**DATE CONCEPTS**

**Data Format Condition**

Calendar cal = Calendar.getInstance();

cal.add(Calendar.DATE, -1);

Date yesterday = cal.getTime();

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

Date a=sdf.parse(req.getEffectivedate());

if (req.getEffectivedate() == null || StringUtils.isBlank(req.getEffectivedate().toString()))

{

errors.add(new Error("09", "Effective Date", "Please Enter Effective Date"));

}

else if(a.before(yesterday)) {

errors.add(new Error("09", "EffectiveDate", "Please Enter Future Date as EffectiveDate"));

}

else if(! req.getEffectivedate().matches("([0-9]{2})/([0-9]{2})/([0-9]{4})") )

{

errors.add(new Error("09","EffectiveDate","Effective Date format should be dd/MM/yyyy only allowed . Example :- 15/12/2020" ));

}

**Auto Increment Runnning Code and Auto Entry Date**

CRM CITY MASTER

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

String pattern = "#####0";

DecimalFormat decimalFormat = **new** DecimalFormat(pattern);

//////// Validation

@Override

**public** List<Error> validateCrmCityMaster(CrmCityMasterSaveReq req) {

// **TODO** Auto-generated method stub

List<Error> errors = **new** ArrayList<Error>();

**try** {

**if**( StringUtils.*isNotBlank*(req.getCitycode())) {

**if** ( ! StringUtils.*isNumeric*(req.getCitycode()) )

{

errors.add(**new** Error("01", "City Code", "Please Enter Valid City Code "));

}

}

**if**(req.getCityname()==**null** || StringUtils.*isBlank*(req.getCityname()))

{

errors.add(**new** Error("02", "City Name", "Please Enter City Name"));

}

**else** **if**(req.getCityname().length()>70)

{

errors.add(**new** Error("02", "City Name", "Please Enter City Name within 70 Characters"));

}

**if**(req.getStatecode()==**null** || StringUtils.*isBlank*(req.getStatecode()))

{

errors.add(**new** Error("03", "State Code", "Please Enter State Code"));

}

**if**(req.getStatus()==**null** || StringUtils.*isBlank*(req.getStatus()))

{

errors.add(**new** Error("04", "Status", "Please Enter Status"));

}

**else** **if**(req.getStatus().length()>1)

{

errors.add(**new** Error("04", "Status", "Please Enter Status within 1 Character"));

}

**if**(req.getCoreAppCode()==**null** || StringUtils.*isBlank*(req.getCoreAppCode()))

{

errors.add(**new** Error("05", "CoreAppCode", "Please Enter CoreAppCode"));

}

**else** **if**(req.getCoreAppCode().length()>50)

{

errors.add(**new** Error("05", "CoreAppCode", "Please Enter CoreAppCode within 50 Characters"));

}

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

{

errors.add(**new** Error("06", "Remarks", "Please Enter Remarks"));

}

**else** **if**(req.getRemarks().length()>200)

{

errors.add(**new** Error("06", "Remarks", "Please Enter Remarks within 200 Characters"));

}

Calendar cal = Calendar.*getInstance*();

cal.add(Calendar.***DATE***, -1);

Date yesterday = cal.getTime();

Date a= sdf.parse(req.getEffectivedate());

**if** (req.getEffectivedate() == **null** || StringUtils.*isBlank*(req.getEffectivedate().toString()))

{

errors.add(**new** Error("07", "Effective Date", "Please Enter Effective Date"));

}

**else** **if**(a.before(yesterday)) {

errors.add(**new** Error("07", "EffectiveDate", "Please Enter Future Date as EffectiveDate"));

}

**else** **if**(! req.getEffectivedate().matches("([0-9]{2})/([0-9]{2})/([0-9]{4})") )

{

errors.add(**new** Error("07","EffectiveDate","Effective Date format should be dd/MM/yyyy only allowed . Example :- 15/12/2020" ));

