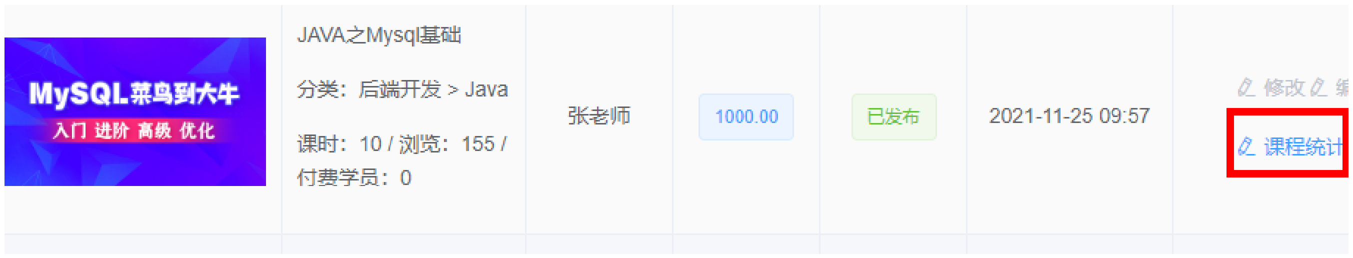
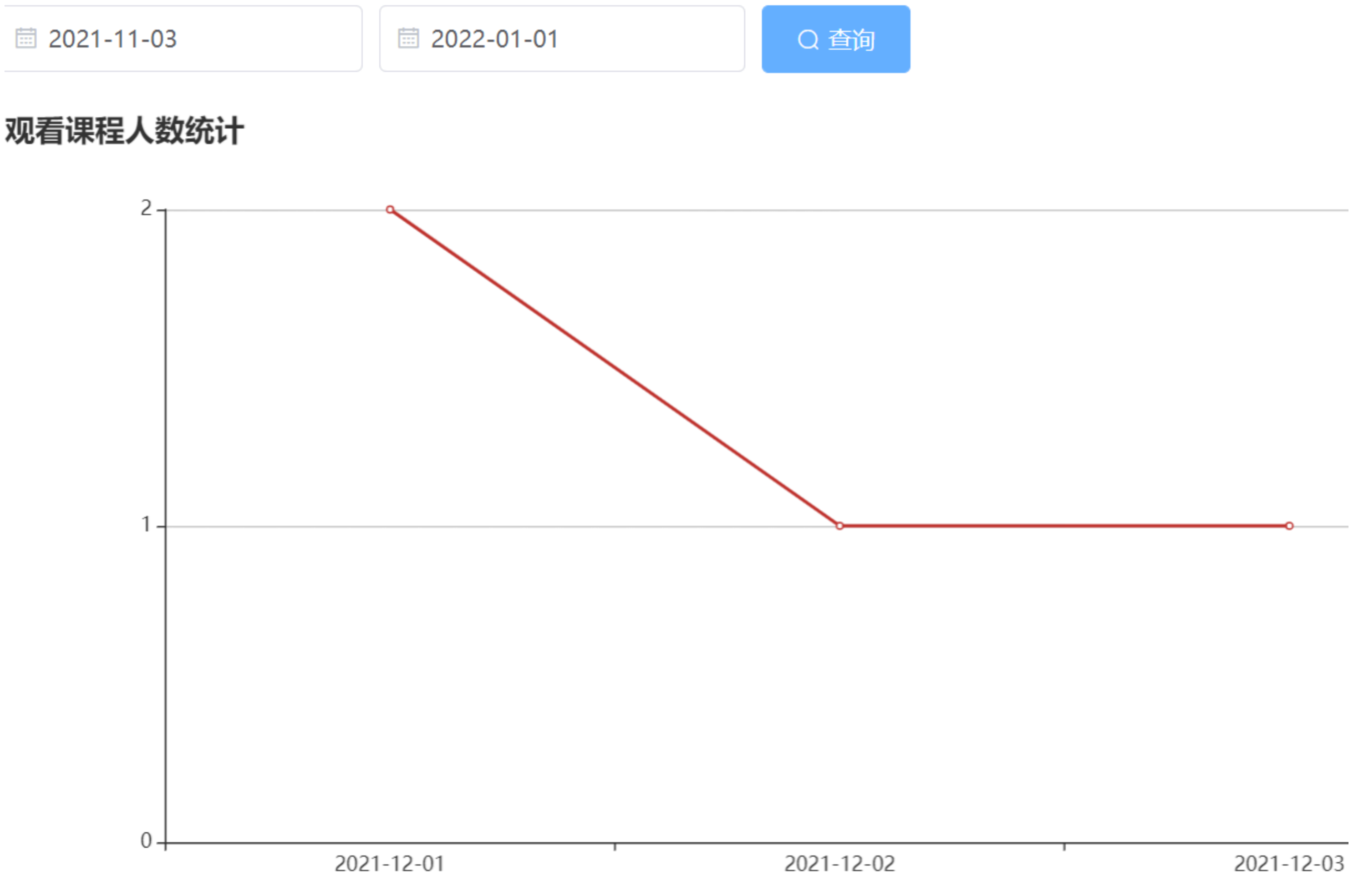
硅谷课堂第九天-点播管理模块（三）

一、点播管理模块-课程统计

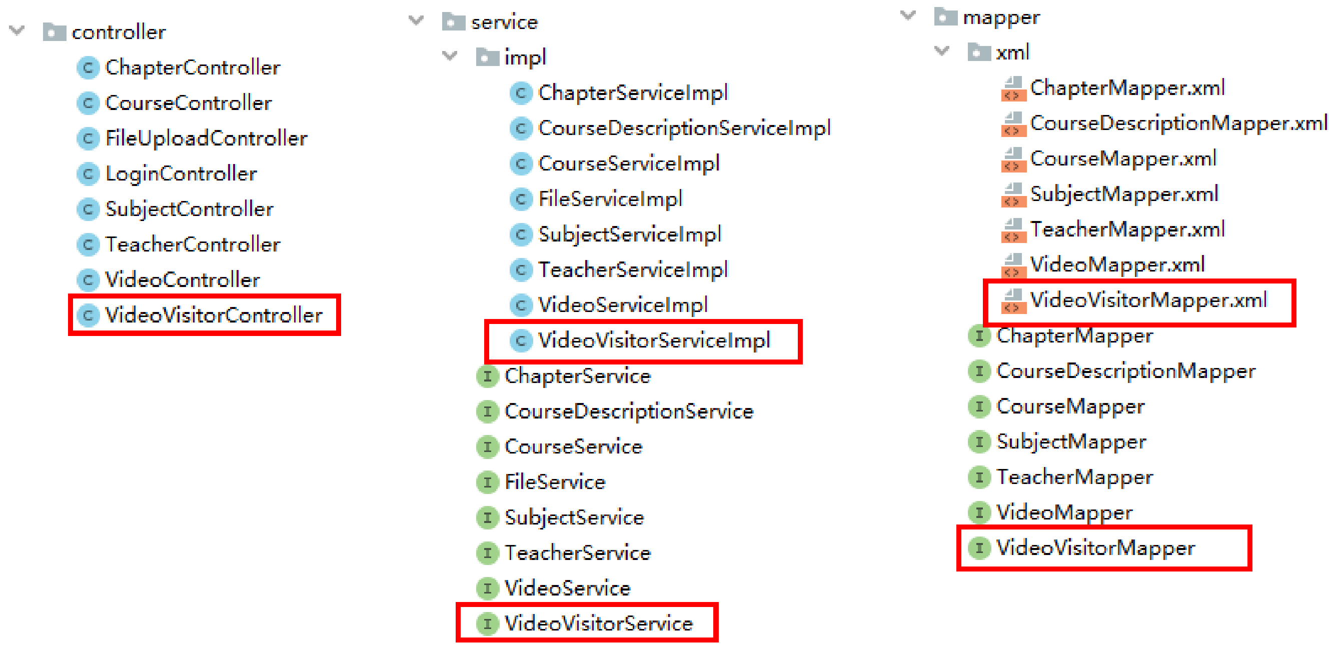
1、课程统计需求


JAVA之Mysql基础
分类：后端开发 > Java
课时：10 / 浏览：155 / 付费学员：0
张老师
1000.00
已发布
2021-11-25 09:57



2、课程统计接口

2.1、创建相关代码



2.2、编写 Controller

VideoVisitorController

```
@Api(value = "VideoVisitor管理", tags = "VideoVisitor管理")
@RestController
@RequestMapping(value="/admin/vod/videoVisitor")
@CrossOrigin
public class VideoVisitorController {

    @Autowired
    private VideoVisitorService videoVisitorService;

    @ApiOperation("显示统计数据")
    @GetMapping("findCount/{courseId}/{startDate}/{endDate}")
    public Result showChart(
            @ApiParam("开始时间") @PathVariable Long courseId,
            @ApiParam("开始时间") @PathVariable String startDate,
            @ApiParam("结束时间") @PathVariable String endDate){

        Map<String, Object> map = videoVisitorService.findCount(courseId, startDate, endDate);
        return Result.ok(map);
    }
}
```

2.3、编写 Service 和实现

VideoVisitorService 和 VideoVisitorServiceImpl

```
@Service
public class VideoVisitorServiceImpl extends ServiceImpl<VideoVisitorMapper, VideoVisitor> impleme

    //课程统计的接口
    @Override
    public Map<String, Object> findCount(Long courseId, String startDate, String endDate) {
        //调用mapper的方法
        List<VideoVisitorCountVo> videoVisitorVoList =
                baseMapper.findCount(courseId,startDate,endDate);
        //创建map集合
        Map<String, Object> map = new HashMap<>();
        //创建两个list集合，一个代表所有日期，一个代表日期对应数量
        //封装数据，代表所有日期
        List<String> dateList =
                videoVisitorVoList.stream().map(VideoVisitorCountVo::getJoinTime).
                               collect(Collectors.toList());
        //代表日期对应数量
        List<Integer> countList = videoVisitorVoList.stream().map(VideoVisitorCountVo::getUserCou
                                .collect(Collectors.toList());
        //放到map集合
        map.put("xData", dateList);
        map.put("yData", countList);
        return map;
    }
}
```

2.4、编写 Mapper

（1）VideoVisitorMapper

```
public interface VideoVisitorMapper extends BaseMapper<VideoVisitor> {
    ////显示统计数据
    List<VideoVisitorCountVo> findCount(
            @Param("courseId") Long courseId,
            @Param("startDate") String startDate,
            @Param("endDate") String endDate);
}
```

（2）**VideoVisitorMapper.xml 文件**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-ma
<mapper namespace="com.atguigu.ggkt.vod.mapper.VideoVisitorMapper">
    <select id="findCount" resultType="com.atguigu.ggkt.vo.vod.VideoVisitorCountVo">
        SELECT
            DATE(join_time) AS joinTime,
            COUNT(*) AS userCount
        FROM video_visitor
        <where>
            <if test="startDate != null and startDate != ''">
                AND DATE(join_time) &gt;= #{startDate}
            </if>
            <if test="endDate != null and endDate != ''">
                AND DATE(join_time) &lt;= #{endDate}
            </if>
            and course_id=#{courseId}
        </where>
        GROUP BY DATE(join_time)
        ORDER BY DATE(join_time)
    </select>
</mapper>
```

3、课程统计前端

3.1、定义接口

创建 videoVisitor.js 定义接口

```
import request from "@/utils/request";

const api_name = "/admin/vod/videoVisitor";

export default {
  findCount(courseId, startDate, endDate) {
    return request({
      url: `${api_name}/findCount/${courseId}/${startDate}/${endDate}`,
      method: "get",
    });
  },
};
```

3.2、安装 ECharts 组件

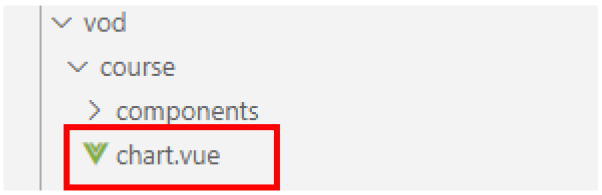
ECharts 是百度的一个项目，后来百度把 Echart 捐给 apache，用于图表展示，提供了常规的折线/面积图、散点（气泡）图、饼图、K 线图，用于统计的箱形图，用于地理数据可视化的地图、热力图、路径图，用于关系数据可视化的关系图、Treemap、旭日图，多维数据可视化的平行坐标图，还有用于 BI 的漏斗图，仪表盘，并且支持图与图之间的混搭。

官方网站：https://echarts.apache.org/zh/index.html

