Document Upload Code

In Controller create validation and save method like this

```
@PostMapping(value = "/master/save/branch")
public ResponseEntity<CommonCrmRes> saveBranchMaster(@RequestBody
CrmBranchMasterSaveReq req) {
reqPrinter.reqPrint(req);
CommonCrmRes data = new CommonCrmRes();

// Base 64 to File
MultipartFile headerfile=null;
MultipartFile footerfile=null;
MultipartFile signfile = null;
MultipartFile stampfile= null;
BranchFiles bf = new BranchFiles();

headerfile= new
com.maan.crm.service.impl.BASE64DecodedMultipartFile(Base64Utils.decode
FromString(req.getHeaderImg().split(",")[1]),req.getHeaderFileName());
footerfile = new
com.maan.crm.service.impl.BASE64DecodedMultipartFile(Base64Utils.decode
FromString(req.getFooterImg().split(",")[1]),req.getFooterFileName());
signfile  = new
com.maan.crm.service.impl.BASE64DecodedMultipartFile(Base64Utils.decode
FromString(req.getSignImg().split(",")[1]),req.getSignFileName());
stampfile = new
com.maan.crm.service.impl.BASE64DecodedMultipartFile(Base64Utils.decode
FromString(req.getStamp().split(",")[1]),req.getStampFileName());

bf.setHeaderfile(headerfile);
bf.setFooterfile(footerfile);
bf.setSignfile(signfile);
bf.setStampfile(stampfile);


List<Error> validation = entityService.validateBranchMaster(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+

CrmBranchMasterSuccessRes res = entityService.saveCrmBranchMaster(req,bf
);
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);
}
}

}
```

