存在自己服务器上

1. 上传-修改图片

// 图片上传-修改功能  
@PostMapping("uploadImg")  
 public Map<String, Object> uploadImg(MultipartFile file, HttpServletRequest request) {  
 Map<String, Object> returnDataMap = new HashMap<>();  
  
 // 上传的文件为空  
 if(file.isEmpty()) {  
 returnDataMap.put("result", "上传的文件为空");  
 return returnDataMap;  
 }  
  
 // 前端传过来的图片的名称  
 String originalFilename = file.getOriginalFilename();  
 // 图片后缀扩展名  
 String substring = originalFilename.substring(originalFilename.lastIndexOf("."));  
 // UUID设置为图片的名字，防止重复  
 String uuid = UUID.*randomUUID*().toString();  
 // 图片在服务器上的完整路径（包括磁盘目录, path为磁盘路径：E:/xuxiao/shangyitong/code/static, 读的配置文件里面的）  
 String imgPath = path + "/" + uuid + substring;  
  
  
 // 将图片的完整路径转化成 file 文件  
 File imgFile = new File(imgPath);  
 // 文件目录不存才就创建目录  
 if(!imgFile.getParentFile().exists()) {  
 imgFile.getParentFile().mkdirs();  
 }  
  
 // 获取前端传过来的要删除的图片参数，前端传过来的参数（imgName是图片的名字 比如：01.jpg）， path为磁盘路径：E:/xuxiao/shangyitong/code/static, 读的配置文件里面的  
 String deleteImgName = path + "/" + request.getParameter("imgName");  
 File deleteImgFile = new File(deleteImgName);  
 deleteImgFile.delete();  
  
 // 获取服务器的协议，域名，端口  
 String basePath = request.getScheme() + "//" + request.getServerName() + ":" + request.getServerPort() + request.getContextPath();  
  
 try {  
 // 最关键的一不， 上传文件  
 file.transferTo(imgFile);  
  
 // 拼路径， 给前端返回可以展示的图片  
 String displayImg = basePath + "/uploadFile/img/" + uuid + substring;  
 returnDataMap.put("result", displayImg);  
  
 return returnDataMap;  
 } catch (IOException e) {  
 e.printStackTrace();  
  
 returnDataMap.put("result", "文件上传失败");  
 return returnDataMap;  
 }  
}

增加配置类，将服务器上的图片映射成网络资源

// 配置类将服务器上的图片映射成网络资源  
@Configuration  
public class MyWebAppConfig implements WebMvcConfigurer {  
  
 @Override  
 public void addResourceHandlers(ResourceHandlerRegistry registry) {  
  
 // 前面(addResourceHandler)就是前段请求接口的路径(和上传图片时展示给前端的路径是一致的), 后面(addResourceLocations)就是服务器的磁盘路径( 和配置文件里面的路径是一致的，只是前缀加上 file: )  
 registry.addResourceHandler("uploadFile/img/\*\*").addResourceLocations("file:E:/xuxiao/shangyitong/code/static/");  
  