```
npm install --save echarts@4.1.0
```

3.3、编写页面

创建 chart.vue 页面



```
<template>
  <div class="app-container">
    <!--表单-->
    <el-form :inline="true" class="demo-form-inline">
      <el-form-item>
        <el-date-picker
          v-model="startDate"
          type="date"
          placeholder="选择开始日期"
          value-format="yyyy-MM-dd"
        />
      </el-form-item>
      <el-form-item>
        <el-date-picker
          v-model="endDate"
          type="date"
          placeholder="选择截止日期"
          value-format="yyyy-MM-dd"
        />
      </el-form-item>
      <el-button
        :disabled="btnDisabled"
        type="primary"
        icon="el-icon-search"
        @click="showChart()"
        >查询</el-button
      >
    </el-form>
    <div id="chart" class="chart" style="height:500px;" />
  </div>
</template>
<script>
  import echarts from "echarts";
  import api from "@/api/vod/videoVisitor";

  export default {
    data() {
```

```
    return {
      courseId: "",
      startDate: "",
      endDate: "",
      btnDisabled: false,
    };
  },
  created() {
    this.courseId = this.$route.params.id;
    // 初始化最近十天数据
    let currentDate = new Date();
    this.startDate = this.dateFormat(
      new Date(currentDate.getTime() - 7 * 24 * 3600 * 1000)
    );
    this.endDate = this.dateFormat(currentDate);
    this.showChart();
  },
  methods: {
    showChart() {
      api
        .findCount(this.courseId, this.startDate, this.endDate)
        .then((response) => {
          this.setChartData(response.data);
        });
    },
    setChartData(data) {
      // 基于准备好的dom，初始化echarts实例
      var myChart = echarts.init(document.getElementById("chart"));
      // 指定图表的配置项和数据
      var option = {
        title: {
          text: "观看课程人数统计",
        },
        xAxis: {
          data: data.xData,
        },
        yAxis: {
```