In service Implementation save method create like this

Add this two lines before method creation in service implementation class

```
@Value("${file.upload-dir2}")
private String directoryPath;

@Value("${common.file.path}")
private String compressedPath;



/////// Validation and Save Method Creation


@Override
public CrmBranchMasterSuccessRes
saveCrmBranchMaster(CrmBranchMasterSaveReq req, BranchFiles bf) {
CrmBranchMasterSuccessRes res = new CrmBranchMasterSuccessRes();
CrmBranchMaster entity = new CrmBranchMaster();
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
```

```
ModelMapper mapper = new ModelMapper();
Date entryDate = null;
Double branchCode = 0D;
try {
if (StringUtils.isBlank(req.getBranchCode())) {
CrmBranchMaster crmBranchMaster =
repository.findTop1ByOrderByBranchCodeDesc();
branchCode = Double.valueOf( crmBranchMaster.getBranchCode() + 1);
entryDate = new Date();
res= mapper.map(req, CrmBranchMasterSuccessRes.class);
res.setResponse("Saved Successfully ");
res.setBranchcode(decimalFormat.format(branchCode));
}
else {
// Update
branchCode = Double.valueOf(req.getBranchCode());
CrmBranchMaster data =
repository.findTop1ByBranchCodeOrderByEffectiveDateDesc(branchCode);
entryDate = data.getEntryDate();
res.setResponse("Updated Successfully ");
res.setBranchcode(decimalFormat.format(branchCode));
}

// Upload Images
String res1="";
MultipartFile headerfile = bf.getHeaderfile();
MultipartFile footerfile = bf.getFooterfile();
MultipartFile signfile = bf.getSignfile();
MultipartFile stampfile = bf.getStampfile();
Random random = new Random();
String newfilename, fileextension,newfilename1;
String headerfilename, footerfilename, signfilename, stampfilename ;
String headerfilename1= "";
String footerfilename1= "";
String signfilename1 = "";
String stampfilename1 ="";
File file1;
Path destination;

// Save Original Image
destination = Paths.get(directoryPath) ;
// (i) Header Image
```

```
headerfilename= random.nextInt(100) +
generateFileName()+"."+FilenameUtils.getExtension(headerfile.getOriginalFile
name());
Files.copy(headerfile.getInputStream(),destination.resolve(headerfilename));
fileextension = FilenameUtils.getExtension(headerfile.getOriginalFilename());
// (ii) Footer Image
footerfilename= random.nextInt(100) +
generateFileName()+"."+FilenameUtils.getExtension(footerfile.getOriginalFilen
ame());
Files.copy(footerfile.getInputStream(),destination.resolve(footerfilename));
fileextension = FilenameUtils.getExtension(footerfile.getOriginalFilename());
// (iii) Sign Image
signfilename= random.nextInt(100) +
generateFileName()+"."+FilenameUtils.getExtension(signfile.getOriginalFilena
me());
Files.copy(signfile.getInputStream(),destination.resolve(signfilename));
fileextension = FilenameUtils.getExtension(signfile.getOriginalFilename());
//(iv) Stamp
stampfilename= random.nextInt(100) +
generateFileName()+"."+FilenameUtils.getExtension(stampfile.getOriginalFilen
ame());
Files.copy(stampfile.getInputStream(),destination.resolve(stampfilename));
fileextension = FilenameUtils.getExtension(stampfile.getOriginalFilename());

//File Name (Ex:Jpg)

// Save Compress Header Image
if(fileextension.equals("bmp") || fileextension.equals("jpg") ||
fileextension.equals("jpeg")) {
headerfilename1= compressedPath + random.nextInt(100) +
generateFileName()+"."+FilenameUtils.getExtension(headerfile.getOriginalFile
name());
file1 = new File(directoryPath+headerfilename);
CompressImage(file1,headerfilename1);
}else {
headerfilename1 = directoryPath+headerfilename;
}

// Save Compress Header Image
if(fileextension.equals("bmp") || fileextension.equals("jpg") ||
fileextension.equals("jpeg")) {
```

```
footerfilename1= compressedPath + random.nextInt(100) +
generateFileName()+"."+FilenameUtils.getExtension(footerfile.getOriginalFilen
ame());
file1 = new File(directoryPath+footerfilename);
CompressImage(file1,footerfilename1);
}else {
footerfilename1 = directoryPath+footerfilename;
}

// Save Compress Sign Image