}

Date endDate =sdf.parse(req.getEnddate());

Date effectiveDate = sdf.parse(req.getEffectivedate());

**if** (req.getEnddate() == **null** || StringUtils.*isBlank*(req.getEnddate().toString()))

{

errors.add(**new** Error("08", "End Date", "Please Enter End Date"));

}

**else** **if**(endDate.before(effectiveDate))

{

errors.add(**new** Error("08","End Date", "End Date not before Effective Date"));

}

**else** **if**(! req.getEnddate().matches("([0-9]{2})/([0-9]{2})/([0-9]{4})") )

{

errors.add(**new** Error("08","End Date","End Date format should be dd/MM/yyyy only allowed . Example :- 15/12/2020" ));

}

} **catch**(Exception e) {

e.printStackTrace();

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

**return** errors;

}

**return** errors;

}

//////// Save

@Override

@Transactional

**public** SuccessRes saveCrmCityMaster(CrmCityMasterSaveReq req) {

// **TODO** Auto-generated method stub

SuccessRes res = **new** SuccessRes();

CrmCityMaster entity = **new** CrmCityMaster();

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

Date entryDate = **null**;

Double cityCode = 0D;

**try**

{

**if**(StringUtils.*isBlank*(req.getCitycode()) ) {

CrmCityMaster crmcitymaster = repository.findTop1ByOrderByCitycodeDesc();

cityCode =Double.*valueOf*(crmcitymaster.getCitycode()+1 );

entryDate = **new** Date();

res.setResponse("Saved Successfully");

}

**else** {

//Update

cityCode =Double.*valueOf*(req.getCitycode());

CrmCityMaster data = repository.findTop1ByCitycodeOrderByEffectiveDateDesc(cityCode);

entryDate = data.getEntryDate();

res.setResponse("Updated Successfully");

}

entity.setCitycode(Double.*valueOf*(cityCode));

entity.setCityname(req.getCityname());

entity.setCoreAppCode(req.getCoreAppCode());

entity.setEffectiveDate(sdf.parse(req.getEffectivedate()));

entity.setEndDate(sdf.parse(req.getEnddate()));

entity.setStatecode(Double.*valueOf*(req.getStatecode()));

entity.setStatus(req.getStatus());

entity.setRemarks(req.getRemarks());

entity.setEntryDate(entryDate);

repository.save(entity);

log.info("Saved Details is ---> "+ json.toJson(entity));

}

**catch**(Exception ex) {

log.error(ex);

**return** **null**;

}

**return** res;

}

//////Get All

@Override

**public** List<CrmCityMasterRes> getAllCrmCityMaster() {

// **TODO** Auto-generated method stub

List<CrmCityMasterRes> resList = **new** ArrayList<CrmCityMasterRes>();

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

**try** {

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

List<CrmCityMaster> crmcitymaster = repository.OrderByCitycodeDesc();

**for**(CrmCityMaster data: crmcitymaster)

{

CrmCityMasterRes res = **new** CrmCityMasterRes();

res.setCoreAppCode(data.getCoreAppCode());

res.setEffectivedate(data.getEffectiveDate()==**null**?"":sdf.format(data.getEffectiveDate()));

res.setEnddate(data.getEffectiveDate()==**null**?"":sdf.format(data.getEndDate()));

res.setEntrydate(data.getEffectiveDate()==**null**?"":sdf.format(data.getEntryDate()));

res.setRemarks(data.getRemarks());

res.setStatus(data.getStatus());

res.setStatecode(decimalFormat.format(data.getStatecode()));

res.setCitycode(decimalFormat.format(data.getCitycode()));

res.setCityname(data.getCityname());

resList.add(res);

}

} **catch**(Exception e) {

e.printStackTrace();

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

**return** **null**;

}

**return** resList;

}

/////////// Get

@Override

**public** CrmCityMasterRes getCrmCityMaster(String citycode) {

CrmCityMasterRes res = **new** CrmCityMasterRes();

