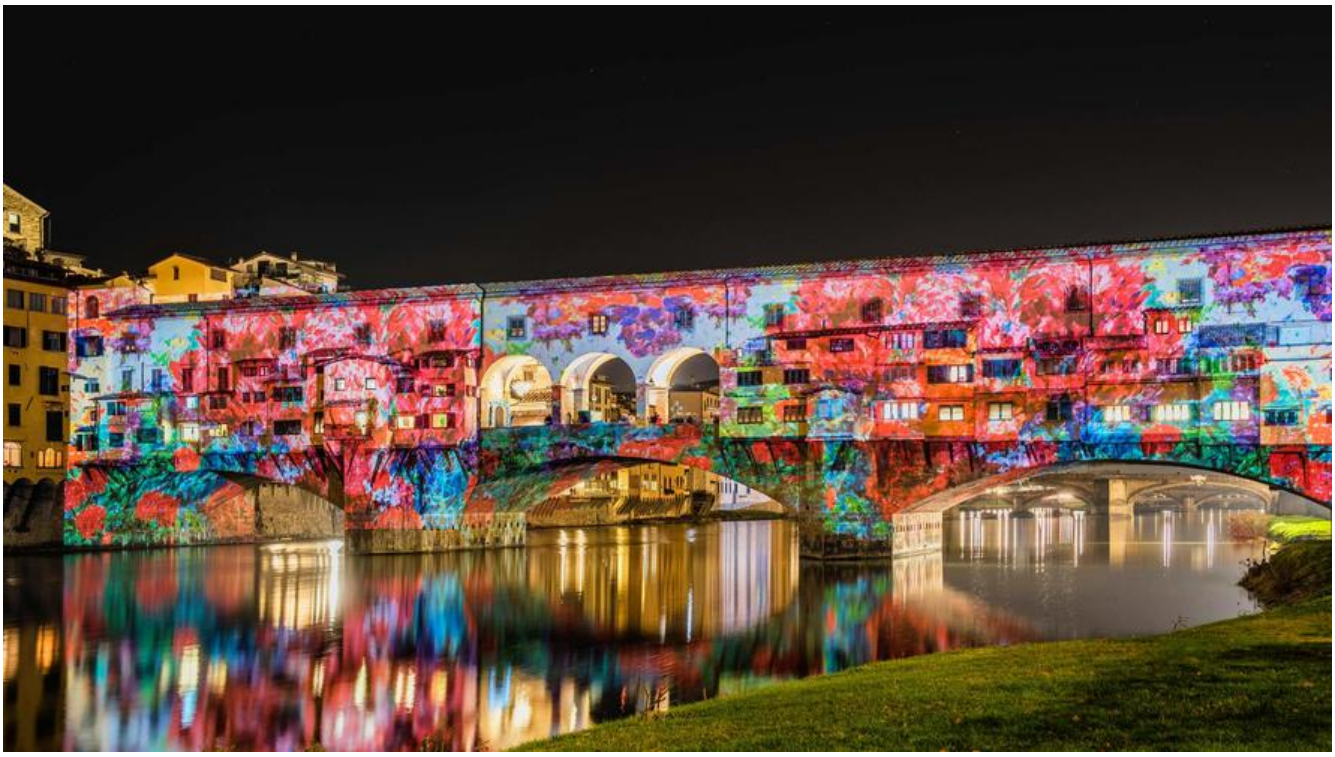
# 利用 vue 实现 java 后端视频点播（阿里云 OSS 上传）

作者：dakoukou
原文链接：https://ld246.com/article/1575020683048
来源网站：链滴



前几天做一个视频点播的功能点，遇到了一些困难，索性写一个方便大家学习。

---

**一 前端上传**

  前端用的是 **vue**，组件库是Element， 上传是其中的**el_upoad**

  先写个框框来上传视频

```
 <el-upload
        ref="uploadVideo"
        :action="uploadVideoUrl"      //上传地址（重要）
        accept='.mp4,.flv'           //限制上传后缀
        :data="uploadVideo"          //上传带的数据（如果只上传视频，可有可无）
        :multiple="false"
        :limit="1"
        :show-file-list="false"
        :before-upload="beforeUploadVideo"          //上传前
        :on-success="handleUploadVideoSuccess"      //上传成功后
        :on-progress="uploadVideoProcess">          //上传中
        <el-button size="small" type="primary" class="uploadButton">上传视频文件</el-button>
      </el-upload>
```