```
        minInterval: 1,
      },
      series: [
        {
          type: "line",
          data: data.yData,
        },
      ],
    };
    // 使用刚指定的配置项和数据显示图表。
    myChart.setOption(option);
  },
  dateFormat(date) {
    let fmt = "YYYY-mm-dd";
    let ret;
    const opt = {
      "Y+": date.getFullYear().toString(), // 年
      "m+": (date.getMonth() + 1).toString(), // 月
      "d+": date.getDate().toString(), // 日
      "H+": date.getHours().toString(), // 时
      "M+": date.getMinutes().toString(), // 分
      "S+": date.getSeconds().toString(), // 秒
      // 有其他格式化字符需求可以继续添加，必须转化成字符串
    };
    for (let k in opt) {
      ret = new RegExp("(" + k + ")").exec(fmt);
      if (ret) {
        fmt = fmt.replace(
          ret[1],
          ret[1].length == 1 ? opt[k] : opt[k].padStart(ret[1].length, "0")
        );
      }
    }
    return fmt;
  },
 },
};
```

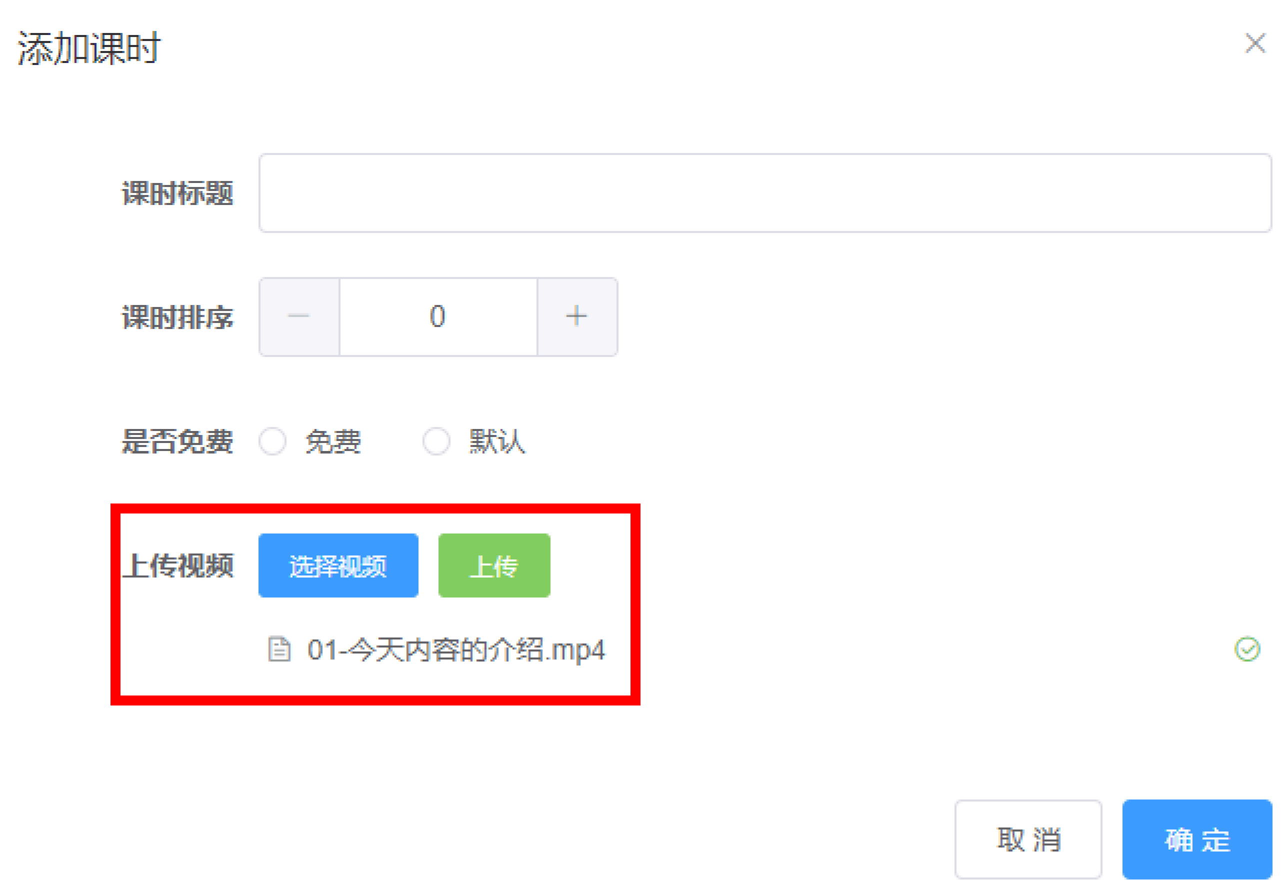
```
</script>
```

二、整合腾讯云点播

1、功能需求介绍

1.1、上传视频

在发布课程时候，需要添加课时并且上传课程视频，这个时候需要使用到腾讯云点播服务进行上传视频管理


添加课时

课时标题

课时排序 0

是否免费 ○ 免费 ○ 默认

上传视频

01-今天内容的介绍.mp4

1.2、删除视频

（1）添加小节，上传课程视频

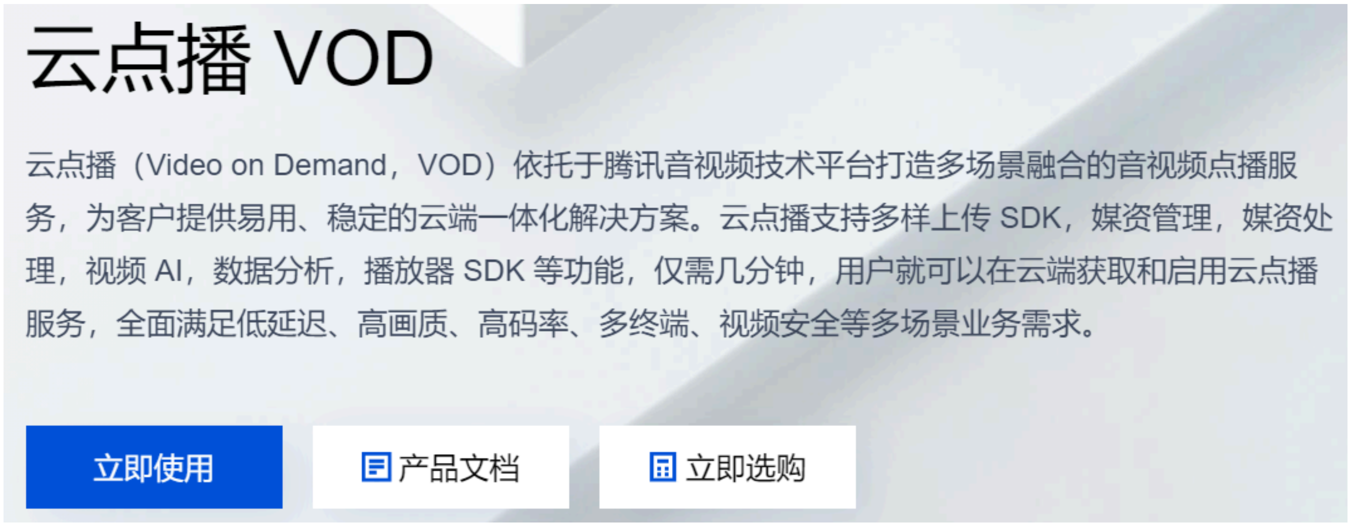
（2）删除小节时候，需要删除视频

（3）删除课程时候，需要删除课程，章节，小节和视频

1.3、视频播放（后续完成）

2、腾讯云点播介绍

腾讯云点播（Video on Demand，VOD）基于腾讯多年技术积累与基础设施建设，为有音视频应用相关需求的客户提供包括音视频存储管理、音视频转码处理、音视频加速播放和音视频通信服务的一站式解决方案。

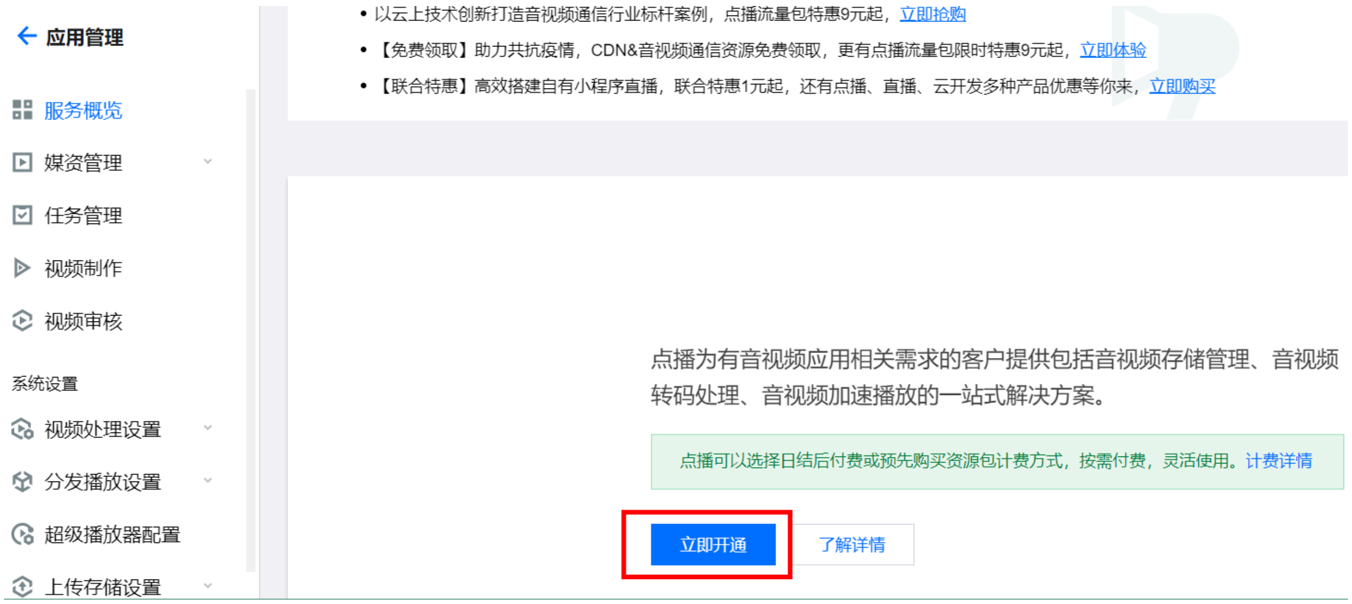


云点播 VOD

云点播（Video on Demand，VOD）依托于腾讯音视频技术平台打造多场景融合的音视频点播服务，为客户提供易用、稳定的云端一体化解决方案。云点播支持多样上传 SDK，媒资管理，媒资处理，视频 AI，数据分析，播放器 SDK 等功能，仅需几分钟，用户就可以在云端获取和启用云点播服务，全面满足低延迟、高画质、高码率、多终端、视频安全等多场景业务需求。

文档中心：https://cloud.tencent.com/document/product/266

2.1、开通"云点播"服务

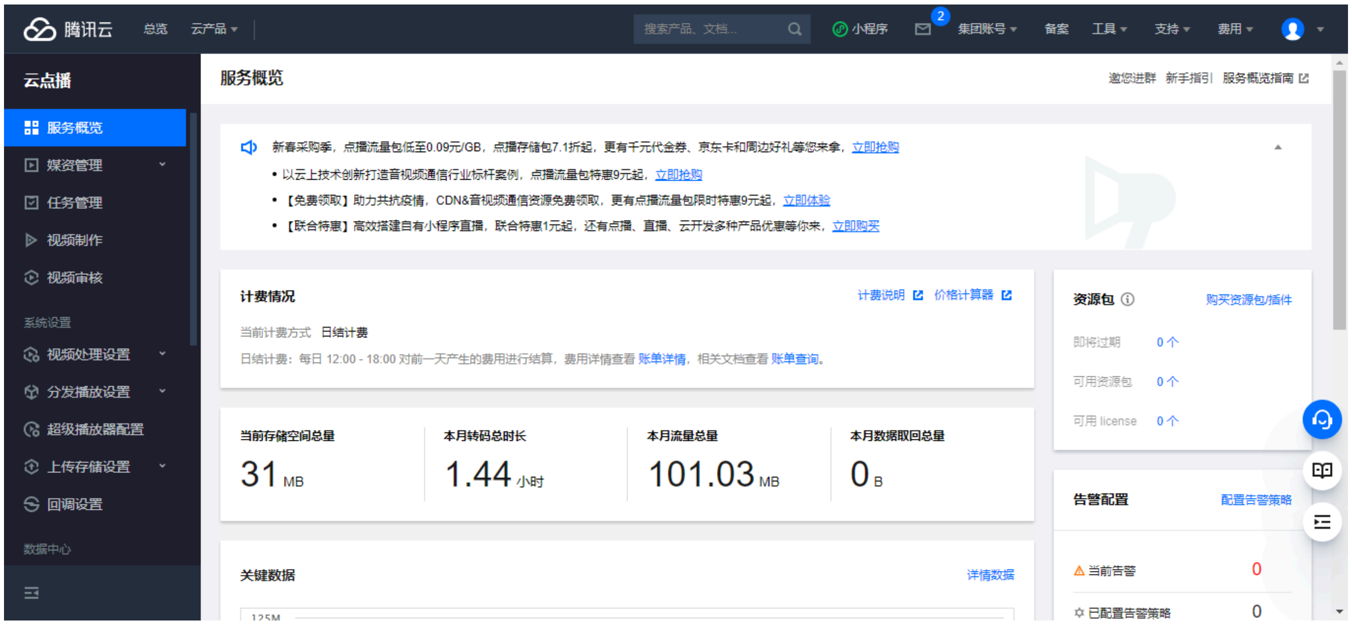


- 以云上技术创新打造音视频通信行业标杆案例，点播流量包特惠9元起，立即抢购
- 【免费领取】助力共抗疫情，CDN&音视频通信资源免费领取，更有点播流量包限时特惠9元起，立即体验
- 【联合特惠】高效搭建自有小程序直播，联合特惠1元起，还有点播、直播、云开发多种产品优惠等你来，立即购买

点播为有音视频应用相关需求的客户提供包括音视频存储管理、音视频转码处理、音视频加速播放的一站式解决方案。

点播可以选择日结后付费或预先购买资源包计费方式，按需付费，灵活使用。计费详情

2.2、管理控制台



服务概览

新春采购季，点播流量包低至0.09元/GB，点播存储包7.1折起，更有千元代金券、京东卡和周边好礼等您来拿，立即抢购
- 以云上技术创新打造音视频通信行业标杆案例，点播流量包特惠9元起，立即抢购
- 【免费领取】助力共抗疫情，CDN&音视频通信资源免费领取，更有点播流量包限时特惠9元起，立即体验
- 【联合特惠】高效搭建自有小程序直播，联合特惠1元起，还有点播、直播、云开发多种产品优惠等你来，立即购买

计费情况　计费说明　价格计算器

当前计费方式　日结计费

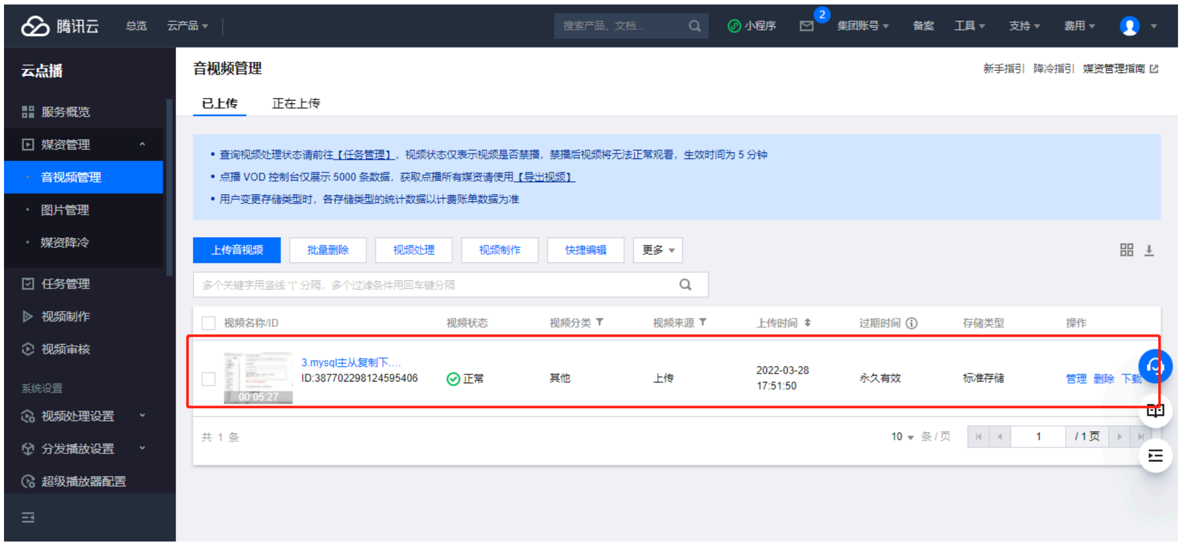
日结计费：每日 12:00 - 18:00 对前一天产生的费用进行结算，费用详情查看 账单详情，相关文档查看 账单查询。

当前存储空间总量	本月转码总时长	本月流量总量	本月数据取回总量
31 MB	1.44 小时	101.03 MB	0 B

2.3、上传视频

上传视频可将视频上传到云点播的存储中，以进行后续的处理和分发等。

- 单击左侧菜单栏【媒资管理 > 视频管理】，默认展示【已上传】标签页；
- 点击【上传视频】按钮；
- 单击【选择视频】，选择本地视频文件；
- 单击【开始上传】；
- 页面将自动跳转至【正在上传】标签页，本地文件所在行【状态】栏为“上传成功”时，单击【已上传】标签页，可见完成上传的视频；



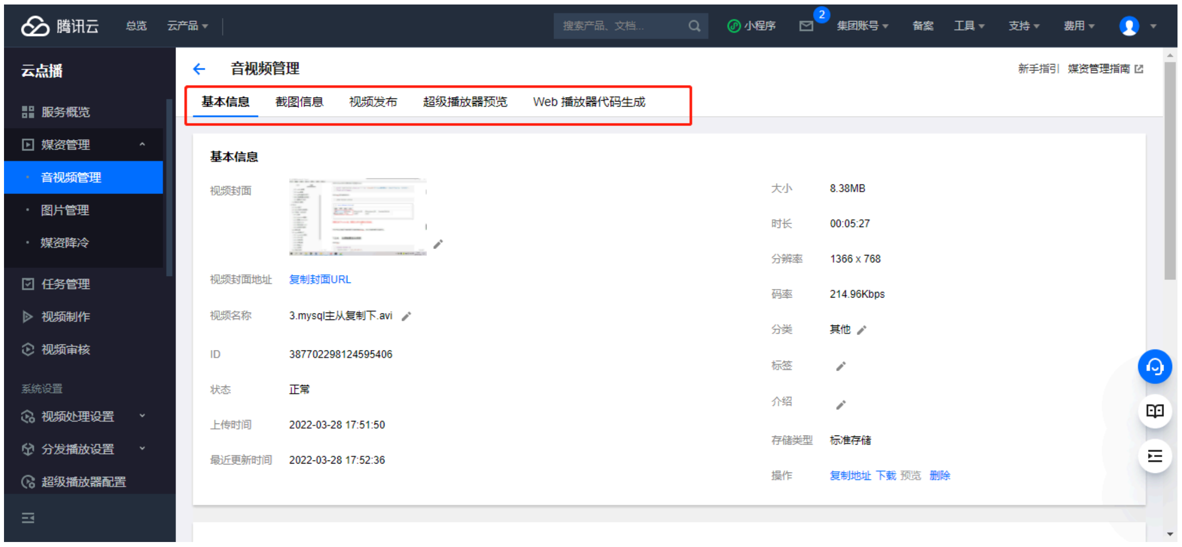
音视频管理

- 查询视频处理状态请前往【任务管理】，视频状态仅表示视频是否禁播，禁播后视频将无法正常观看，生效时间为 5 分钟
- 点播 VOD 控制台仅展示 5000 条数据，获取点播所有媒资请使用【导出视频】
- 用户变更存储类型时，各存储类型的统计数据以计费账单数据为准

视频名称/ID	视频状态	视频分类	视频来源	上传时间	过期时间	存储类型	操作
3.mysql主从复制下.... ID:387702298124595406 (00:05:27)	正常	其他	上传	2022-03-28 17:51:50	永久有效	标准存储	管理 删除 下载