if(fileextension.equals("bmp") || fileextension.equals("jpg") ||
fileextension.equals("jpeg")) {
signfilename1= compressedPath + random.nextInt(100) +
generateFileName()+"."+FilenameUtils.getExtension(signfile.getOriginalFilena
me());
file1 = new File(directoryPath+signfilename);
CompressImage(file1,signfilename1);
}else {
signfilename1 = directoryPath+signfilename;
}
// Save Compress Stamp Image
if(fileextension.equals("bmp") || fileextension.equals("jpg") ||
fileextension.equals("jpeg")) {
stampfilename1= compressedPath + random.nextInt(100) +
generateFileName()+"."+FilenameUtils.getExtension(stampfile.getOriginalFilen
ame());
file1 = new File(directoryPath+stampfilename);
CompressImage(file1,stampfilename1);
}else {
stampfilename1 = directoryPath+stampfilename;
}


entity.setBranchCode(branchCode);
entity.setEntryDate(entryDate);
entity.setAddress1(req.getAddress1());
entity.setAddress2(req.getAddress2());
entity.setAddress3(req.getAddress3());
entity.setBelongingBranch(req.getBelongingBranch());
entity.setBelongingType(req.getBelongingType());
entity.setBranchName(req.getBranchName());
entity.setBranchPrefix(req.getBranchPrefix());
entity.setCity(req.getCity());
entity.setCoreAppCode(req.getCoreAppCode());
entity.setCountry(req.getCountry());
entity.setCurrencyAbbreviation(req.getCurrencyAbbreviation());
entity.setCurrencyAcronym(req.getCurrencyAcronym());
entity.setCurrencyDecimalDigit(new
BigDecimal(Double.valueOf(req.getCurrencyDecimalDigit())));
entity.setCurrencyDecimalName(req.getCurrencyDecimalName());
entity.setCurrencyName(req.getCurrencyName());
entity.setDecimalPlaces(new
BigDecimal(Double.valueOf(req.getDecimalPlaces())));
entity.setDeptCode(req.getDeptCode());
entity.setDestinationCountryId(new
BigDecimal(Double.valueOf(req.getDestinationCountryId())));
entity.setEffectiveDate(sdf.parse(req.getEffectiveDate()));
entity.setEmail(req.getEmail());
entity.setExchangeId(new BigDecimal(Integer.valueOf(req.getExchangeId())));
entity.setFax(new BigDecimal(req.getFax()));
entity.setFooterImg(footerfilename);
entity.setHeaderCompressedImg(headerfilename1);
entity.setHeaderOriginalImg(headerfilename);
entity.setFooterCompressedImg(footerfilename1);
entity.setFooterOriginalImg(footerfilename);
entity.setSignCompressedImg(signfilename1);
entity.setSignOriginalImg(signfilename);
entity.setStampCompressedImg(stampfilename1);
entity.setStampOriginalImg(stampfilename);
entity.setInsCompanyId(Double.valueOf(req.getInsCompanyId()));
entity.setLongitudeYn(req.getLongitudeYn());
entity.setOriginationCountryId(new
BigDecimal(Double.valueOf(req.getOriginationCountryId())));
entity.setPckeyCloseTrn(req.getPckeyCloseTrn());
entity.setPhone(req.getPhone());
entity.setRegionCode(req.getRegionCode());
entity.setRemarks(req.getRemarks());
entity.setSignImg(signfilename);
entity.setStamp(stampfilename);
entity.setHeaderImg(headerfilename);
entity.setStatus(req.getStatus());
entity.setTax(new BigDecimal(Double.valueOf(req.getTax())));
repository.save(entity);
log.info("Saved Details is ---> " + json.toJson(entity));
} catch (Exception ex) {
log.error(ex);
```

```
return null;
}
return res;
}
private String generateFileName() {
SimpleDateFormat sdf = new
SimpleDateFormat("ddMMyyyyhmmssSSSSSSa");
Calendar cal = Calendar.getInstance();
String date = sdf.format(cal.getTime());
return date;
}
public void CompressImage(File uploadFile, String documentPath) {

try {
String extension = FilenameUtils.getExtension(documentPath);
File jpgoutput = new File("thumbnail." + extension);
BufferedImage originalImage = ImageIO.read(uploadFile);
Thumbnails.of(originalImage).size(750,
750).outputFormat(extension).toFile(jpgoutput);
FileUtils.copyFile(jpgoutput, new File(documentPath));
if(jpgoutput.exists()) {
log.info("Thumbnail File Deleted after conversion");
FileUtils.deleteQuietly(jpgoutput);
}
} catch (IOException e) {
e.printStackTrace();
try {
FileUtils.copyFile(uploadFile, new File(documentPath));
} catch (IOException ex) {
ex.printStackTrace();
}
}

}
```