  这里基本就这样写，在 Element(el-upload)中有详细的说明

```
uploadVideoUrl: 后端地址/upload_video     //后端上传视频的接口，后面会写到
```

  前端基本就这样了，就是传个文件到后端

---

## 二 后端处理

  因为是利用的阿里云的视频点播技术，我是 **maven**项目，以需要在**pom.xml**中添加以下内容(都跟aliyun有关):

```
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-core</artifactId>
    <version>3.2.8</version>
</dependency>
<dependency>
    <groupId>com.aliyun.oss</groupId>
    <artifactId>aliyun-sdk-oss</artifactId>
    <version>3.1.0</version>
</dependency>
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-dysmsapi</artifactId>
    <version>1.1.0</version>
</dependency>
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-vod</artifactId>
    <version>2.15.2</version>
</dependency>
```

  再写个关于视频的配置文件，方便后期维护和调取

```
## 阿里云accessKey
accessKeyId= **********
accessKeySecret = **********
#点播服务接入区域
regionId=cn-shanghai
```

  accessKey这东西可以在 阿里云申请获取就行，我记得是头那里

  有了这玩意，用这个在 **初始化**一下，就可以无阻碍的弄他了

```
//注册
public DefaultAcsClient initVodClient() {
    String accessKeyId = SystemConfig.getProperty("accessKeyId");
    String accessKeySecret = SystemConfig.getProperty("accessKeySecret");
    String regionId = SystemConfig.getProperty("regionId");
    DefaultProfile profile = DefaultProfile.getProfile(regionId, accessKeyId, accessKeySecret);
    DefaultAcsClient client = new DefaultAcsClient(profile);
    return client;
}
```

  继续，可以上传文件了，就是你前端上传的视频,首先是准备上传阶段，通这个接口，你会得到**VideoId**、**UploadAddress**和**UploadAuth**

```
public  CreateUploadVideoResponse createUploadVideo(DefaultAcsClient vodClient,String f
leName) throws ClientException {
```

```
    CreateUploadVideoRequest request = new CreateUploadVideoRequest();
    request.setFileName(fileName);
    //request.setTitle(title);
    //request.setDescription("this is desc");
    //request.setTags("tag1,tag2");
    //request.setCoverURL("http://vod.aliyun.com/test_cover_url.jpg");
    //request.setCateId(-1L);
    //request.setTemplateGroupId("");
    //request.setWorkflowId("");
    //request.setStorageLocation("");
    //request.setAppId("app-1000000");
    //设置请求超时时间
    return vodClient.getAcsResponse(request);
}
```

  在这之前，我们可以对获取到的 **UploadAddress**和**Uploa**
**Auth** 进行**编码**处理

```
JSONObject uploadAuth = JSONObject.parseObject(decodeBase64(createUploadVideoRespo
se.getUploadAuth()));
JSONObject uploadAddress = JSONObject.parseObject(decodeBase64(createUploadVideoRe
ponse.getUploadAddress()));
```

  附上方法：

```
public  String decodeBase64(String s) {
    byte[] b = null;
    String result = null;
    if (s != null) {
        Base64 decoder = new Base64();
        try {
            b = decoder.decode(s);
            result = new String(b, "utf-8");
        } catch (Exception e) {
        }
    }
    return result;
}
```

  有了 **UploadAddress**和**UploadAuth** 后，我们就可以**初**
**化一个OSS客户端**来上传视频文件

```
public  OSSClient initOssClient(JSONObject uploadAuth, JSONObject uploadAddress) {
    String endpoint = uploadAddress.getString("Endpoint");
    String accessKeyId = uploadAuth.getString("AccessKeyId");
    String accessKeySecret = uploadAuth.getString("AccessKeySecret");
    String securityToken = uploadAuth.getString("SecurityToken");
    return new OSSClient(endpoint, accessKeyId, accessKeySecret, securityToken);
}
```

  接下来，就是最主要的功能了，终于到了上传文件环节（在这之前，把文件
型转换成File,进度条可以正确显示百分比）

```
public  void uploadLocalFile(OSSClient ossClient, JSONObject uploadAddress, File  file, Http
ervletRequest request) throws FileNotFoundException {
```

```
        String bucketName = uploadAddress.getString("Bucket");
        String objectName = uploadAddress.getString("FileName");
        ossClient.putObject(new PutObjectRequest(bucketName, objectName,file).
                    <PutObjectRequest>withProgressListener(new PutObjectProgressListener(req
est.getSession())));
    }
```