共 1 条　10 条/页　1　/ 1 页

- 单击【管理】，可以查看视频详情。



音视频管理

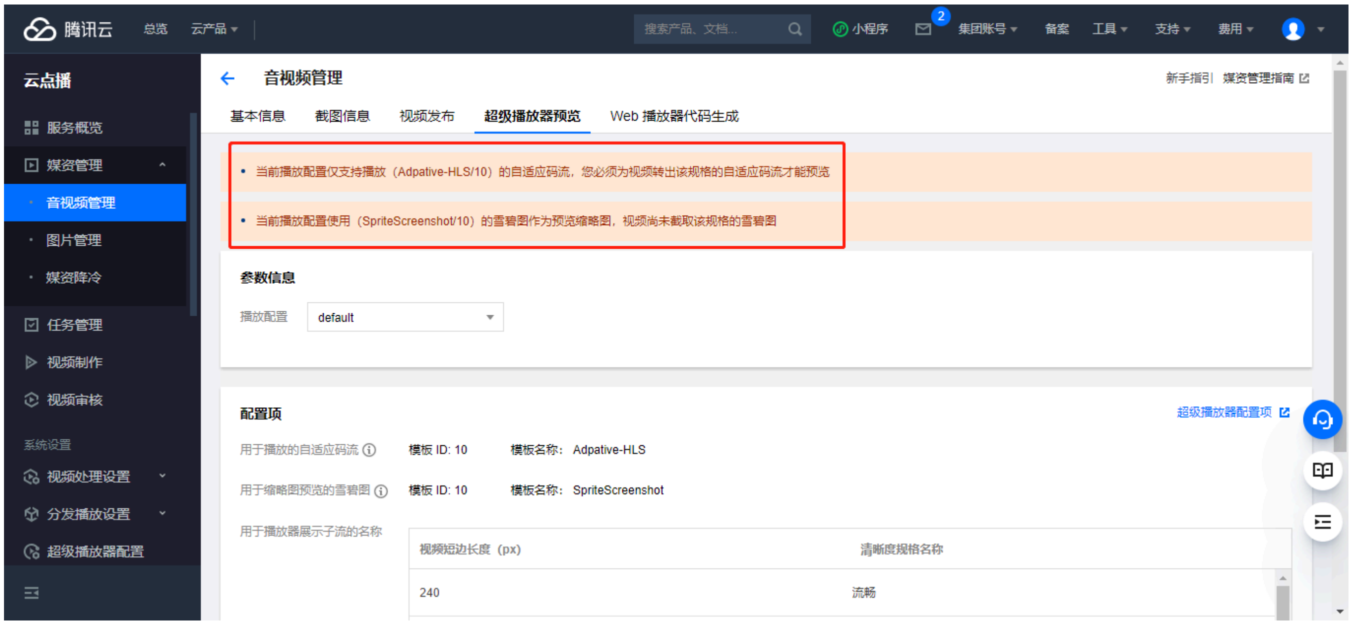
基本信息

视频封面		大小	8.38MB
		时长	00:05:27
		分辨率	1366 x 768
视频封面地址	复制封面URL	码率	214.96Kbps
视频名称	3.mysql主从复制下.avi	分类	其他
ID	387702298124595406	标签	
状态	正常	介绍	
上传时间	2022-03-28 17:51:50	存储类型	标准存储
最近更新时间	2022-03-28 17:52:36	操作	复制地址 下载 预览 删除

2.4、前端集成

前端集成有两种方式，使用“超级播放器预览”与“web 播放器预览”，后者代码已经不更新，推荐使用前者，因此“web 播放器预览”仅做了解。

1、查看“web 播放器预览”



音视频管理

- 当前播放配置仅支持播放（Adpative-HLS/10）的自适应码流，您必须为视频转出该规格的自适应码流才能预览
- 当前播放配置使用（SpriteScreenshot/10）的雪碧图作为预览缩略图，视频尚未截取该规格的雪碧图

参数信息

播放配置 default

配置项

用于播放的自适应码流 模板 ID：10 模板名称：Adpative-HLS

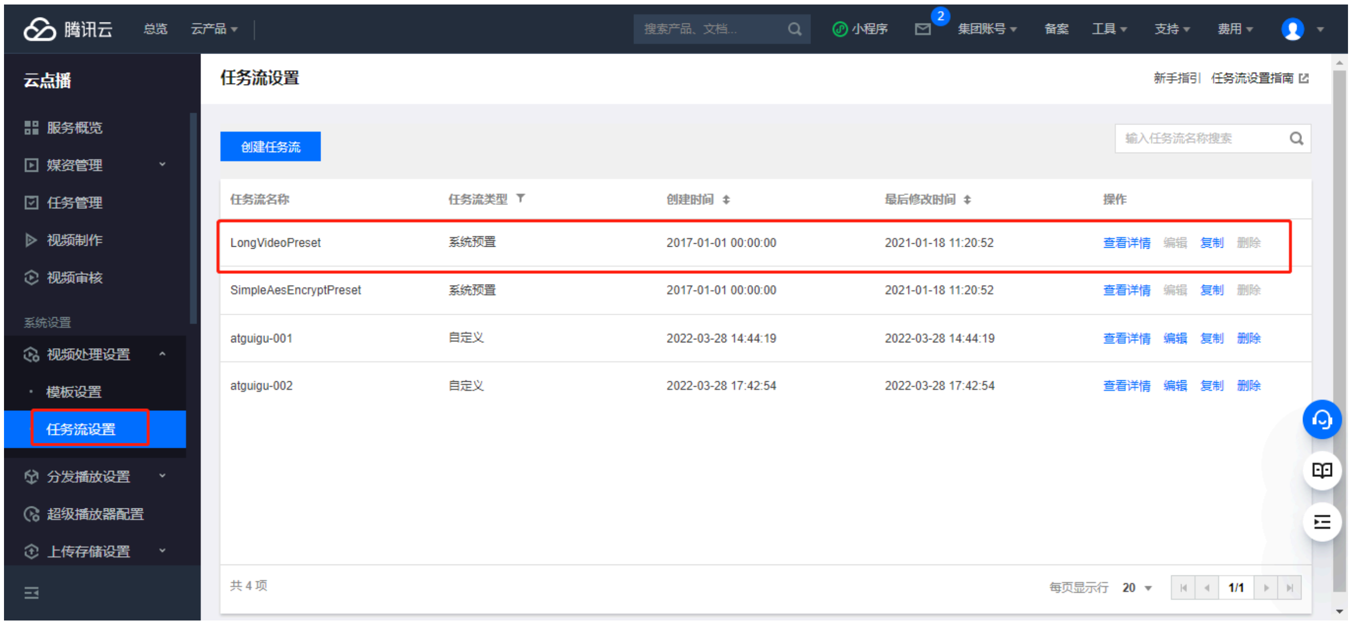
用于缩略图预览的雪碧图 模板 ID：10 模板名称：SpriteScreenshot

用于播放器展示子流的名称

视频短边长度（px）	清晰度规格名称
240	流畅

说明：需要将视频进行转码，才能支持超级播放器播放，转码为：自适应码流

2、查看“任务流设置”

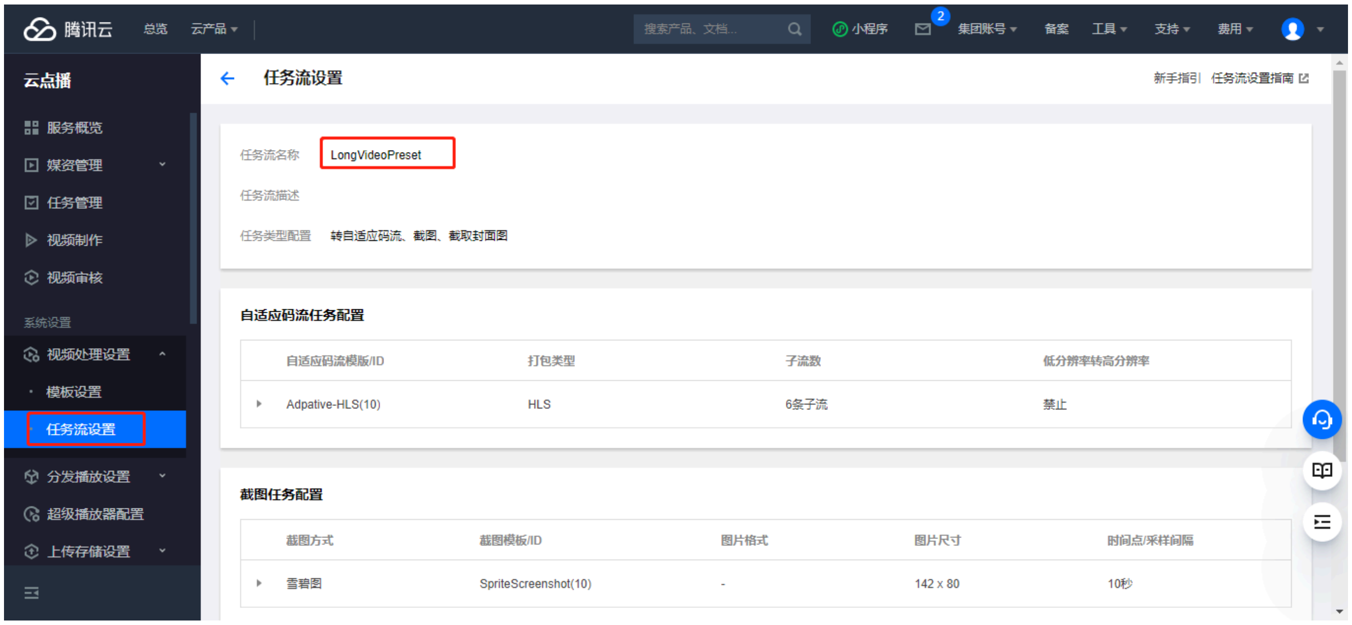


任务流设置

任务流名称	任务流类型	创建时间	最后修改时间	操作
LongVideoPreset	系统预置	2017-01-01 00:00:00	2021-01-18 11:20:52	查看详情 编辑 复制 删除
SimpleAesEncryptPreset	系统预置	2017-01-01 00:00:00	2021-01-18 11:20:52	查看详情 编辑 复制 删除
atguigu-001	自定义	2022-03-28 14:44:19	2022-03-28 14:44:19	查看详情 编辑 复制 删除
atguigu-002	自定义	2022-03-28 17:42:54	2022-03-28 17:42:54	查看详情 编辑 复制 删除

共 4 项　每页显示行 20　1/1

3、查看详情



任务流设置

任务流名称 LongVideoPreset

任务流描述

任务类型配置 转自适应码流、截图、截取封面图

自适应码流任务配置

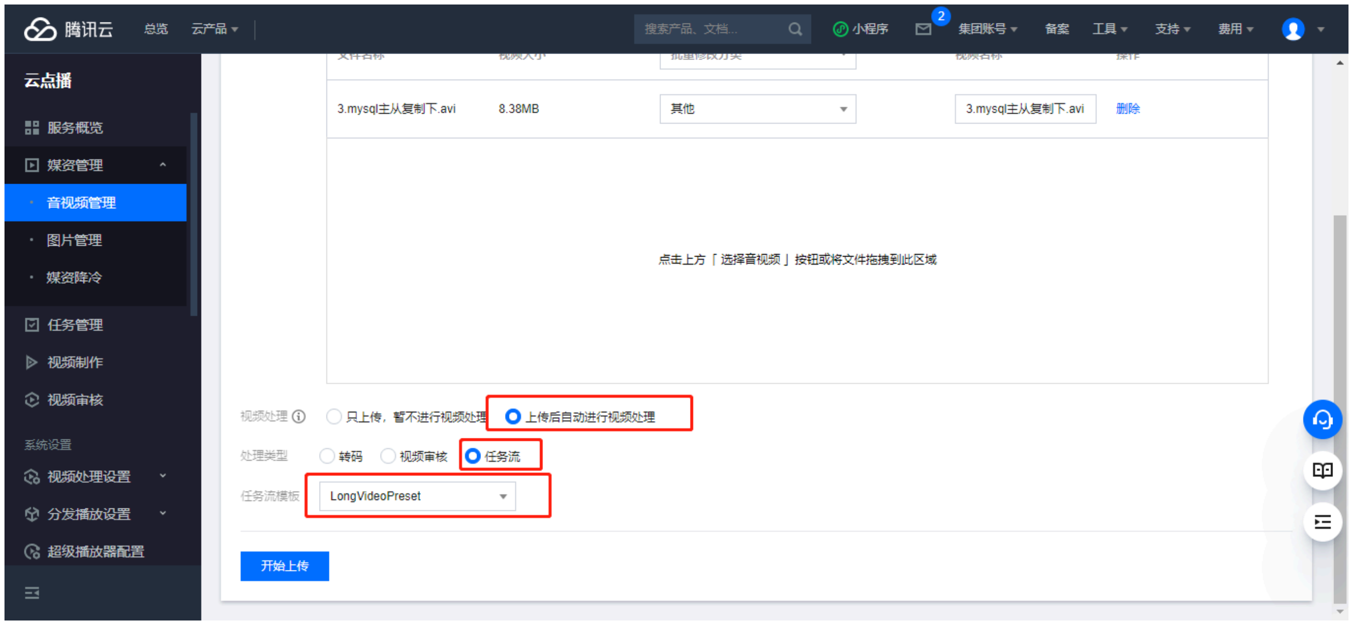
自适应码流模版ID	打包类型	子流数	低分辨率转高分辨率
Adpative-HLS(10)	HLS	6条子流	禁止

截图任务配置

截图方式	截图模板ID	图片格式	图片尺寸	时间点/采样间隔
雪碧图	SpriteScreenshot(10)	-	142 x 80	10秒

当前任务流就是系统默认的“自适应码流”任务流

4、在【音视频管理】重新上传视频



3.mysql主从复制下.avi	8.38MB	其他	3.mysql主从复制下.avi	删除

点击上方「选择音视频」按钮或将文件拖拽到此区域

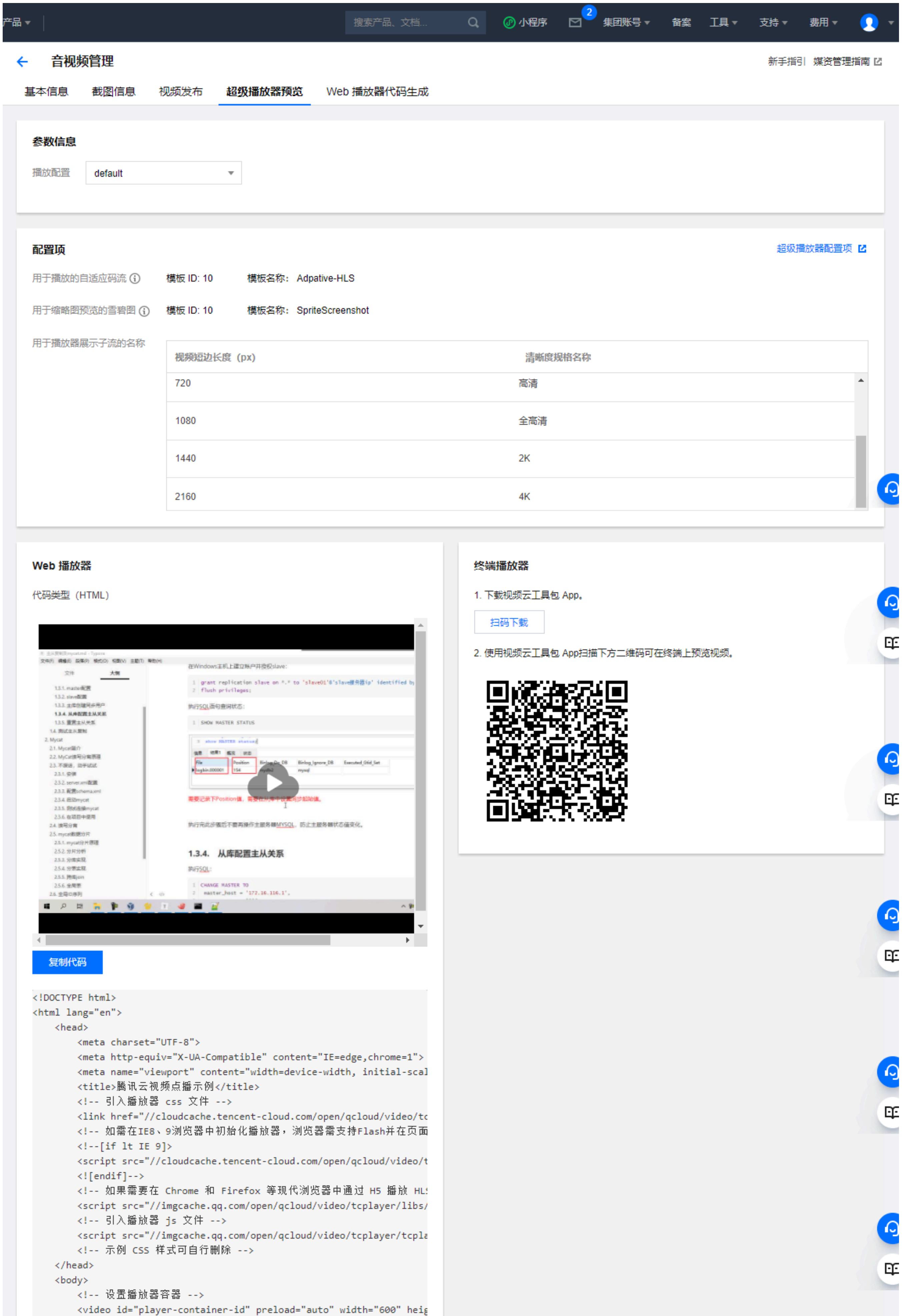
视频处理 ⓘ 只上传，暂不进行视频处理 [✓ 上传后自动进行视频处理]

处理类型 转码 视频审核 [✓ 任务流]

任务流模板 LongVideoPreset

开始上传

5、查看详情



音视频管理

新手指引 | 媒资管理指南

基本信息 | 截图信息 | 视频发布 | **超级播放器预览** | Web 播放器代码生成

参数信息

播放配置: default

配置项

超级播放器配置项

用于播放的自适应码流 | 模板 ID: 10 | 模板名称: Adpative-HLS

用于缩略图预览的雪碧图 | 模板 ID: 10 | 模板名称: SpriteScreenshot

用于播放器展示子流的名称

视频短边长度（px）	清晰度规格名称
720	高清
1080	全高清
1440	2K
2160	4K

Web 播放器

代码类型（HTML）

复制代码

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
        <meta name="viewport" content="width=device-width, initial-scal
        <title>腾讯云视频点播示例</title>
        <!-- 引入播放器 css 文件 -->
        <link href="//cloudcache.tencent-cloud.com/open/qcloud/video/tc
        <!-- 如需在IE8、9浏览器中初始化播放器，浏览器需支持Flash并在页面
        <!--[if lt IE 9]>
        <script src="//cloudcache.tencent-cloud.com/open/qcloud/video/t
        <![endif]-->
        <!-- 如果需要在 Chrome 和 Firefox 等现代浏览器中通过 H5 播放 HL
        <script src="//imgcache.qq.com/open/qcloud/video/tcplayer/libs/
        <!-- 引入播放器 js 文件 -->
        <script src="//imgcache.qq.com/open/qcloud/video/tcplayer/tcpla
        <!-- 示例 CSS 样式可自行删除 -->
    </head>
    <body>
        <!-- 设置播放器容器 -->
        <video id="player-container-id" preload="auto" width="600" heig
```

