**Controller**

@PostMapping(value = "/savelead")

// Insert

**public** ResponseEntity<CommonCrmRes> saveCrmCityMaster( @RequestBody CrmLeadSaveReq req) {

reqPrinter.reqPrint(req);

CommonCrmRes data = **new** CommonCrmRes();

List<Error> validation = entityService.validateCrmCityMaster(req);

//// validation

**if** (validation != **null** && validation.size() != 0) {

data.setCommonResponse(**null**);

data.setIsError(**true**);

data.setErrorMessage(validation);

data.setMessage("Failed");

**return** **new** ResponseEntity<CommonCrmRes>(data, HttpStatus.***OK***);

} **else** {

/////// save

CrmLeadSuccessRes res = entityService.saveCrmCityMaster(req);

data.setCommonResponse(res);

data.setIsError(**false**);

data.setErrorMessage(Collections.*emptyList*());

data.setMessage("Success");

**if** (res != **null**) {

**return** **new** ResponseEntity<CommonCrmRes>(data, HttpStatus.***CREATED***);

} **else** {

**return** **new** ResponseEntity<>(**null**, HttpStatus.***BAD\_REQUEST***);

}

}

}

**Service Impl**

// Validation

@Override

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

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

**try** {

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

**if** (!StringUtils.*isNumeric*(req.getLeadId())) {

errors.add(**new** Error("01", "Lead Id", "Please Enter Valid Lead Id "));

}

}

**if** (req.getClientId() == **null** || StringUtils.*isBlank*(req.getClientId())) {

errors.add(**new** Error("02", "Client Id", "Please Enter Client Id"));

}

**if** (req.getBusinessType() == **null** || StringUtils.*isBlank*(req.getBusinessType())) {

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

}**else** **if** (req.getBusinessType().length() > 100) {

errors.add(**new** Error("03", "Business Type", "Please Enter Business Type within 100 Characters"));

}

**if** (req.getBusinessTypeId() == **null** || StringUtils.*isBlank*(req.getBusinessTypeId())) {

errors.add(**new** Error("04", "Business Type Id", "Please Enter Business Type Id"));

}

} **catch** (Exception e) {

e.printStackTrace();

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

**return** errors;

}

**return** errors;

}

@Override

**public** CrmLeadSuccessRes saveCrmCityMaster(CrmLeadSaveReq req) {

CrmLeadSuccessRes res = **new** CrmLeadSuccessRes();

LeadDetails entity = **new** LeadDetails();

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

SimpleDateFormat dbf = **new** SimpleDateFormat("yyyy-MM-dd");

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

Integer leadId = 0;

Date entryDate = **null**;

**try** {

req.setDueDate(dbf.format(sdf.parse(req.getDueDate())));

req.setLeadGenDate(dbf.format(sdf.parse(req.getLeadGenDate())));

entity = mapper.map(req, LeadDetails.**class**);

entity.setStatus("Y");

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

Long totalLeadCount = repository.count();

leadId = Integer.*valueOf*(totalLeadCount.toString()) + 10001;

entryDate = **new** Date();

res.setResponse("Saved Successfully ");

res.setLeadid(Integer.*valueOf*(leadId).toString());

entity.setEntryDate(entryDate);

entity.setLeadId(leadId);

} **else** {

// Update

leadId = Integer.*valueOf*(req.getLeadId());

LeadDetails data = repository.findByLeadId(leadId);

entryDate = data.getEntryDate();

res.setResponse("Updated Successfully ");

res.setLeadid(Integer.*valueOf*(leadId).toString());

entity.setEntryDate(entryDate);

entity.setLeadId(leadId);

}

repository.save(entity);

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

} **catch** (Exception ex) {

log.error(ex);

**return** **null**;

}

**return** res;

}

**Get All**

////////Get All

@GetMapping("/getall/lead")

**public** ResponseEntity<CommonCrmRes> getAllLead() {

CommonCrmRes data = **new** CommonCrmRes();

// Get All

List<CrmLeadRes> res = entityService.getAllLead();

data.setCommonResponse(res);

data.setIsError(**false**);

data.setErrorMessage(Collections.*emptyList*());

data.setMessage("Success");

**if** (res != **null**) {

**return** **new** ResponseEntity<CommonCrmRes>(data, HttpStatus.***CREATED***);

} **else** {

**return** **new** ResponseEntity<>(**null**, HttpStatus.***BAD\_REQUEST***);

}

}

**Service Impl**

/// Get All Lead

@Override

**public** List<CrmLeadRes> getAllLead() {

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

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

**try** {

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

List<LeadDetails> leadDetails = repository.OrderByLeadIdAsc();

**for** (LeadDetails data : leadDetails) {

CrmLeadRes res = **new** CrmLeadRes();

mapper.map(data, res);

res.setDueDate(data.getDueDate()==**null**?"":sdf.format( data.getDueDate()));

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

res.setLeadGenDate(data.getLeadGenDate() ==**null**?"":sdf.format(data.getLeadGenDate()));

resList.add(res);

}

}

**catch** (Exception e) {

e.printStackTrace();

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

**return** **null**;

}

**return** resList;

}

**Controller**

//// Get By Lead Id

@GetMapping("/get/lead/{leadId}")

**public** ResponseEntity<CommonCrmRes> getLead(@PathVariable("leadId") String leadId) {

CommonCrmRes data = **new** CommonCrmRes();

CrmLeadRes res = entityService.getLead(leadId);

data.setCommonResponse(res);

data.setIsError(**false**);

data.setErrorMessage(Collections.*emptyList*());

data.setMessage("Success");

**if** (res != **null**) {

**return** **new** ResponseEntity<CommonCrmRes>(data, HttpStatus.***CREATED***);

} **else** {

**return** **new** ResponseEntity<>(**null**, HttpStatus.***BAD\_REQUEST***);

}

}

}

**Service Impl**

// Get by Lead Id

@Override

**public** CrmLeadRes getLead(String leadId) {

CrmLeadRes res = **new** CrmLeadRes();

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

**try** {

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

LeadDetails data = repository.findByLeadId(Integer.*valueOf*(leadId));

mapper.map(data, res);

res.setDueDate(data.getDueDate()==**null**?"":sdf.format( data.getDueDate()));

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

res.setLeadGenDate(data.getLeadGenDate() ==**null**?"":sdf.format(data.getLeadGenDate()));

}

**catch** (Exception e) {

e.printStackTrace();

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

**return** **null**;

}

**return** res;

}