@Service

@Transactional

public class MotorColorMasterServiceImpl implements MotorColorMasterService {

@Autowired

private MotorColorMasterRepository repo;

@PersistenceContext

private EntityManager em;

Gson json = new Gson();

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

@Override

public List<Error> validateColorMotor(MotorColorSaveReq req) {

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

try {

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

errorList.add(new Error("01", "Color Code", "Please Enter Color Code "));

}

else if (req.getColorCode().length()>100) {

errorList.add(new Error("01", "Color Code", "Please Enter Color Code within 100 Characters "));

}

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

errorList.add(new Error("02", "Color Desc", "Please Enter Color Desc "));

}

else if (req.getColorDesc().length()>100) {

errorList.add(new Error("02", "Color Desc", "Please Enter Color Desc within 100 Characters "));

}

// 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 || StringUtils.isBlank(req.getEffectiveDateStart().toString())) {

errorList.add(new Error("03", "EffectiveDateStart", "Please Enter Effective Date Start"));

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

errorList

.add(new Error("03", "EffectiveDateStart", "Please Enter Effective Date Start as Future Date"));

}

// Status Validation

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"));

}

} catch (Exception e) {

log.error(e);

e.printStackTrace();

}

return errorList;

}

@Override

public SuccessRes saveColor(MotorColorSaveReq req) {

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

SuccessRes res = new SuccessRes();

MotorColorMaster saveData = new MotorColorMaster();

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

DozerBeanMapper dozerMapper = new DozerBeanMapper();

try {

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 = sdformat.parse("12/12/2050");

String colorId = "";

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

// Save

// Long totalCount = repo.count();

Long totalCount = getMasterTableCount();

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

res.setResponse("Saved Successfully ");

res.setSuccessId(colorId);

} else {

// Update

// Get Less than Equal Today Record

// Criteria

colorId = req.getColorId().toString();

CriteriaBuilder cb = em.getCriteriaBuilder();

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

// Find All

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

// Select

query.select(b);

// Effective Date Max Filter

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

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

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

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

Predicate a2 = cb.lessThanOrEqualTo(ocpm1.get("effectiveDateStart"), startDate);

effectiveDate.where(a1, a2);

// Where

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

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

Predicate n3 = cb.equal(b.get("colorId"), req.getColorId());

query.where(n1, n2, n3);

// Get Result

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

list = result.getResultList();

if (list.size() > 0) {

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

}

res.setResponse("Updated Successfully ");

res.setSuccessId(colorId);

}

dozerMapper.map(req, saveData);

saveData.setColorId(Integer.valueOf(colorId));

saveData.setColorDesc(req.getColorDesc());

saveData.setEffectiveDateStart(req.getEffectiveDateStart());

saveData.setEffectiveDateEnd(endDate);

saveData.setStatus("Y");

saveData.setEntryDate(new Date());

saveData.setAmendId(0);

repo.saveAndFlush(saveData);

if (list.size() > 0) {

// Update Old Record

MotorColorMaster lastRecord = list.get(0);

lastRecord.setEffectiveDateEnd(oldEndDate);

repo.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;

}

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<MotorColorMaster> b = query.from(MotorColorMaster.class);

// Select

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

// Effective Date Max Filter

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

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

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

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

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;

}

@Override

public MotorColorGetRes getMotorColor(MotorColorGetReq req) {

MotorColorGetRes res = new MotorColorGetRes();

ModelMapper mapper = new ModelMapper();

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

try {

// Criteria

CriteriaBuilder cb = em.getCriteriaBuilder();

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

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

// Find All

Root<MotorColorMaster> c = query.from(MotorColorMaster.class);

// Select

query.select(c);

// Effective Date Max Filter

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

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

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

javax.persistence.criteria.Predicate a1 = cb.equal(c.get("colorId"), ocpm1.get("colorId"));

effectiveDate.where(a1);

// Order By

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

orderList.add(cb.asc(c.get("effectiveDateStart")));

// Where

javax.persistence.criteria.Predicate n1 = cb.equal(c.get("effectiveDateStart"), effectiveDate);

javax.persistence.criteria.Predicate n2 = cb.equal(c.get("colorId"), req.getColorId());

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

// Get Result

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

list = result.getResultList();

res = mapper.map(list.get(0), MotorColorGetRes.class);

res.setColorId(list.get(0).getColorId());

res.setEntryDate(list.get(0).getEntryDate());

res.setEffectiveDateStart(list.get(0).getEffectiveDateStart());

res.setEffectiveDateEnd(list.get(0).getEffectiveDateEnd());

res.setRemarks(list.get(0).getRemarks());

} catch (Exception e) {

e.printStackTrace();

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

return null;

}

return res;

}

@Override

public List<MotorColorGetRes> getallMotorColor(MotorColorGetAllReq req) {

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

DozerBeanMapper mapper = new DozerBeanMapper();

try {

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

// Pagination

int limit = StringUtils.isBlank(req.getLimit()) ? 0 : Integer.valueOf(req.getLimit());

int offset = StringUtils.isBlank(req.getOffset()) ? 100 : Integer.valueOf(req.getOffset());

// Find Latest Record

CriteriaBuilder cb = em.getCriteriaBuilder();

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

// Find All

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

// Select

query.select(b);

// Effective Date Max Filter

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

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

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

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

effectiveDate.where(a1);

// Order By

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

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

// Where

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

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

// Get Result

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

result.setFirstResult(limit \* offset);

result.setMaxResults(offset);

list = result.getResultList();

// Map

for (MotorColorMaster data : list) {

MotorColorGetRes res = new MotorColorGetRes();

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

res.setColorId(data.getColorId());

resList.add(res);

}

} catch (Exception e) {

e.printStackTrace();

log.info(e.getMessage());

return null;

}

return resList;

}

@Override

public List<MotorColorGetRes> getactiveMotorColor(MotorColorGetAllReq req) {

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

DozerBeanMapper mapper = new DozerBeanMapper();

try {

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

// Pagination

int limit = StringUtils.isBlank(req.getLimit()) ? 0 : Integer.valueOf(req.getLimit());

int offset = StringUtils.isBlank(req.getOffset()) ? 100 : Integer.valueOf(req.getOffset());

// Find Latest Record

CriteriaBuilder cb = em.getCriteriaBuilder();

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

// Find All

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

// Select

query.select(b);

// Effective Date Max Filter

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

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

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

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

effectiveDate.where(a1);

// Order By

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

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

// Where

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

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

// Get Result

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

result.setFirstResult(limit \* offset);

result.setMaxResults(offset);

list = result.getResultList();

// Map

for (MotorColorMaster data : list) {

MotorColorGetRes res = new MotorColorGetRes();

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

res.setColorId(data.getColorId());

resList.add(res);

}

} catch (Exception e) {

e.printStackTrace();

log.info(e.getMessage());

return null;

}

return resList;

}

}