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

**try** {

//Map

CrmCityMaster data = repository.findByCitycode(Double.*valueOf*(citycode));

//res = mapper.map(data,CrmCityMasterRes.class);

res.setEffectivedate(data.getEffectiveDate()==**null**?"":sdf.format(data.getEffectiveDate()));

res.setEnddate(data.getEffectiveDate()==**null**?"":sdf.format(data.getEndDate()));

res.setEntrydate(data.getEffectiveDate()==**null**?"":sdf.format(data.getEntryDate()));

res.setCitycode(decimalFormat.format(data.getCitycode()));

res.setStatecode(decimalFormat.format(data.getStatecode()));

res.setRemarks(data.getRemarks());

res.setStatus(data.getStatus());

res.setCityname(data.getCityname());

res.setCoreAppCode(data.getCoreAppCode());

}

**catch**(Exception e) {

e.printStackTrace();

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

**return** **null**;

}

**return** res;

}

}

**CRM STATE MASTER**

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

/////Decimal to String Convert/////

String pattern = "#####0";

DecimalFormat decimalFormat = **new** DecimalFormat(pattern);

@Override

**public** List<Error> validateCrmStateMaster(CrmStateMasterSaveReq req) {

// **TODO** Auto-generated method stub

List<Error> errors = **new** ArrayList<Error>();

**try** {

**if**( StringUtils.*isNotBlank*(req.getStatecode())) {

**if** ( ! StringUtils.*isNumeric*(req.getStatecode()) )

{

errors.add(**new** Error("01", "State Code", "Please Enter Valid State Code "));

}

}

**if**(req.getStatename()== **null** || StringUtils.*isBlank*(req.getStatename()))

{

errors.add(**new** Error("02", "State Name", "Please Enter State Name"));

}

**else** **if**(req.getStatename().length()>200)

{

errors.add(**new** Error("02", "State Name", "Please Enter State Name within 200 Characters"));

}

**if**(req.getStatus()== **null** || StringUtils.*isBlank*(req.getStatus()))

{

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

}

**else** **if**(req.getStatus().length()>1)

{

errors.add(**new** Error("03", "Status", "Please Enter Status within 1 Character"));

}

Calendar cal = Calendar.*getInstance*();

cal.add(Calendar.***DATE***, -1);

Date yesterday = cal.getTime();

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

Date a= sdf.parse(req.getEffectivedate());

**if** (req.getEffectivedate() == **null** || StringUtils.*isBlank*(req.getEffectivedate().toString()))

{

errors.add(**new** Error("04", "Effective Date", "Please Enter Effective Date"));

}

**else** **if**(a.before(yesterday)) {

errors.add(**new** Error("04", "EffectiveDate", "Please Enter Future Date as EffectiveDate"));

}

**else** **if**(! req.getEffectivedate().matches("([0-9]{2})/([0-9]{2})/([0-9]{4})") )

{

errors.add(**new** Error("04","EffectiveDate","Effective Date format should be dd/MM/yyyy only allowed . Example :- 15/12/2020" ));

}

Date endDate =sdf.parse(req.getEnddate());

Date effectiveDate = sdf.parse(req.getEffectivedate());

**if** (req.getEnddate() == **null** || StringUtils.*isBlank*(req.getEnddate().toString()))

{

errors.add(**new** Error("05", "End Date", "Please Enter End Date"));

}

**else** **if**(endDate.before(effectiveDate))

{

errors.add(**new** Error("05","End Date", "End Date not before Effective Date"));

}

**else** **if**(! req.getEnddate().matches("([0-9]{2})/([0-9]{2})/([0-9]{4})") )

{

errors.add(**new** Error("05","End Date","End Date format should be dd/MM/yyyy only allowed . Example :- 15/12/2020" ));

}

**if**(req.getCoreappcode()== **null** || StringUtils.*isBlank*(req.getCoreappcode()))

{

errors.add(**new** Error("06", "Core App Code", "Please Enter Core App Code"));