终端播放器

1. 下载视频云工具包 App。

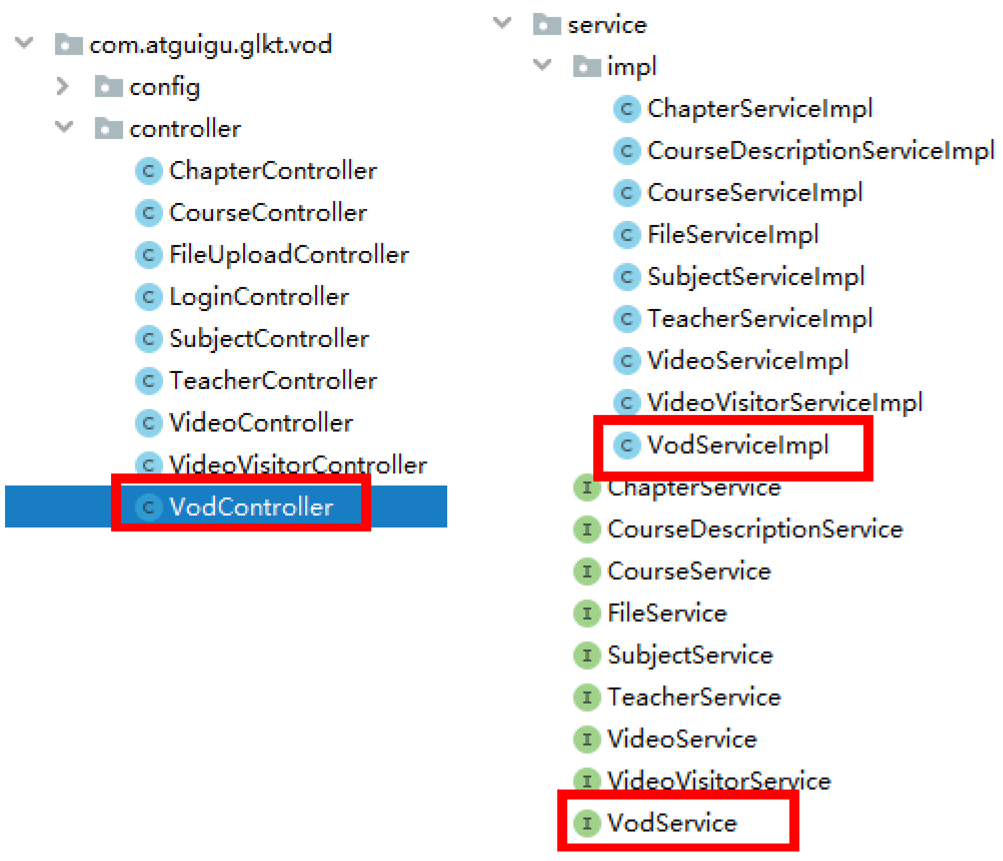
扫码下载

2. 使用视频云工具包 App扫描下方二维码可在终端上预览视频。

6、复制代码 index.html 到项目，即可播放

3、编写视频点播接口

3.1、创建相关类


com.atguigu.glkt.vod
config
controller
ChapterController
CourseController
FileUploadController
LoginController
SubjectController
TeacherController
VideoController
VideoVisitorController
VodController

service
impl
ChapterServiceImpl
CourseDescriptionServiceImpl
CourseServiceImpl
FileServiceImpl
SubjectServiceImpl
TeacherServiceImpl
VideoServiceImpl
VideoVisitorServiceImpl
VodServiceImpl
ChapterService
CourseDescriptionService
CourseService
FileService
SubjectService
TeacherService
VideoService
VideoVisitorService
VodService

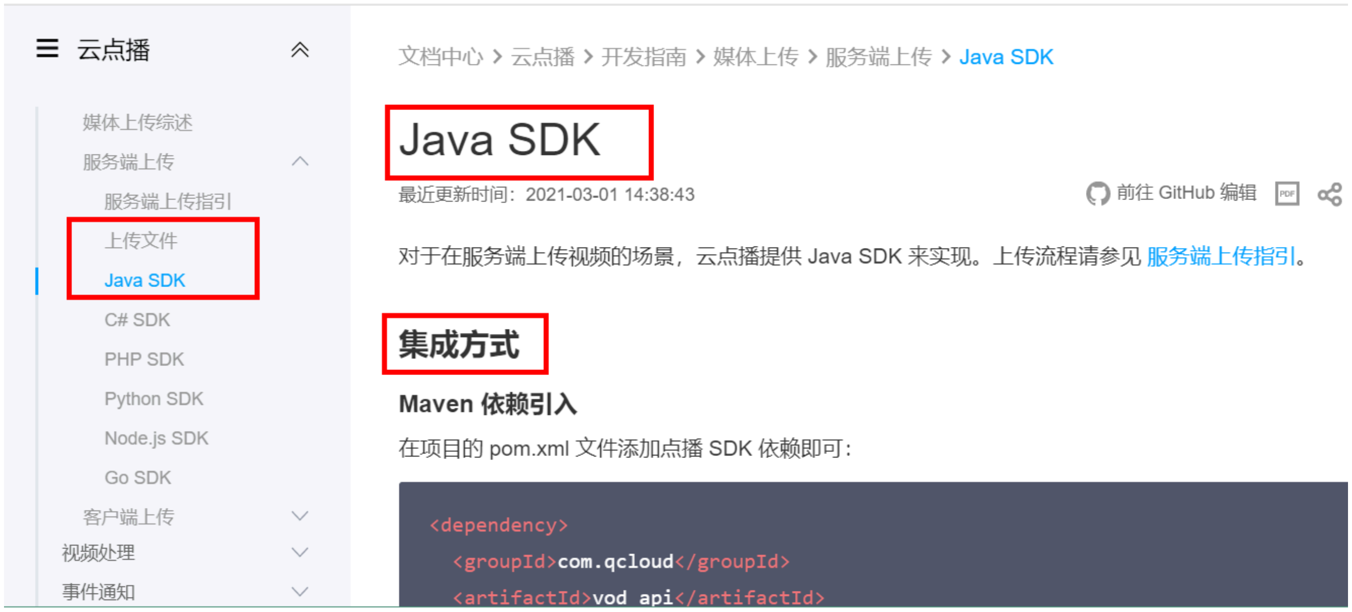
3.2、引入相关依赖

（1）在 service_vod 模块引入

```
<dependency>
    <groupId>com.qcloud</groupId>
    <artifactId>vod_api</artifactId>
    <version>2.1.4</version>
    <exclusions>
        <exclusion>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-log4j12</artifactId>
        </exclusion>
    </exclusions>
</dependency>
```

