@Service

@Transactional

public class DocumentMasterServiceImpl implements DocumentMasterService {

@PersistenceContext

private EntityManager em;

Gson json = new Gson();

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

@Autowired

private DocumentMasterRepository repo;

@Transactional

@Override

public SuccessRes insertDocument(DocumentMasterSaveReq req) {

SuccessRes res = new SuccessRes();

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

DozerBeanMapper mapper = new DozerBeanMapper();

DocumentMaster saveData = new DocumentMaster();

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

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());

cal.set(Calendar.SECOND, today.getSeconds());

Date oldEndDate = cal.getTime() ;

cal.setTime(req.getEffectiveDateStart()); cal.set(Calendar.HOUR\_OF\_DAY, today.getHours()); cal.set(Calendar.MINUTE, today.getMinutes()) ;

cal.set(Calendar.SECOND, today.getSeconds());

Date effDate = cal.getTime();

Date endDate = sdf.parse("12/12/2050");

String documentId = "";

if(StringUtils.isBlank(req.getDocumentId().toString())) {

Long totalcount = getmastertablecount(req.getCompanyId());

documentId =Long.valueOf(totalcount+1).toString();

saveData.setDocumentId(Integer.valueOf(documentId));

res.setSuccessId(documentId);

res.setResponse("Saved Successful");

}

else {

documentId = req.getDocumentId().toString();

CriteriaBuilder cb = em.getCriteriaBuilder();

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

//Find All

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

//Select

query.select(b);

//Effective Date Max Filter

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

Root<DocumentMaster> ocpm1 = effectiveDate.from(DocumentMaster.class);effectiveDate.select(cb.max(ocpm1.get("effectiveDateStart")));

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

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

effectiveDate.where(a1,a2);

// Order By

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

// orderList.add(cb.asc(b.get("branchName")));

// Where

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

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

Predicate n3 = cb.equal(b.get("documentId"), req.getDocumentId() );

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

// Get Result

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

list = result.getResultList();

if( list.size() > 0) {

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

}

res.setResponse("Updated Successfully ");

res.setSuccessId(documentId);

}

mapper.map(req, saveData );

saveData.setDocumentId(Integer.valueOf(documentId));

saveData.setEffectiveDateStart(effDate);

saveData.setEffectiveDateEnd(endDate);

saveData.setStatus("Y");

saveData.setEntryDate(new Date());

repo.saveAndFlush(saveData);

if(list.size() > 0 ) {

// Update Old Record

DocumentMaster 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("Log Details"+e.getMessage());

return null;

}

return res;

}

public Long getmastertablecount (String companyId) {

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

//Select

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

//Effective Date Max Filter

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

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

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

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

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

list = result.getResultList();

data = list.get(0);

}

catch (Exception e) {

e.printStackTrace();

log.info("Log Details",e.getMessage());

return null;

}

return data;

}

@Override

public DocumentMasterGetRes getByDocumentId(DocumentMasterGetReq req) {

DocumentMasterGetRes res = new DocumentMasterGetRes();

DozerBeanMapper mapper = new DozerBeanMapper();

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

try {

//Criteria

CriteriaBuilder cb = em.getCriteriaBuilder();

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

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

//Find All

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

//Select

query.select(c);

//Effective Date Max Filter

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

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

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

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

javax.persistence.criteria.Predicate a2 = cb.equal(c.get("companyId"),ocpm1.get("companyId") ) ;

effectiveDate.where(a1,a2);

// 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("documentId"),req.getDocumentId()) ;

javax.persistence.criteria.Predicate n3 = cb.equal(c.get("companyId"),req.getCompanyId()) ;

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

// Get Result

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

list = result.getResultList();

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

res.setDocumentId(list.get(0).getDocumentId());

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("Log Details",e.getMessage());

return null;

}

return res;

}

@Override

public List<DocumentMasterGetRes> getallDocuments(DocumentMasterGetAllReq req) {

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

ModelMapper mapper = new ModelMapper();

try {

List<DocumentMaster> documentlist = new ArrayList<DocumentMaster>();

// Pagination

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

int offset = StringUtils.isBlank(req.getOffset())?10: Integer.valueOf(req.getOffset());

// Find Last Record

CriteriaBuilder cb = em.getCriteriaBuilder();

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

// Find all

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

// Select

query.select(b);

// Effective Date Max Filter

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

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

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

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

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

effectiveDate.where(a1,a2);

// Order By

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

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

// Where

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

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

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

// Get Result

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

result.setFirstResult(limit \* offset);

result.setMaxResults(offset);

documentlist = result.getResultList();

// Map

for (DocumentMaster data : documentlist) {

DocumentMasterGetRes res = new DocumentMasterGetRes();

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

mapper.getConfiguration().setAmbiguityIgnored(true);

res.setCompanyId(data.getCompanyId());

resList.add(res);

}

}

catch (Exception e) {

e.printStackTrace();

log.info("Log Details",e.getMessage());

return null;

}

return resList;

}

@Override

public List<DocumentMasterGetRes> getActiveDocument(DocumentMasterGetAllReq req) {

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

ModelMapper mapper = new ModelMapper();

try {

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

// Pagination

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

int offset = StringUtils.isBlank(req.getOffset())?10: Integer.valueOf(req.getOffset());

// Find Last Record

CriteriaBuilder cb = em.getCriteriaBuilder();

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

//Find all

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

//select

query.select(b);

// Effective Date Max Filter

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

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

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

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

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

effectiveDate.where(a1,a2);

//Order By

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

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

// Where

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

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

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

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

// Get Result

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

result.setFirstResult(limit \* offset);

result.setMaxResults(offset);

list = result.getResultList();

// Map

for (DocumentMaster data : list) {

DocumentMasterGetRes res = new DocumentMasterGetRes();

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

mapper.getConfiguration().setAmbiguityIgnored(true);

res.setDocumentId(data.getDocumentId());

resList.add(res);

}

}

catch(Exception e) {

e.printStackTrace();

log.info("Log Details", e.getMessage());

return null;

}

return resList;

}

}