For Get Method create like this

```
@Override
public CrmBranchMasterRes getCrmBranchMaster(String branchCode) {
CrmBranchMasterRes res = new CrmBranchMasterRes();
```

```
ModelMapper mapper = new ModelMapper();
try {
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
CrmBranchMaster data =
repository.findByBranchCode(Double.valueOf(branchCode));
res.setEffectiveDate(data.getEffectiveDate()==null?"":sdf.format(
data.getEffectiveDate()));
res.setAddress1(data.getAddress1()== null ?"": data.getAddress1());
res.setAddress2(data.getAddress2() == null ?"": data.getAddress2());
res.setAddress3(data.getAddress3() == null ?"": data.getAddress3());
res.setBelongingBranch(data.getBelongingBranch()==null ? "" :
data.getBelongingBranch());
res.setBelongingType(data.getBelongingType() ==null ? "" :
data.getBelongingType());
res.setBranchCode(data.getBranchCode()== null ?"" :
decimalFormat.format(data.getBranchCode()));
res.setBranchName(data.getBranchName()== null ?"" :
data.getBranchName());
res.setBranchPrefix(data.getBranchPrefix() == null ?"":
data.getBranchPrefix());
res.setCity(data.getCity() == null ?"":data.getCity());
res.setCoreAppCode(data.getCoreAppCode() == null?"":
data.getCoreAppCode());
res.setCountry(data.getCountry()==null?"":data.getCountry());
res.setCurrencyAbbreviation(data.getCurrencyAbbreviation()==null ?"":
data.getCurrencyAbbreviation());
res.setCurrencyAcronym(data.getCurrencyAcronym()==null ?"":
data.getCurrencyAcronym());
res.setCurrencyDecimalDigit(data.getCurrencyDecimalDigit()== null ?"":
decimalFormat.format(data.getCurrencyDecimalDigit()));
res.setCurrencyDecimalName(data.getCurrencyDecimalName() == null ?"":
data.getCurrencyDecimalName());
res.setCurrencyName(data.getCurrencyName()== null ?"":
data.getCurrencyName());
res.setDecimalPlaces(data.getDecimalPlaces()== null
?"":decimalFormat.format(data.getDecimalPlaces()));
res.setDeptCode(data.getDeptCode()==null ?"": data.getDeptCode());
res.setDestinationCountryId(data.getDestinationCountryId() == null
?"":decimalFormat.format(data.getDestinationCountryId()));
res.setEmail(data.getEmail()==null ?"": data.getEmail());
res.setEntryDate(data.getEntryDate()==null ?"":
sdf.format(data.getEntryDate()));
res.setExchangeId(data.getExchangeId()==null
?"":decimalFormat.format(data.getExchangeId()));
res.setFax(data.getFax()==null ?"":decimalFormat.format(data.getFax()));
res.setFooterImg(data.getFooterImg()==null ?"": data.getFooterImg());
res.setHeaderImg(data.getHeaderImg()==null ?"": data.getHeaderImg());
res.setInsCompanyId(data.getInsCompanyId()==null?"":decimalFormat.format(
data.getInsCompanyId()));
res.setLongitudeYn(data.getLongitudeYn()==null ?"": data.getLongitudeYn());
res.setOriginationCountryId(data.getOriginationCountryId()==null?""
        :decimalFormat.format(data.getOriginationCountryId()));
res.setPckeyCloseTrn(data.getPckeyCloseTrn()==null ?"":
data.getPckeyCloseTrn());
res.setPhone(data.getPhone()==null ?"": data.getPhone());
res.setRegionCode(data.getRegionCode() == null ?"": data.getRegionCode());
res.setRemarks(data.getRemarks()==null ?"": data.getRemarks());
res.setSignImg(data.getSignImg()==null ?"": data.getSignImg());
res.setStamp(data.getStamp()== null ?"": data.getStamp());
res.setStatus(data.getStatus()== null ?"": data.getStatus());
res.setTax(data.getTax()== null ?"": decimalFormat.format(data.getTax()));
res.setHeaderImg(data.getHeaderImg());
res.setFooterImg(data.getFooterImg());
res.setSignImg(data.getSignImg());
res.setStamp(data.getStamp());

/*
* if (StringUtils.isNotBlank(data.getHeaderCompressedImg()) && new
* File(data.getHeaderCompressedImg()).exists()) { res.setHeaderImg(new
* GetFileFromPath(data.getHeaderCompressedImg()).call().getImgUrl()); }
*
* if (StringUtils.isNotBlank(data.getFooterCompressedImg()) && new
* File(data.getFooterCompressedImg()).exists()) { res.setFooterImg(new
* GetFileFromPath(data.getFooterCompressedImg()).call().getImgUrl()); }
*
* if (StringUtils.isNotBlank(data.getSignCompressedImg()) && new
* File(data.getSignCompressedImg()).exists()) { res.setSignImg(new
* GetFileFromPath(data.getSignCompressedImg()).call().getImgUrl()); }
*
* if (StringUtils.isNotBlank(data.getStampCompressedImg()) && new
* File(data.getStampCompressedImg()).exists()) { res.setStamp(new
* GetFileFromPath(data.getStampCompressedImg()).call().getImgUrl()); }
*/
}
```

```
catch (Exception e) {
e.printStackTrace();
log.info("Exception is ---> " + e.getMessage());
return null;
}
return res;

}
```

