)) {

amendId = list.get(0).getAmendId() + 1;

}

}

}

res.setResponse("Updated Successfully");

res.setSuccessId(sno);

}

dozermapper.map(req, saveData);

saveData.setSNo(Integer.valueOf(sno));

saveData.setEffectiveDateStart(effDate);

saveData.setEffectiveDateEnd(endDate);

saveData.setEntryDate(new Date());

saveData.setAmendId(amendId);

mailRepo.saveAndFlush(saveData);

if (list.size() > 0) {

// Update Old Record

MailMaster lastRecord = list.get(0);

lastRecord.setEffectiveDateEnd(oldEndDate);

mailRepo.saveAndFlush(lastRecord);

}

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

} catch (Exception e) {

e.printStackTrace();

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

return null;

}

return res;

}

@Override

public MailMasterGetRes getById(MailMasterGetReq req) {

MailMasterGetRes res = new MailMasterGetRes();

DozerBeanMapper mapper = new DozerBeanMapper();

try {

Date today = new Date();

Calendar cal = new GregorianCalendar();

cal.setTime(today);

cal.set(Calendar.HOUR\_OF\_DAY, 23);

cal.set(Calendar.MINUTE, 1);

today = cal.getTime();

List<MailMaster> list = new ArrayList<MailMaster>();

// Find Latest Record

CriteriaBuilder cb = em.getCriteriaBuilder();

CriteriaQuery<MailMaster> query = cb.createQuery(MailMaster.class);

// Find all

Root<MailMaster> b = query.from(MailMaster.class);

// Select

query.select(b);

// Effective Date Max Filter

Subquery<Long> effectiveDate = query.subquery(Long.class);

Root<MailMaster> ocpm1 = effectiveDate.from(MailMaster.class);

effectiveDate.select(cb.max(ocpm1.get("effectiveDateStart")));

Predicate a1 = cb.lessThanOrEqualTo(ocpm1.get("effectiveDateStart"), today);

effectiveDate.where(a1);

// Order By

List<Order> orderList = new ArrayList<Order>();

orderList.add(cb.asc(b.get("sNo")));

Predicate n1 = cb.equal(b.get("companyId"), req.getCompanyId());

Predicate n2 = cb.equal(b.get("sNo"), req.getSNo());

query.where(n1, n2).orderBy(orderList);

// Get Result

TypedQuery<MailMaster> result = em.createQuery(query);

list = result.getResultList();

// Map

for (MailMaster data : list) {

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

res.setSNo(data.getSNo().toString());

}

} catch (Exception e) {

e.printStackTrace();

log.info(e.getMessage());

return null;

}

return res;

}

}