 }  
}

二、删除图片

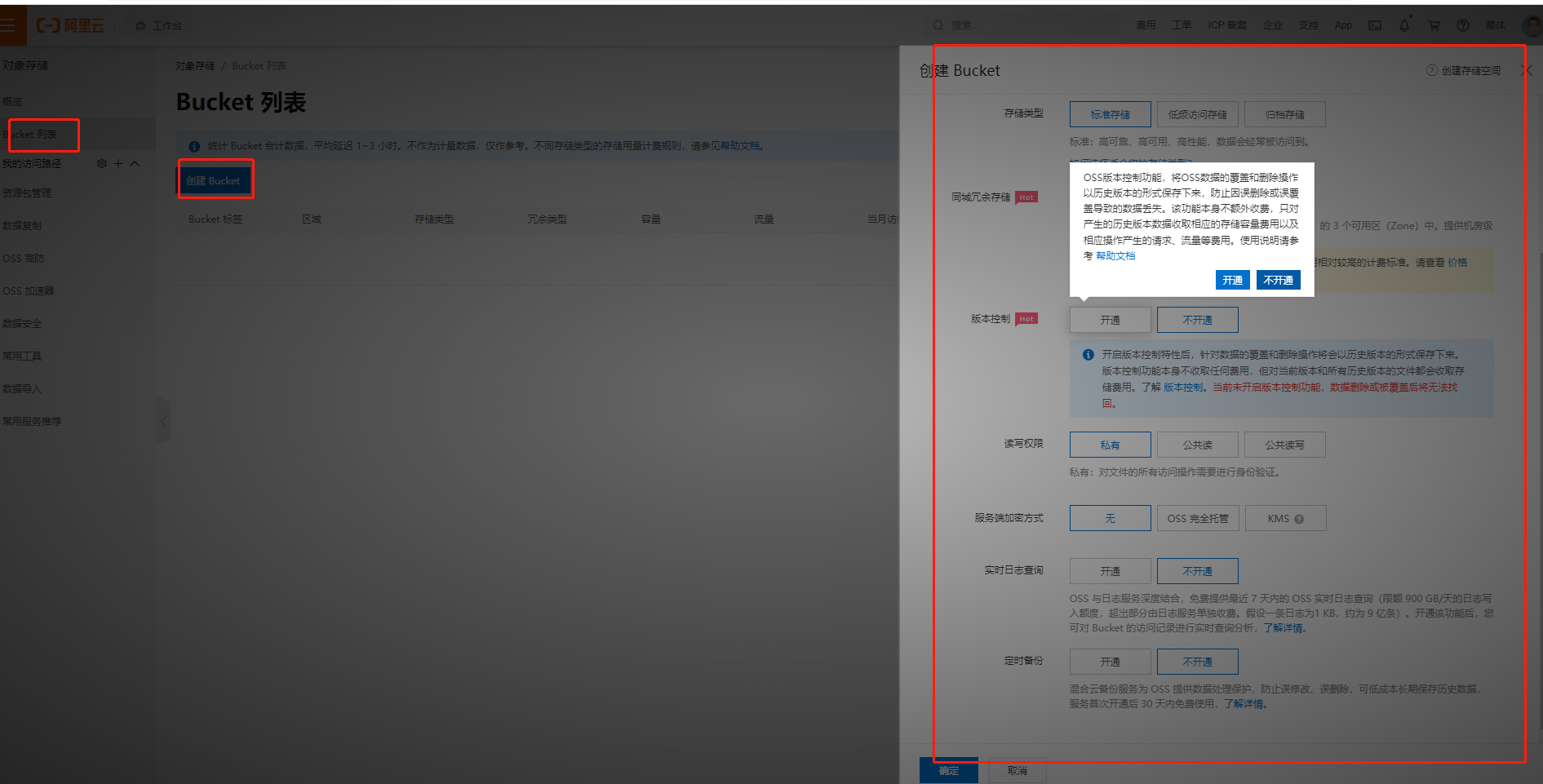
// 删除图片  
@GetMapping("/deleteImg")  
public String deleteImg(String imgName) {  
 try {  
 // path为磁盘路径：E:/xuxiao/shangyitong/code/static, 读的配置文件里面的  
 String fullpath = path + "/" + imgName;  
 File file = new File(fullpath);  
 if (file.delete()) {  
 return "删除成功";  
 } else {  
 return "删除的图片不存在";  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 return "异常";  
 }  
}

阿里云oss 云存储对象

1. 上阿里云官网https://www.aliyun.com/ 搜索 对象存储oss 跳转到



1. 开通之后创建bucket



1. bucket创建之后是上传文件



以上是通过控制台上传的， 一般都是用代码上传文件（以下就是用代码上传文件）

官方文档：https://help.aliyun.com/document\_detail/32010.html

1. 引入依赖

<dependency>

<groupId>com.aliyun.oss</groupId>

<artifactId>aliyun-sdk-oss</artifactId>

<version>3.10.2</version>

</dependency>

1. 创建bucket，上传文件

// yourEndpoint填写Bucket所在地域对应的Endpoint。以华东1（杭州）为例，Endpoint填写为https://oss-cn-hangzhou.aliyuncs.com。

String endpoint = "yourEndpoint";

// 阿里云账号AccessKey拥有所有API的访问权限，风险很高。强烈建议您创建并使用RAM用户进行API访问或日常运维，请登录RAM控制台创建RAM用户。

String accessKeyId = "yourAccessKeyId";

String accessKeySecret = "yourAccessKeySecret";

// 创建OSSClient实例。

OSS ossClient = new OSSClientBuilder().build(endpoint, accessKeyId, accessKeySecret);

// 上传文件

ossClient.putObject("examplebucket", "exampledir/exampleobject.png", inputStream);

// 关闭OSSClient。

ossClient.shutdown()