In application Properties add these lines for setting path of the image

```
#dumm
// For External Storage
#common.file.path=\\\\192.168.1.99\\CommonPath\\compressedImg\\
#file.upload-dir=\\\\192.168.1.99\\CommonPath\\orginalimg\\
// For Local Storage
common.file.path=E:\\compressedImg\\
file.upload-dir2=E:\\orginalimg\\
```

Add this base 64 class in your service implementation

```
package com.maan.crm.service.impl;


import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;


import org.apache.tika.Tika;
import org.springframework.web.multipart.MultipartFile;
```

```
public class BASE64DecodedMultipartFile implements MultipartFile {

private byte[] data;
private String filename;
public BASE64DecodedMultipartFile(byte[] data,String filename) {
this.data=data;
this.filename=filename;
}

@Override
public String getName() {
// TODO Auto-generated method stub
return filename;
}

@Override
public String getOriginalFilename() {
// TODO Auto-generated method stub
return filename;
}

@Override
public String getContentType() {
if(data!=null  ||data.length==0) {
Tika tika = new Tika();
return tika.detect(data);
}else
return null;
}

@Override
public boolean isEmpty() {
return (data==null||data.length==0)?true:false;
}

@Override
public long getSize() {
// TODO Auto-generated method stub
return data.length;
}

@Override
public byte[] getBytes() throws IOException {
// TODO Auto-generated method stub
return data;
}

@Override
public InputStream getInputStream() throws IOException {
// TODO Auto-generated method stub
return new ByteArrayInputStream(data);
}
```

```
@Override
public void transferTo(File dest) throws IOException, IllegalStateException {
// TODO Auto-generated method stub

}

}
```

Get File From Path (Class Name)

Add this class in your project

```
package com.maan.crm.service.impl;

import java.io.File;
import java.util.concurrent.Callable;

import org.apache.commons.io.FileUtils;
import org.apache.commons.lang3.StringUtils;
import org.springframework.util.Base64Utils;
import org.springframework.web.multipart.MultipartFile;

import com.maan.crm.res.Document;

public class GetFileFromPath implements Callable<Object> {
private String path;

public GetFileFromPath(String path) {
super();
this.path = path;
}

@Override
public Document call() throws Exception {

File file=new File(path);
if(StringUtils.isNotBlank(path) && new File(path).exists())  {
byte[] array = FileUtils.readFileToByteArray(new File(path));
MultipartFile baseM = new
BASE64DecodedMultipartFile(array,file.getName());
String contenttype=baseM.getContentType();
String prefix = "data:"+contenttype+";base64,";

Document doc= new Document();
String imgurlen=Base64Utils.encodeToString(array);
doc.setImgUrl(prefix+imgurlen);
return doc;
}else {
System.out.println("File Is Not Found");
}
return null;
}

}
```

Add Document Class in your response project

Add this class in your package

```
package com.maan.crm.res;


import java.io.File;


import com.fasterxml.jackson.annotation.JsonProperty;


import lombok.AllArgsConstructor;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;
```

```
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class Document {

@JsonProperty("ProductId")
private String productId;
@JsonProperty("agencyCode")
private String agencycode;
@JsonProperty("amendId")
private String amendId;
@JsonProperty("apiCheck")
private String apiCheck;
@JsonProperty("apiCheckName")
private String apiCheckName;
@JsonProperty("companyId")
private String companyId;
@JsonProperty("coreAppcode")
private String coreAppcode;
@JsonProperty("displayOrder")
private String displayOrder;
@JsonProperty("docApplicable")
private String docApplicable;
@JsonProperty("documentDesc")
private String documentDesc;
@JsonProperty("documentId")
private String documentId;
@JsonProperty("effectiveDate")
private String effectiveDate;
@JsonProperty("mandatoryStatus")
private String mandatoryStatus;
@JsonProperty("policyType")
private String policyType;
@JsonProperty("remarks")
private String remarks;
@JsonProperty("status")
private String status;
@JsonProperty("imgUrl")
private String imgUrl;
//private File imgUrl;



}
```