@Service

@Transactional

public class FactorTypeMasterServiceImpl implements FactorTypeMasterService {

@PersistenceContext

private EntityManager em;

@Autowired

private FactorTypeMasterRepository factorRepo;

Gson json = new Gson();

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

@Override

public List<Error> validateFactorType(FactorTypeMasterSaveReq req) {

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

try {

// 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("01", "EffectiveDateStart", "Please Enter Effective Date Start "));

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

errorList

.add(new Error("01", "EffectiveDateStart", "Please Enter Effective Date Start as Future Date"));

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

errorList.add(new Error("02", "EffectiveDateEnd", "Please Enter Effective Date End "));

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

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

errorList.add(new Error("02", "EffectiveDateEnd",

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

}

// Status Validation

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

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

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

errorList.add(new Error("03", "Status", "Status 1 Character Only"));

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

errorList.add(new Error("03", "Status", "Enter Status Y or N Only"));

}

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

errorList.add(new Error("04", "Factor Name", "Please Select Factor Name "));

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

errorList.add(new Error("04","Factor Name", "Please Enter Factor Name within 100 Characters"));

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

List<FactorTypeMaster> factorList = getFactorNameExistDetails(req.getFactorName(),req.getProductId());

if (factorList.size()>0 ) {

errorList.add(new Error("04", "Factor Name", "This Factor Name Already Exist "));

}

}else {

List<FactorTypeMaster> factorList = getFactorNameExistDetails(req.getFactorName(),req.getProductId() );

if (factorList.size()>0 && (! req.getFactorId().equalsIgnoreCase(factorList.get(0).getFactorId().toString())) ) {

errorList.add(new Error("04", "Factor Name", "This Factor Name Already Exist "));

}

}

if (req.getFactorName().length() > 100) {

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

}

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

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

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

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

}

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

errorList.add(new Error("07", "Input Column", "Please Enter Input Column"));

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

errorList.add(new Error("07", "Input Column", "Please Enter Input Column within 100 Characters"));

}

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

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

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

errorList.add(new Error("08", "CreatedBy", "Please Enter CreatedBy within 100 Characters"));

}

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

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

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

errorList.add(new Error("09", "Remarks", "Please Enter Remarks within 100 Characters"));

}

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

errorList.add(new Error("10", "Product Id", "Please Enter Product Id"));

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

errorList.add(new Error("10", "Product Id", "Please Enter Product Id within 100 Characters"));

}

} catch (Exception e) {

log.error(e);

e.printStackTrace();

}

return errorList;

}

@Override

public SuccessRes insertfactortype(FactorTypeMasterSaveReq req) {

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

SuccessRes res = new SuccessRes();

FactorTypeMaster saveData = new FactorTypeMaster();

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

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 factorId = "";

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

// Save

Long totalCount = getMasterTableCount(req.getProductId());

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

res.setResponse("Saved Successfully");

res.setSuccessId(factorId);

} else {

// Update

factorId = req.getFactorId();

CriteriaBuilder cb = em.getCriteriaBuilder();

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

// Find all

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

// Select

query.select(b);

// Effective Date Max Filter

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

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

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

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

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

Predicate a3 = cb.equal(ocpm1.get("productId"), b.get("productId"));

effectiveDate.where(a1, a2, a3);

// Where

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

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

Predicate n3 = cb.equal(b.get("factorId"), req.getFactorId());

Predicate n4 = cb.equal(b.get("productId"), req.getProductId());

query.where(n1, n2, n3, n4);

// Get Result

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

list = result.getResultList();

if (list.size() > 0) {

factorRepo.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(factorId);

}

dozermapper.map(req, saveData);

saveData.setFactorId(Integer.valueOf(factorId));

saveData.setEffectiveDateStart(effDate);

saveData.setEffectiveDateEnd(endDate);

saveData.setEntryDate(new Date());

saveData.setAmendId(amendId);

factorRepo.saveAndFlush(saveData);

if (list.size() > 0) {

// Update Old Record

FactorTypeMaster lastRecord = list.get(0);

lastRecord.setEffectiveDateEnd(oldEndDate);

factorRepo.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(String productId) {

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

// Select

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

// Effective Date Max Filter

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

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

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

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

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

effectiveDate.where(a1, a2);

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

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

query.where(n1, n2);

// 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;

}

// Factor Name Exist Details validation

public List<FactorTypeMaster> getFactorNameExistDetails(String factorName, String productId) {

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

try {

// Find Latest Record

CriteriaBuilder cb = em.getCriteriaBuilder();

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

// Find All

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

// Select

query.select(b);

// Effective Date Max Filter

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

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

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

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

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

effectiveDate.where(a1,a2);

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

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

Predicate n3 = cb.equal(b.get("productId"),productId);

query.where(n1, n2,n3);

// Get Result

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

list = result.getResultList();

} catch (Exception e) {

e.printStackTrace();

log.info(e.getMessage());

}

return list;

}

@Override

public List<FactorTypeMasterGetRes> getallFactorType(FactorTypeMasterGetAllReq req) {

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

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<FactorTypeMaster> list = new ArrayList<FactorTypeMaster>();

// 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<FactorTypeMaster> query = cb.createQuery(FactorTypeMaster.class);

// Find all

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

// Select

query.select(b);

