@Override

**public** MenuServiceRes menudisplay(MenuServiceReq req) {

MenuServiceRes response = **new** MenuServiceRes();

ModelMapper mapper = **new** ModelMapper();

**try** {

CriteriaBuilder cb= em.getCriteriaBuilder();

CriteriaQuery<MenuMaster> criteriaQuery = cb.createQuery(MenuMaster.**class**);

// Find All

Root<MenuMaster> m =criteriaQuery.from(MenuMaster.**class**);

// Select

criteriaQuery.select(m);

// Order By

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

orderList.add(cb.asc(m.get("menuId")));

Predicate p1 = cb.equal(m.get("displayYn") , "Y");

Predicate p2 = cb.equal(m.get("Status") , "Y");

Predicate p3 = **null** ;

**if**(req.getSubUserType().equalsIgnoreCase("high") ) {

p3 = cb.like(m.get("usertype"), "%" + "admin" + "%" );

} **else** {

p3 = cb.like(m.get("usertype"), "%" + req.getUserType() + "%" );

}

criteriaQuery.where(p1,p2,p3).orderBy(orderList);

TypedQuery<MenuMaster> query = em.createQuery(criteriaQuery);

List<MenuMaster> menulist = query.getResultList();

List<MenuMaster> adminMenulist = menulist.stream().filter( o -> o.getUsertype().equalsIgnoreCase("admin")).collect(Collectors.*toList*());

List<MenuMaster> otherMenulist = menulist.stream().filter( o ->! o.getUsertype().equalsIgnoreCase("admin")).collect(Collectors.*toList*());

List<AdminListRes> adminReslist = **new** ArrayList<AdminListRes>();

Type listType = **new** TypeToken<List<AdminListRes>>(){}.getType();

adminReslist = mapper.map(adminMenulist ,listType);

List<UserMenuListRes> userResList = **new** ArrayList<UserMenuListRes>();

Type listType2 = **new** TypeToken<List<UserMenuListRes>>(){}.getType();

userResList = mapper.map(otherMenulist ,listType2);

response.setAdminlist(adminReslist);

response.setUserList(userResList);

}

**catch**(Exception e) {

e.printStackTrace();

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

**return** **null**;

}

**return** response;

}