3.3、编写 Controller

（1）上传视频集成方案： https://cloud.tencent.com/document/product/266/10276



Java SDK

最近更新时间：2021-03-01 14:38:43

对于在服务端上传视频的场景，云点播提供 Java SDK 来实现。上传流程请参见 服务端上传指引。

集成方式

Maven 依赖引入

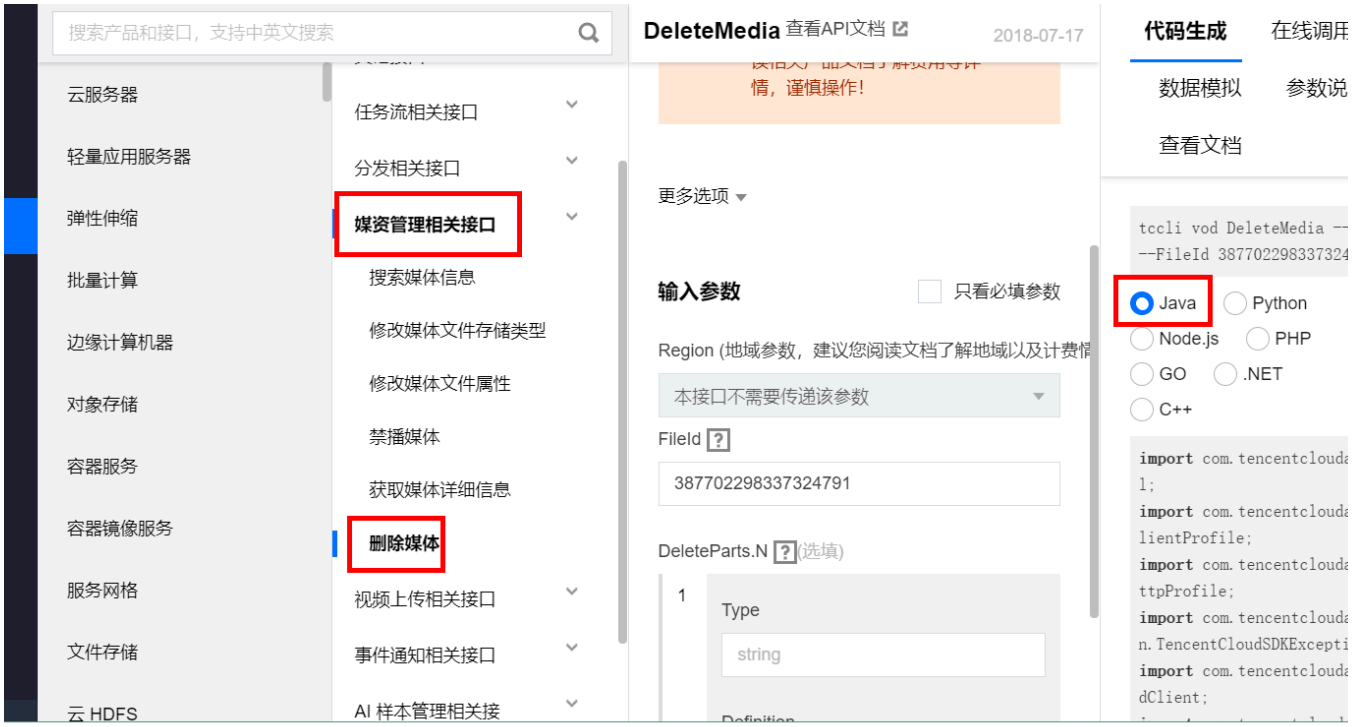
在项目的 pom.xml 文件添加点播 SDK 依赖即可：

```
<dependency>
  <groupId>com.qcloud</groupId>
  <artifactId>vod_api</artifactId>
```

（2）删除视频

可在线生成代码

地址：https://console.cloud.tencent.com/api/explorer?Product=vod&Version=2018-07-17&Action=DeleteMedia



Region (地域参数，建议您阅读文档了解地域以及计费情

本接口不需要传递该参数

FileId: 387702298337324791

DeleteParts.N (选填)

1 Type: string

Definition

```
tccli vod DeleteMedia --
--FileId 387702298337324
```

[✓ Java] Python Node.js PHP GO .NET C++

```
import com.tencentclouda
l;
import com.tencentclouda
lientProfile;
import com.tencentclouda
ttpProfile;
import com.tencentclouda
n.TencentCloudSDKExcepti
import com.tencentclouda
dClient;
```

```java
@Api(tags = "腾讯云点播")
@RestController
@RequestMapping("/admin/vod")
@CrossOrigin
public class VodController {

    @Autowired
    private VodService vodService;

    //上传视频
    @PostMapping("upload")
    public Result uploadVideo(
            @ApiParam(name = "file", value = "文件", required = true)
            @RequestParam("file") MultipartFile file) throws IOException {
        InputStream inputStream = file.getInputStream();
        String originalFilename = file.getOriginalFilename();
        String videoId = vodService.uploadVideo(inputStream, originalFilename);
        return Result.ok(videoId);
    }

    //删除视频
    @DeleteMapping("remove/{videoSourceId}")
    public Result removeVideo( @PathVariable String videoSourceId) {
        vodService.removeVideo(videoSourceId);
        return Result.ok();
    }
}
```

3.4、编写 Service

（1）VodService 定义方法

```java
public interface VodService {
    //上传视频
    String uploadVideo(InputStream inputStream, String originalFilename);
    //删除视频
    void removeVideo(String videoSourceId);
}
```

（2）VodServiceImpl 实现方法

```java
@Service
public class VodServiceImpl implements VodService {

    //上传视频
    @Override
    public String uploadVideo(InputStream inputStream, String originalFilename) {
        try {
            VodUploadClient client =
                    new VodUploadClient(ConstantPropertiesUtil.ACCESS_KEY_ID,
                                        ConstantPropertiesUtil.ACCESS_KEY_SECRET);
            VodUploadRequest request = new VodUploadRequest();
            //视频本地地址
            request.setMediaFilePath("D:\\001.mp4");
            //指定任务流
            request.setProcedure("LongVideoPreset");
            //调用上传方法，传入接入点地域及上传请求。
            VodUploadResponse response = client.upload("ap-guangzhou", request);
            //返回文件id保存到业务表，用于控制视频播放
            String fileId = response.getFileId();
            System.out.println("Upload FileId = {}"+response.getFileId());
            return fileId;
        } catch (Exception e) {
            System.out.println(e.toString());
        }
        return null;
    }

    //删除视频
    @Override
    public void removeVideo(String videoSourceId) {
       try{
            // 实例化一个认证对象，入参需要传入腾讯云账户secretId，secretKey,此处还需注意密钥对的保密
            Credential cred =
                    new Credential(ConstantPropertiesUtil.ACCESS_KEY_ID,
                                   ConstantPropertiesUtil.ACCESS_KEY_SECRET);
            // 实例化要请求产品的client对象,clientProfile是可选的
            VodClient client = new VodClient(cred, "");
            // 实例化一个请求对象,每个接口都会对应一个request对象
            DeleteMediaRequest req = new DeleteMediaRequest();
            req.setFileId(videoSourceId);
            // 返回的resp是一个DeleteMediaResponse的实例，与请求对象对应
            DeleteMediaResponse resp = client.DeleteMedia(req);
            // 输出json格式的字符串回包
            System.out.println(DeleteMediaResponse.toJsonString(resp));
        } catch (TencentCloudSDKException e) {
            System.out.println(e.toString());
        }
    }
}
```

4、完善上传视频功能

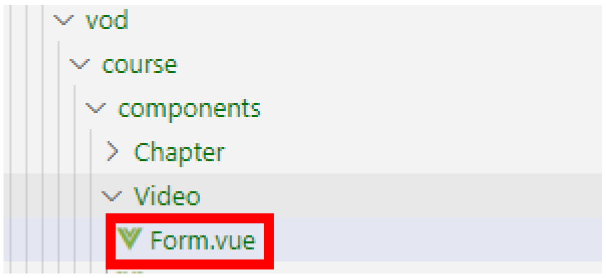
4.1、定义接口

创建 vod.js 定义接口

```
import request from "@/utils/request";

export default {
  //删除视频
  removeByVodId(id) {
    return request({
      url: `/admin/vod/remove/${id}`,
      method: "delete",
    });
  },
};
```