  进度条是参考这个 老哥的，但是前后端session不一致，还啥原因，无法传递session的值，导致无法传到前端，这还没解决。。。

  最后写一个整合的 方便查看

```
 @RequestMapping("upload_video")  //前端的那个上传地址`uploadVideoUrl`
 @ResponseBody
 public ReturnObject uploadVideo(@RequestBody MultipartFile file,HttpServletRequest reque
t) {
        HashMap<String, String> hashmap = new HashMap<String, String>();
        if (file != null) {
            String fileName = file.getOriginalFilename();// 原文件名称
            String trueFileName = String.valueOf(System.currentTimeMillis()) + fileName;
            //先初始化
            DefaultAcsClient vodClient = vodService.initVodClient();
            try {
                //转换成File
                InputStream inputStream = file.getInputStream();
                File f =new File(fileName);
                this.inputStreamToFile(inputStream, f);
                //准备上传阶段
                CreateUploadVideoResponse createUploadVideoResponse = vodService.createUplo
dVideo(vodClient, trueFileName);
                //转化一下
                JSONObject uploadAuth = JSONObject.parseObject(vodService.decodeBase64(crea
eUploadVideoResponse.getUploadAuth()));
                JSONObject uploadAddress = JSONObject.parseObject(vodService.decodeBase64(c
eateUploadVideoResponse.getUploadAddress()));
                // 使用UploadAuth和UploadAddress初始化OSS客户端
                OSSClient ossClient = vodService.initOssClient(uploadAuth, uploadAddress);
                // 上传文件
                vodService.uploadLocalFile(ossClient, uploadAddress, f,request);
               //最后，你可以对这个videoId,进行保存到自个的数据库里的操作， 方便后面通过Id获取视
地址进行视频点播操作
                 String videoId = createUploadVideoResponse.getVideoId();
            } catch (ClientException e) {
                logger.error("视频文件上传错误("+e.getLocalizedMessage()+")");
                hashmap.put("Put local file fail, ErrorMessage :", e.getLocalizedMessage());
            } catch (FileNotFoundException e) {
                logger.error("视频文件上传错误("+e.getLocalizedMessage()+")");
                hashmap.put("Put FileNotFoundException :", e.getLocalizedMessage());
            } catch (IOException e) {
                e.printStackTrace();
            }
            return ReturnObject.ok("上传视频成功").data(hashmap);
        } else {
            logger.error("上传音/视频失败失败：file为NULL");
```

```
            return ReturnObject.ok("上传视频失败，文件为空");
        }
    }
```

  这样基本就结束了，整个上传过程，**videoId**很重要

---

### 三 前端播放

  当你需要播放上传的视频的视频，就需要那个最重要的**videoId**(上传视频会返回一个**videoId**)的帮助了 ，我们可以通过videoId直接获取到视频播放地址

  通过这个接口，获取传递的**videoId**的播放地址

```
  public GetPlayInfoResponse getPlayInfo(DefaultAcsClient client, String videoId) throws Clien
Exception {
        GetPlayInfoRequest request = new GetPlayInfoRequest();
        request.setVideoId(videoId);
        return client.getAcsResponse(request);
    }
```

  接下来就写一个**整合**的，方便查看

```
@GetMapping("get_url")
@ResponseBody
public  ReturnObject getPlayUrl (String videoId)  {
        //先初始化
        DefaultAcsClient client = vodService.initVodClient();
        GetPlayInfoResponse response = new GetPlayInfoResponse();
        HashMap<String,String> hashmap =  new HashMap<String,String>();
        try {
          //获取videoId对应的Info
            response = vodService.getPlayInfo(client,videoId);
            List<GetPlayInfoResponse.PlayInfo> playInfoList = response.getPlayInfoList();
            for (GetPlayInfoResponse.PlayInfo playInfo : playInfoList) {
                //获取播放地址
                hashmap.put("PlayUrl", playInfo.getPlayURL());
                hashmap.put("Duration", playInfo.getDuration());
            }
        } catch (Exception e) {
            logger.error("获取播放地址错误("+e.getLocalizedMessage()+")");
            hashmap.put("ErrorMessage", e.getLocalizedMessage());
        }
        return  ReturnObject.ok("获取播放地址成功").data(hashmap);
    }
```

  最后前端就得到这个**PlayUrl**，这就是传说中的播放地址，
频点播也结束了。

  希望，互相学习，大家都得到进步