// Effective Date Max Filter

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

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

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

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

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

effectiveDate.where(a1, a2);

// Order By

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

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

// Where

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

Predicate n2 = cb.equal(b.get("productId"), req.getProductId());

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

// Get Result

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

result.setFirstResult(limit \* offset);

result.setMaxResults(offset);

list = result.getResultList();

// Map

for (FactorTypeMaster data : list) {

FactorTypeMasterGetRes res = new FactorTypeMasterGetRes();

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

res.setFactorId(data.getFactorId().toString());

resList.add(res);

}

} catch (Exception e) {

e.printStackTrace();

log.info(e.getMessage());

return null;

}

return resList;

}

@Override

public List<FactorTypeMasterGetRes> getActiveFactorType(FactorTypeMasterGetAllReq req) {

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

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<FactorTypeMaster> list = new ArrayList<FactorTypeMaster>();

// 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<FactorTypeMaster> query = cb.createQuery(FactorTypeMaster.class);

// Find all

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

// Select

query.select(b);

// Effective Date Max Filter

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

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

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

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

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

effectiveDate.where(a1, a2);

// Order By

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

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

// Where

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

Predicate n2 = cb.equal(b.get("productId"), req.getProductId());

Predicate n3 = cb.equal(b.get("status"), "Y");

query.where(n2, n3).orderBy(orderList);

// Get Result

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

result.setFirstResult(limit \* offset);

result.setMaxResults(offset);

list = result.getResultList();

// Map

for (FactorTypeMaster data : list) {

FactorTypeMasterGetRes res = new FactorTypeMasterGetRes();

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

res.setFactorId(data.getFactorId().toString());

resList.add(res);

}

} catch (Exception e) {

e.printStackTrace();

log.info(e.getMessage());

return null;

}

return resList;

}

@Override

public FactorTypeMasterGetRes getByFactorId(FactorTypeMasterGetReq req) {

FactorTypeMasterGetRes res = new FactorTypeMasterGetRes();

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<FactorTypeMaster> list = new ArrayList<FactorTypeMaster>();

CriteriaBuilder cb = em.getCriteriaBuilder();

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

// Find all

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

// Select

query.select(b);

// Effective Date Max Filter

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

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

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

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

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

Predicate a3 = cb.lessThanOrEqualTo(ocpm1.get("effectiveDateStart"), today);

effectiveDate.where(a1, a2,a3);

// Order By

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

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

// Where

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

javax.persistence.criteria.Predicate n2 = cb.equal(b.get("productId"), req.getProductId());

Predicate n3 = cb.equal(b.get("factorId"), req.getFactorId());

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

// Get Result

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

list = result.getResultList();

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

res.setFactorId(list.get(0).getFactorId().toString());

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

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

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

} catch (Exception e) {

e.printStackTrace();

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

return null;

}

return res;

}

@Override

public SuccessRes changeStatusOfFactorType(FactorTypeMasterChangeStatusReq req) {

SuccessRes res = new SuccessRes();

try {

Date today = new Date();

Calendar cal = new GregorianCalendar();

FactorTypeMaster updateRecord = new FactorTypeMaster();

cal.setTime(today);

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

cal.set(Calendar.MINUTE, 1);

today = cal.getTime();

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

// Find Latest Record

CriteriaBuilder cb = em.getCriteriaBuilder();

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

// Find All

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

// Select

query.select(b);

// Effective Date Max Filter

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

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

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

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

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

Predicate a3 = cb.equal(ocpm1.get("factorId"), b.get("factorId"));

effectiveDate.where(a1, a2,a3);

// Order By

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

orderList.add(cb.desc(b.get("effectiveDateStart")));

// Where

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

Predicate n2 = cb.equal(b.get("factorId"), req.getFactorId());

Predicate n3 = cb.equal(b.get("productId"),req.getProductId());

query.where(n1, n2,n3).orderBy(orderList);

// Get Result

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

list = result.getResultList();

updateRecord = list.get(0);

if (req.getStatus().equalsIgnoreCase("N")) {

// Delete Old Records

cal.setTime(today);

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

cal.set(Calendar.MINUTE, 30);

today = cal.getTime();

// create update

CriteriaDelete<FactorTypeMaster> delete = cb.createCriteriaDelete(FactorTypeMaster.class);

Root<FactorTypeMaster> pm = delete.from(FactorTypeMaster.class);

// Where

javax.persistence.criteria.Predicate n4 = cb.equal(pm.get("factorId"), req.getFactorId());

javax.persistence.criteria.Predicate n5 = cb.greaterThanOrEqualTo(pm.get("effectiveDateStart"), today);

javax.persistence.criteria.Predicate n6 = cb.equal(pm.get("productId"),req.getProductId());

delete.where(n4,n5,n6);

em.createQuery(delete).executeUpdate();

// Insert Updated Record

updateRecord.setStatus(req.getStatus());

factorRepo.save(updateRecord);

} else if (req.getStatus().equalsIgnoreCase("Y")) {

// Insert Updated Record

updateRecord.setStatus(req.getStatus());

factorRepo.save(updateRecord);

}

// perform update

res.setResponse("Status Changed");

res.setSuccessId(req.getFactorId());

} catch (Exception e) {

e.printStackTrace();

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

return null;

}

return res;

}

}