4.1、添加上传视频

（1）修改 Video/Form.vue 页面

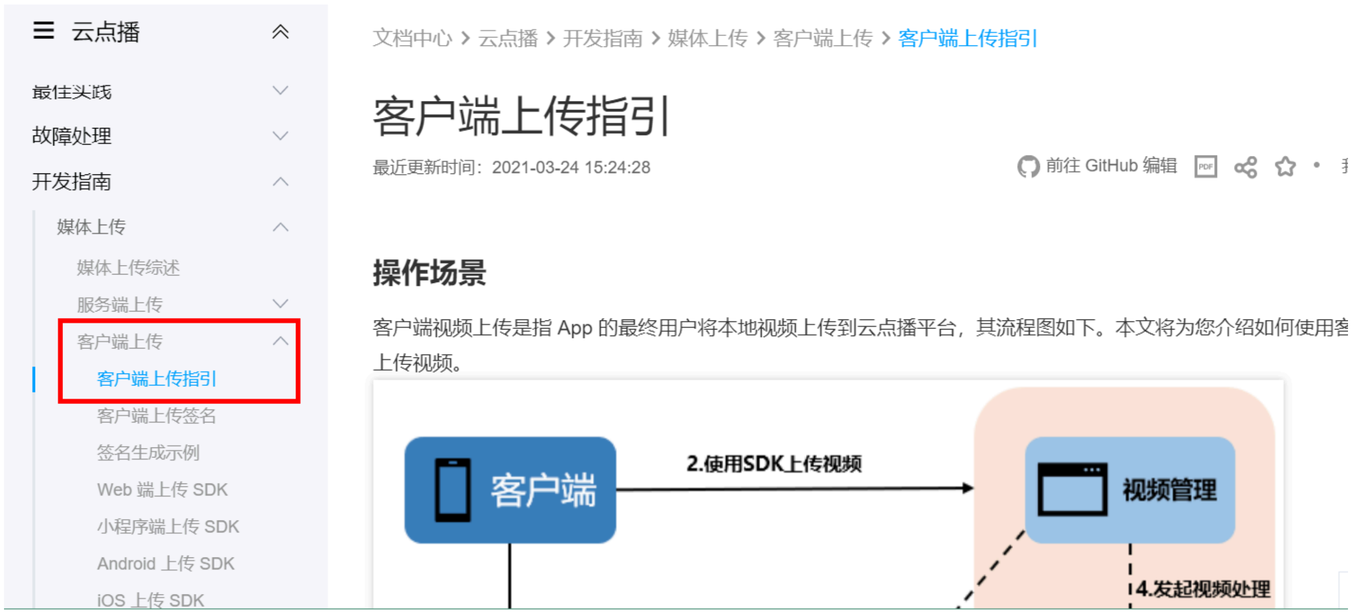

vod
course
components
Chapter
Video
Form.vue

```
<template>
  <!-- 添加和修改课时表单 -->
  <el-dialog :visible="dialogVisible" title="添加课时" @close="close()">
    <el-form :model="video" label-width="120px">
      <el-form-item label="课时标题">
        <el-input v-model="video.title" />
      </el-form-item>
      <el-form-item label="课时排序">
        <el-input-number v-model="video.sort" :min="0" />
      </el-form-item>
      <el-form-item label="是否免费">
        <el-radio-group v-model="video.isFree">
          <el-radio :label="0">免费</el-radio>
          <el-radio :label="1">默认</el-radio>
        </el-radio-group>
      </el-form-item>
      <!-- 上传视频 -->
      <el-form-item label="上传视频">
        <el-upload
          ref="upload"
          :auto-upload="false"
          :on-success="handleUploadSuccess"
          :on-error="handleUploadError"
          :on-exceed="handleUploadExceed"
          :file-list="fileList"
          :limit="1"
          :before-remove="handleBeforeRemove"
          :on-remove="handleOnRemove"
          :action="BASE_API + '/admin/vod/upload'"
        >
          <el-button slot="trigger" size="small" type="primary"
            >选择视频</el-button
          >
          <el-button
            :disabled="uploadBtnDisabled"
            style="margin-left: 10px;"
            size="small"
```

```
              type="success"
              @click="submitUpload()"
              >上传</el-button
            >
          </el-upload>
        </el-form-item>
      </el-form>
      <div slot="footer" class="dialog-footer">
        <el-button @click="close()">取 消</el-button>
        <el-button type="primary" @click="saveOrUpdate()">确 定</el-button>
      </div>
    </el-dialog>
  </template>
  <script>
    import videoApi from "@/api/vod/video";
    import vodApi from "@/api/vod/vod";
    export default {
      data() {
        return {
          BASE_API: "http://localhost:8301",
          dialogVisible: false,
          video: {
            sort: 0,
            free: false,
          },
          fileList: [], // 上传文件列表
          uploadBtnDisabled: false,
        };
      },
      methods: {
        open(chapterId, videoId) {
          this.dialogVisible = true;
          this.video.chapterId = chapterId;
          if (videoId) {
            videoApi.getById(videoId).then((response) => {
              this.video = response.data;
              // 回显
              if (this.video.videoOriginalName) {
                this.fileList = [{ name: this.video.videoOriginalName }];
              }
            });
          }
        },
        close() {
          this.dialogVisible = false;
          // 重置表单
          this.resetForm();
        },
        resetForm() {
          this.video = {
            sort: 0,
            free: false,
          };
          this.fileList = []; // 重置视频上传列表
        },
        saveOrUpdate() {
          if (!this.video.id) {
            this.save();
          } else {
            this.update();
          }
        },
        save() {
          this.video.courseId = this.$parent.$parent.courseId;
          videoApi.save(this.video).then((response) => {
            this.$message.success(response.message);
            // 关闭组件
            this.close();
            // 刷新列表
            this.$parent.fetchNodeList();
          });
        },
        update() {
          videoApi.updateById(this.video).then((response) => {
            this.$message.success(response.message);
            // 关闭组件
            this.close();
            // 刷新列表
            this.$parent.fetchNodeList();
          });
        },
        // 上传多于一个视频
        handleUploadExceed(files, fileList) {
          this.$message.warning("想要重新上传视频，请先删除已上传的视频");
        },
        // 上传
        submitUpload() {
          this.uploadBtnDisabled = true;
          this.$refs.upload.submit(); // 提交上传请求
        },
        // 视频上传成功的回调
        handleUploadSuccess(response, file, fileList) {
          this.uploadBtnDisabled = false;
          this.video.videoSourceId = response.data;
          this.video.videoOriginalName = file.name;
        },
        // 失败回调
        handleUploadError() {
          this.uploadBtnDisabled = false;
          this.$message.error("上传失败2");
        },
        // 删除视频文件确认
        handleBeforeRemove(file, fileList) {
          return this.$confirm(`确定移除 ${file.name}? `);
        },
        // 执行视频文件的删除
        handleOnRemove(file, fileList) {
          if (!this.video.videoSourceId) {
            return;
          }
          vodApi.removeByVodId(this.video.videoSourceId).then((response) => {
            this.video.videoSourceId = "";
            this.video.videoOriginalName = "";
            videoApi.updateById(this.video);
            this.$message.success(response.message);
          });
        },
      },
    };
  </script>
```

5、腾讯云上传视频其他方式

5.1、客户端上传视频



客户端上传指引

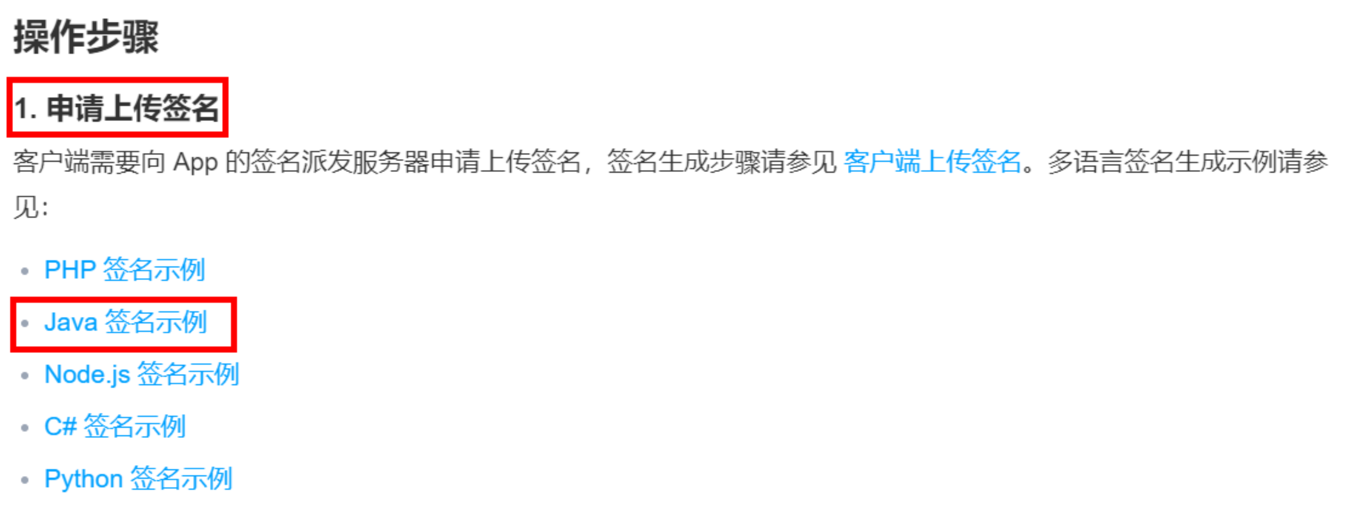
最近更新时间：2021-03-24 15:24:28

操作场景

客户端视频上传是指 App 的最终用户将本地视频上传到云点播平台，其流程图如下。本文将为您介绍如何使用客
上传视频。

5.2、操作步骤一（申请上传签名）

5.2.1、找到 Java 签名示例

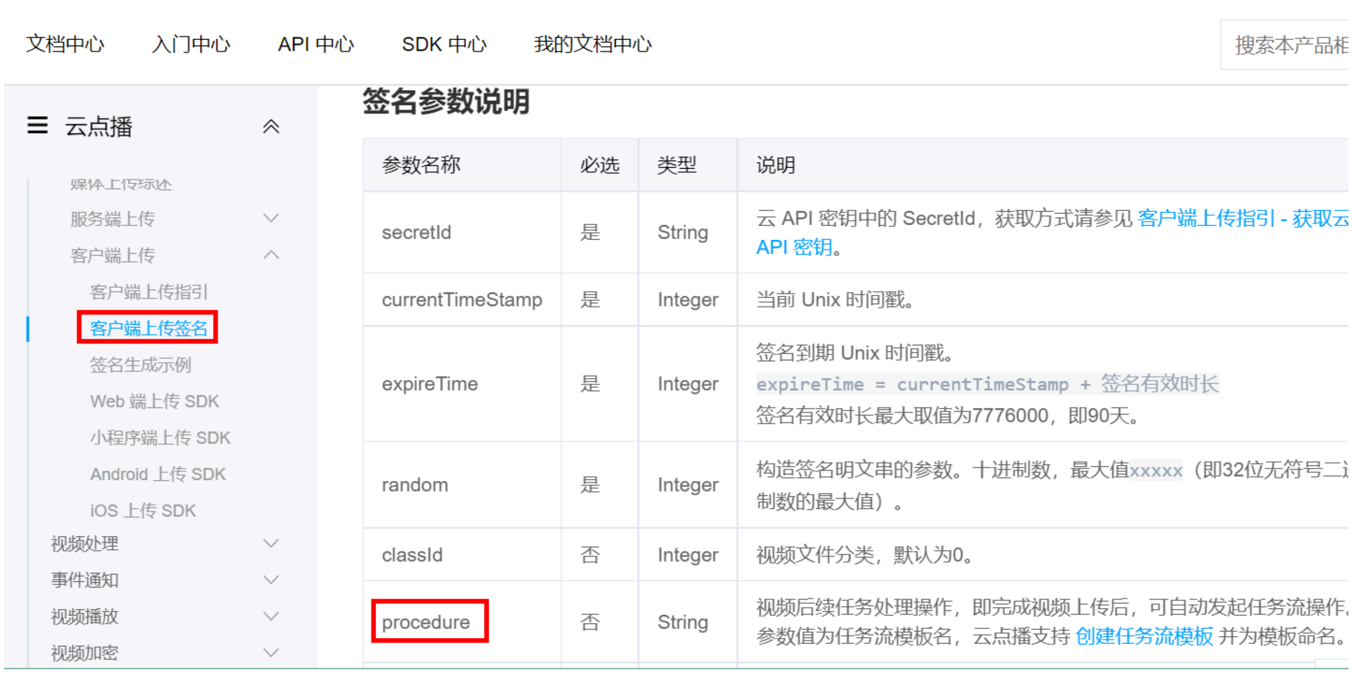


操作步骤

1. 申请上传签名

客户端需要向 App 的签名派发服务器申请上传签名，签名生成步骤请参见 客户端上传签名。多语言签名生成示例请参见：

- PHP 签名示例
- Java 签名示例
- Node.js 签名示例
- C# 签名示例
- Python 签名示例



签名参数说明

参数名称	必选	类型	说明
secretId	是	String	云 API 密钥中的 SecretId，获取方式请参见 客户端上传指引 - 获取云 API 密钥。
currentTimeStamp	是	Integer	当前 Unix 时间戳。
expireTime	是	Integer	签名到期 Unix 时间戳。 `expireTime = currentTimeStamp + 签名有效时长` 签名有效时长最大取值为7776000，即90天。
random	是	Integer	构造签名明文串的参数。十进制数，最大值xxxxx（即32位无符号二 制数的最大值）。
classId	否	Integer	视频文件分类，默认为0。
procedure	否	String	视频后续任务处理操作，即完成视频上传后，可自动发起任务流操作, 参数值为任务流模板名，云点播支持 创建任务流模板 并为模板命名。

5.2.2、VodController 编写签名接口

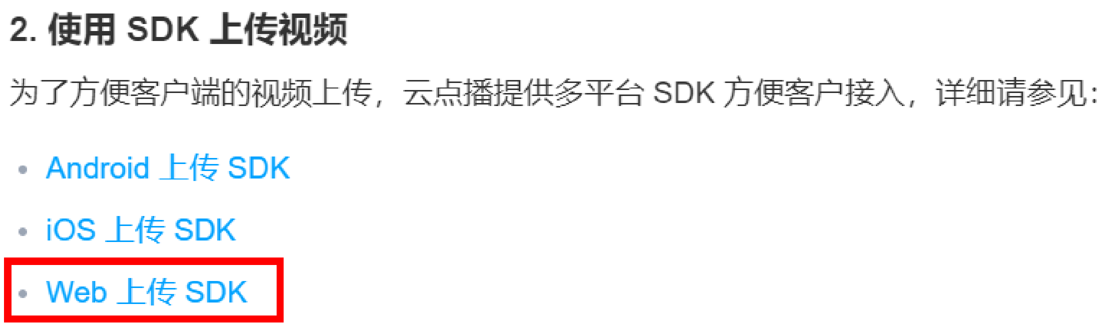
Signature 类

```java
import java.util.Random;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import sun.misc.BASE64Encoder;

public class Signature {
    private String secretId;
    private String secretKey;
    private long currentTime;
    private int random;
    private int signValidDuration;
    private static final String HMAC_ALGORITHM = "HmacSHA1"; //签名算法
    private static final String CONTENT_CHARSET = "UTF-8";
    public static byte[] byteMerger(byte[] byte1, byte[] byte2) {
        byte[] byte3 = new byte[byte1.length + byte2.length];
        System.arraycopy(byte1, 0, byte3, 0, byte1.length);
        System.arraycopy(byte2, 0, byte3, byte1.length, byte2.length);
        return byte3;
    }
    // 获取签名
    public String getUploadSignature() throws Exception {
        String strSign = "";
        String contextStr = "";
        // 生成原始参数字符串
        long endTime = (currentTime + signValidDuration);
        contextStr += "secretId=" + java.net.URLEncoder.encode(secretId, "utf8");
        contextStr += "&currentTimeStamp=" + currentTime;
        contextStr += "&expireTime=" + endTime;
        contextStr += "&random=" + random;
        //设置转码任务流
        contextStr += "&procedure=LongVideoPreset";

        try {
            Mac mac = Mac.getInstance(HMAC_ALGORITHM);
            SecretKeySpec secretKey = new SecretKeySpec(this.secretKey.getBytes(CONTENT_CHARSET),
            mac.init(secretKey);
            byte[] hash = mac.doFinal(contextStr.getBytes(CONTENT_CHARSET));
            byte[] sigBuf = byteMerger(hash, contextStr.getBytes("utf8"));
            strSign = base64Encode(sigBuf);
            strSign = strSign.replace(" ", "").replace("\n", "").replace("\r", "");
        } catch (Exception e) {
            throw e;
        }
        return strSign;
    }
    private String base64Encode(byte[] buffer) {
        BASE64Encoder encoder = new BASE64Encoder();
        return encoder.encode(buffer);
    }
    public void setSecretId(String secretId) {
        this.secretId = secretId;
    }
    public void setSecretKey(String secretKey) {
        this.secretKey = secretKey;
    }
    public void setCurrentTime(long currentTime) {
        this.currentTime = currentTime;
    }
    public void setRandom(int random) {
        this.random = random;
    }
    public void setSignValidDuration(int signValidDuration) {
        this.signValidDuration = signValidDuration;
    }
}
```

