@Service

@Transactional

public class ProductMasterServiceImpl implements ProductMasterService {

@PersistenceContext

private EntityManager em;

@Autowired

private ProductMasterRepository repo;

@Autowired

private BasicValidationService basicvalidateService;

Gson json = new Gson();

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

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*INSERT/UPDATE PRODUCT DETAILS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\\

@Transactional

@Override

public SuccessRes insertProduct(ProductMasterSaveReq req) {

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

SuccessRes res = new SuccessRes();

ProductMaster saveData = new ProductMaster();

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

DozerBeanMapper dozerMapper = new DozerBeanMapper();

try {

Calendar cal = new GregorianCalendar();

cal.setTime(req.getEffectiveDate()); cal.set(Calendar.HOUR\_OF\_DAY, 23); cal.set(Calendar.MINUTE, 59);

Date startDate = cal.getTime() ;

Date today = new Date();

cal.setTime(req.getEffectiveDate()); 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.getEffectiveDate()); 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 = sdformat.parse("12/12/2050");

String productId="";

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

// Save

//Long totalCount = repo.count();

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

productId = Long.valueOf(totalCount + 01).toString();

saveData.setProductId(Integer.valueOf(productId));

saveData.setProductName(req.getProductName());

res.setResponse("Saved Successfully ");

res.setSuccessId(productId);

} else {

// Update

// Get Less than Equal Today Record

// Criteria

productId=req.getProductId();

CriteriaBuilder cb = em.getCriteriaBuilder();

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

// Find All

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

// Select

query.select(b);

// Effective Date Max Filter

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

Root<ProductMaster> ocpm1 = effectiveDate.from(ProductMaster.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") , 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("productId"), req.getProductId() );

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

// Get Result

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

list = result.getResultList();

if( list.size() > 0) {

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

}

res.setResponse("Updated Successfully ");

res.setSuccessId(productId);

}

dozerMapper.map(req, saveData );

saveData.setProductId(Integer.valueOf(productId));

saveData.setProductName(req.getProductName());

saveData.setEffectiveDateStart(effDate);

saveData.setEffectiveDateEnd(endDate);

saveData.setStatus(req.getStatus());

saveData.setEntryDate(new Date());

repo.saveAndFlush(saveData);

if(list.size() > 0 ) {

// Update Old Record

ProductMaster 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;

}

@Override

public List<Error> validateProductDetails(ProductMasterSaveReq req) {

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

try {

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

errorList.add(new Error("02", "ProductName", "Please Select Product Name "));

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

errorList.add(new Error("02","ProductName", "Please Enter Product Name within 100 Characters"));

}else if (StringUtils.isBlank(req.getProductId().toString())) {

Long ProductCount = repo.countByProductNameAndCompanyIdOrderByEntryDateDesc(req.getProductName(), req.getCompanyId());

if (ProductCount > 0 ) {

errorList.add(new Error("01", "Product", "This Product Alrady Exist "));

}

}

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

errorList.add(new Error("03", "Remark", "Please Select Remark "));

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

errorList.add(new Error("03","Remark", "Please Enter Remark 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.getEffectiveDate() == null || StringUtils.isBlank(req.getEffectiveDate().toString())) {

errorList.add(new Error("04", "EffectiveDateStart", "Please Enter Effective Date Start"));

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

errorList.add(new Error("04", "EffectiveDateStart", "Please Enter Effective Date Start as Future Date"));

}

//Status Validation

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

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

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

errorList.add(new Error("05", "Status", "Enter Status 1 Character Only"));

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

errorList.add(new Error("05", "Status", "Enter Status Y or N Only"));

}

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

errorList.add(new Error("06", "CompanyId", "Please Select Company Id "));

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

errorList.add(new Error("06","CompanyId", "Please Enter Company Id within 20 Characters"));

}

if (req.getRsacode().length() > 10) {

errorList.add(new Error("07", "RSACODE", "Please Enter RSACODE within 10 Characters"));

}

if (req.getProductCategory().length() > 25) {

errorList.add(new Error("08", "ProductCategory", "Please Enter Product Category within 25 Characters"));