}

**else** **if**(req.getCoreappcode().length()>50)

{

errors.add(**new** Error("06", "Core App Code", "Please Enter Core App Code within 50 Characters"));

}

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

{

errors.add(**new** Error("07", "Remarks", "Please Enter Remarks"));

}

**else** **if**(req.getRemarks().length()>200)

{

errors.add(**new** Error("07", "Remarks", "Please Enter Remarks within 200 Characters"));

}

} **catch**(Exception e) {

e.printStackTrace();

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

**return** errors;

}

**return** errors;

}

@Override

@Transactional

**public** SuccessRes saveCrmStateMaster(CrmStateMasterSaveReq req) {

SuccessRes res = **new** SuccessRes();

CrmStateMaster entity = **new** CrmStateMaster();

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

Date entryDate = **null**;

Double stateCode = 0D;

**try**

{

**if**(StringUtils.*isBlank*(req.getStatecode()) ) {

CrmStateMaster crmstatemaster = repository.findTop1ByOrderByStatecodeDesc();

stateCode =Double.*valueOf*(crmstatemaster.getStatecode()+1 );

entryDate = **new** Date();

res.setResponse("Saved Successfully");

}

**else** {

//Update

stateCode =Double.*valueOf*( req.getStatecode());

CrmStateMaster data = repository.findTop1ByStatecodeOrderByEffectiveDateDesc(stateCode);

entryDate = data.getEntryDate();

res.setResponse("Updated Successfully");

}

// Primary

entity.setStatecode(Double.*valueOf*(stateCode));

entity.setCoreappcode(req.getCoreappcode());

entity.setEffectiveDate(sdf.parse(req.getEffectivedate()));

entity.setStatename(req.getStatename());

entity.setEndDate(sdf.parse(req.getEnddate()));

entity.setRemarks(req.getRemarks());

entity.setStatus(req.getStatus());

entity.setEntryDate(entryDate);

repository.save(entity);

log.info("Saved Details is ---> "+ json.toJson(entity));

}

**catch**(Exception ex) {

log.error(ex);

**return** **null**;

}

**return** res;

}

@Override

**public** List<CrmStateMasterRes> getAllCrmStateMaster() {

{

List<CrmStateMasterRes> resList = **new** ArrayList<CrmStateMasterRes>();

// ModelMapper mapper = new ModelMapper();

**try** {

List<CrmStateMaster> crmstatemaster = repository.OrderByStatecodeDesc();

**for**(CrmStateMaster data: crmstatemaster)

{

CrmStateMasterRes res = **new** CrmStateMasterRes();

res.setCoreappcode(data.getCoreappcode());

res.setEffectivedate(sdf.format(data.getEffectiveDate()));

res.setEnddate(sdf.format(data.getEndDate()));

res.setEntrydate(sdf.format(data.getEntryDate()));

res.setRemarks(data.getRemarks());

res.setStatename(data.getStatename());

res.setStatus(data.getStatus());

res.setStatecode(decimalFormat.format(data.getStatecode()));

resList.add(res);

}

}

**catch**(Exception e)

{

e.printStackTrace();

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

**return** **null**;

}

**return** resList;

}

}

@Override

**public** CrmStateMasterRes getCrmStateMaster(String statecode) {

CrmStateMasterRes res = **new** CrmStateMasterRes();

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

**try** {

//Map

CrmStateMaster data = repository.findByStatecode(Double.*valueOf*(statecode));

res = mapper.map(data,CrmStateMasterRes.**class**);

res.setEffectivedate(sdf.format(data.getEffectiveDate()));

res.setEnddate(sdf.format(data.getEffectiveDate()));

res.setEntrydate(sdf.format(data.getEntryDate()));

res.setStatecode(decimalFormat.format(data.getStatecode()));

} **catch**(Exception e) {

e.printStackTrace();

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

**return** **null**;

}

**return** res;

}

}