}

if (req.getPaymentYn().length() > 5) {

errorList.add(new Error("09", "PaymentYn", "Please Enter PaymentYn within 5 Characters"));

}if (req.getPaymentRedirUrl().length() > 500) {

errorList.add(new Error("10", "PaymentRedirUrl", "Please Enter getPaymentRedirUrl within 500 Characters"));

}if (req.getAppLoginUrl().length() > 100) {

errorList.add(new Error("11", "AppLoginUrl", "Please Enter AppLoginUrl within 100 Characters"));

}

} catch (Exception e) {

log.error(e);

e.printStackTrace();

}

return errorList;

}

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

// Select

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

// Effective Date Max Filter

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

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

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

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

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(e.getMessage());

}

return data;

}

///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*GET ALL\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\\

@Override

public List<ProductMasterRes> getallProductDetails(ProductMasterGetAllReq req) {

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

ModelMapper mapper = new ModelMapper();

try {

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

//Pagination

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

int offset = StringUtils.isBlank(req.getOffset()) ? 0 : Integer.valueOf(req.getOffset());

// Find Latest Record

CriteriaBuilder cb = em.getCriteriaBuilder();

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

// Find All

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

// Select

query.select(b);

// Effective Date Max Filter

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

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

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

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

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("productId")));

// 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<ProductMaster> result = em.createQuery(query);

result.setFirstResult(limit \* offset);

result.setMaxResults(offset);

list = result.getResultList();

// Map

for (ProductMaster data : list) {

ProductMasterRes res = new ProductMasterRes();

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

mapper.getConfiguration().setAmbiguityIgnored(true);

res.setProductId(data.getProductId().toString());

resList.add(res);

}

} catch (Exception e) {

e.printStackTrace();

log.info(e.getMessage());

return null;

}

return resList;

}

///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*GET BY ID\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\\

@Override

public ProductMasterRes getByProductCode(ProductMasterGetReq req) {

ProductMasterRes res = new ProductMasterRes();

ModelMapper mapper = new ModelMapper();

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

try {

// Criteria

CriteriaBuilder cb = em.getCriteriaBuilder();

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

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

// Find All

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

// Select

query.select(c );

// Effective Date Max Filter

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

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

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

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

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("productId"),req.getProductId()) ;

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

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

// Get Result

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

list = result.getResultList();

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

res.setProductId(list.get(0).getProductId().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;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DROPDOWN\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\\

@Override

public List<DropDownRes> getProductMasterDropdown() {

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

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();

// Criteria

CriteriaBuilder cb = em.getCriteriaBuilder();

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

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

// Find All

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

// Select

query.select(c );

// Order By

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

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

// Effective Date Max Filter

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

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

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

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

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

effectiveDate.where(a1,a2);

// Where

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

javax.persistence.criteria.Predicate n2 = cb.equal(c.get("effectiveDateStart"), effectiveDate);

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

// Get Result

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

list = result.getResultList();

for(ProductMaster data : list ) {

// Response

DropDownRes res = new DropDownRes();

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

res.setCodeDesc(data.getProductName());

resList.add(res);

}

} catch (Exception e) {

e.printStackTrace();

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

return null;

}

return resList;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*GET ACTIVE PRODUCT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\\

@Override

public List<ProductMasterRes> getActiveProductDetails(ProductMasterGetAllReq req) {

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

ModelMapper mapper = new ModelMapper();

try {

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

//Pagination

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

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

// Find Latest Record

CriteriaBuilder cb = em.getCriteriaBuilder();

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

// Find All

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

// Select

query.select(b);

// Effective Date Max Filter

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

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

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

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

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("productId")));

// 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<ProductMaster> result = em.createQuery(query);

result.setFirstResult(limit \* offset);

result.setMaxResults(offset);

list = result.getResultList();

// Map

for (ProductMaster data : list) {

ProductMasterRes res = new ProductMasterRes();

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

mapper.getConfiguration().setAmbiguityIgnored(true);

res.setProductId(data.getProductId().toString());

resList.add(res);

}

} catch (Exception e) {

e.printStackTrace();

log.info(e.getMessage());

return null;

}

return resList;

}

}