VodController 类

```java
@GetMapping("sign")
public Result sign() {
    Signature sign = new Signature();
    // 设置 App 的云 API 密钥
    sign.setSecretId(ConstantPropertiesUtil.ACCESS_KEY_ID);
    sign.setSecretKey(ConstantPropertiesUtil.ACCESS_KEY_SECRET);
    sign.setCurrentTime(System.currentTimeMillis() / 1000);
    sign.setRandom(new Random().nextInt(java.lang.Integer.MAX_VALUE));
    sign.setSignValidDuration(3600 * 24 * 2); // 签名有效期：2天
    try {
        String signature = sign.getUploadSignature();
        System.out.println("signature : " + signature);
        return Result.ok(signature);
    } catch (Exception e) {
        System.out.print("获取签名失败");
        e.printStackTrace();
        return Result.fail(null);
    }
}
```

5.3、操作步骤二（SDK 上传）

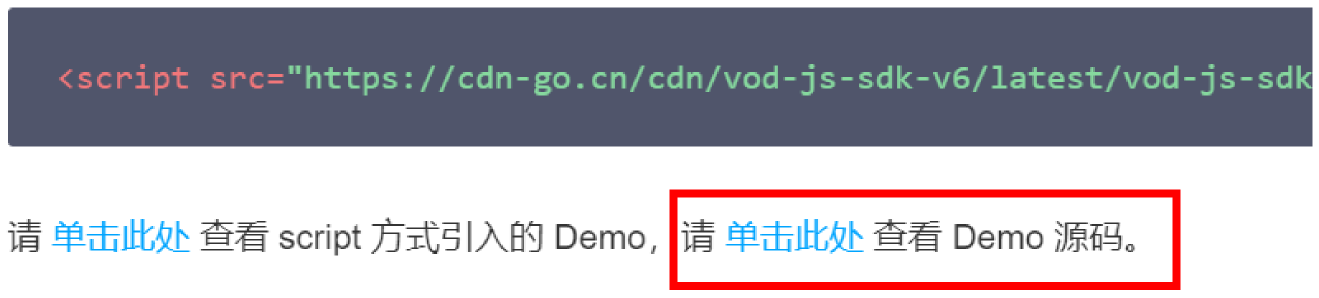


2. 使用 SDK 上传视频

为了方便客户端的视频上传，云点播提供多平台 SDK 方便客户接入，详细请参见：

- Android 上传 SDK
- iOS 上传 SDK
- Web 上传 SDK

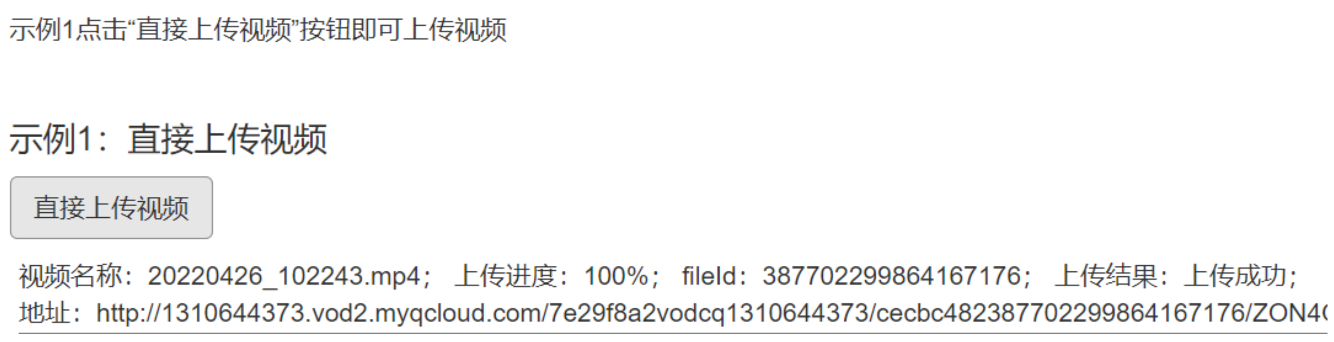
5.3.1、下载 Demo 源码修改



```
<script src="https://cdn-go.cn/cdn/vod-js-sdk-v6/latest/vod-js-sdk
```

请 单击此处 查看 script 方式引入的 Demo，请 单击此处 查看 Demo 源码。

5.3.2、html 文件测试上传



示例1点击“直接上传视频”按钮即可上传视频

示例1：直接上传视频

视频名称：20220426_102243.mp4； 上传进度：100%； fileId：387702299864167176； 上传结果：上传成功；
地址：http://1310644373.vod2.myqcloud.com/7e29f8a2vodcq1310644373/cecbc482387702299864167176/ZON4

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>QCloud VIDEO UGC UPLOAD SDK</title>
    <link
      href="//cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.5/css/bootstrap.min.css"
      rel="stylesheet"
    />
    <style type="text/css">
      .text-danger {
        color: red;
      }

      .control-label {
        text-align: left !important;
      }

      #resultBox {
        width: 100%;
        height: 300px;
        border: 1px solid #888;
        padding: 5px;
        overflow: auto;
        margin-bottom: 20px;
      }

      .uploaderMsgBox {
        width: 100%;
        border-bottom: 1px solid #888;
      }

      .cancel-upload {
        text-decoration: none;
        cursor: pointer;
      }
    </style>
  </head>
  <body>
    <div id="content">
      <div class="container">
        <h1>UGC-Uploader</h1>
      </div>
    </div>
    <div class="container" id="main-area">
      <div class="row" style="padding:10px;">
        <p>示例1点击“直接上传视频”按钮即可上传视频。<br />。</p>
      </div>
      <form ref="vExample">
        <input
          type="file"
          style="display:none;"
          ref="vExampleFile"
          @change="vExampleUpload"
        />
      </form>
      <div class="row" style="padding:10px;">
        <h4>示例1：直接上传视频</h4>
        <a
          href="javascript:void(0);"
          class="btn btn-default"
          @click="vExampleAdd"
          >直接上传视频</a
        >
      </div>
      <!-- 上传信息组件     -->
      <div class="uploaderMsgBox" v-for="uploaderInfo in uploaderInfos">
        <div v-if="uploaderInfo.videoInfo">
          视频名称：{{uploaderInfo.videoInfo.name + '.' +
          uploaderInfo.videoInfo.type}};
          上传进度：{{Math.floor(uploaderInfo.progress * 100) + '%'}};
        fileId: {{uploaderInfo.fileId}};
        上传结果: {{uploaderInfo.isVideoUploadCancel ? '已取消' :
        uploaderInfo.isVideoUploadSuccess ? '上传成功' : '上传中'}};
        <br />
        地址: {{uploaderInfo.videoUrl}};
        <a
          href="javascript:void(0);"
          class="cancel-upload"
          v-if="!uploaderInfo.isVideoUploadSuccess && !uploaderInfo.isVideoUploadCancel"
          @click="uploaderInfo.cancel()"
          >取消上传</a
        ><br />
      </div>
      <div v-if="uploaderInfo.coverInfo">
        封面名称: {{uploaderInfo.coverInfo.name}};
        上传进度: {{Math.floor(uploaderInfo.coverProgress * 100) + '%'}};
        上传结果: {{uploaderInfo.isCoverUploadSuccess ? '上传成功' :
        '上传中'}};
        <br />
        地址: {{uploaderInfo.coverUrl}};
        <br />
      </div>
    </div>
  </div>
  <script src="https://cdn.jsdelivr.net/npm/es6-promise@4/dist/es6-promise.auto.js"></script>
  <script src="//cdnjs.cloudflare.com/ajax/libs/vue/2.5.21/vue.js"></script>
  <script src="//cdnjs.cloudflare.com/ajax/libs/axios/0.18.0/axios.js"></script>
  <script src="https://cdn-go.cn/cdn/vod-js-sdk-v6/latest/vod-js-sdk-v6.js"></script>

  <script type="text/javascript">
    (function () {
      /**
       * 计算签名。调用签名接口获取
       **/
      function getSignature() {
        return axios
          .get("http://localhost:8301/admin/vod/user/sign")
          .then((response) => {
            return response.data.data;
          });
      }
      var app = new Vue({
        el: "#main-area",
        data: {
          uploaderInfos: [],

          vcExampleVideoName: "",
          vcExampleCoverName: "",

          cExampleFileId: "",
        },
        created: function () {
          this.tcVod = new TcVod.default({
            getSignature: getSignature,
          });
        },
        methods: {
          /**
           * vExample示例。添加视频
           **/
          vExampleAdd: function () {
            this.$refs.vExampleFile.click();
          },
          /**
           * vExample示例。上传视频过程。
           **/
          vExampleUpload: function () {
            var self = this;
            var mediaFile = this.$refs.vExampleFile.files[0];

            var uploader = this.tcVod.upload({
              mediaFile: mediaFile,
            });
            uploader.on("media_progress", function (info) {
```

```
        uploaderInfo.progress = info.percent;
      });
      uploader.on("media_upload", function (info) {
        uploaderInfo.isVideoUploadSuccess = true;
      });

      console.log(uploader, "uploader");

      var uploaderInfo = {
        videoInfo: uploader.videoInfo,
        isVideoUploadSuccess: false,
        isVideoUploadCancel: false,
        progress: 0,
        fileId: "",
        videoUrl: "",
        cancel: function () {
          uploaderInfo.isVideoUploadCancel = true;
          uploader.cancel();
        },
      };

      this.uploaderInfos.push(uploaderInfo);
      uploader
        .done()
        .then(function (doneResult) {
          console.log("doneResult", doneResult);
          uploaderInfo.fileId = doneResult.fileId;
          return doneResult.video.url;
        })
        .then(function (videoUrl) {
          uploaderInfo.videoUrl = videoUrl;
          self.$refs.vExample.reset();
        });
    },
    // cExample 上传过程
    cExampleUpload: function () {
      var self = this;
```

```
          var coverFile = this.$refs.cExampleCover.files[0];

          var uploader = this.tcVod.upload({
            fileId: this.cExampleFileId,
            coverFile: coverFile,
          });
          uploader.on("cover_progress", function (info) {
            uploaderInfo.coverProgress = info.percent;
          });
          uploader.on("cover_upload", function (info) {
            uploaderInfo.isCoverUploadSuccess = true;
          });
          console.log(uploader, "uploader");

          var uploaderInfo = {
            coverInfo: uploader.coverInfo,
            isCoverUploadSuccess: false,
            coverProgress: 0,
            coverUrl: "",
            cancel: function () {
              uploader.cancel();
            },
          };

          this.uploaderInfos.push(uploaderInfo);

          uploader.done().then(function (doneResult) {
            console.log("doneResult", doneResult);
            uploaderInfo.coverUrl = doneResult.cover.url;
            self.$refs.cExample.reset();
          });
        },
      },
    });
  })();
</script>
<!-- Global site tag (gtag.js) - Google Analytics -->
```

```
    <script
      async
      src="https://www.googletagmanager.com/gtag/js?id=UA-26476625-7"
    ></script>
    <script>
      // add by alsotang@gmail.com
      window.dataLayer = window.dataLayer || [];
      function gtag() {
        dataLayer.push(arguments);
      }
      gtag("js", new Date());
      gtag("config", "UA-26476625-7");
    </script>
  </body>
</html>
```

6、完善删除视频功能

6.1、修改 VideoController 方法

修改删除小节方法

```
@ApiOperation(value = "删除")
@DeleteMapping("remove/{id}")
public Result remove(@PathVariable Long id) {
    videoService.removeVideoById(id);
    return Result.ok();
}
```

6.2、修改 Service 方法

修改 VideoService 和 VideoServiceImpl

```
@Service
public class VideoServiceImpl extends ServiceImpl<VideoMapper, Video> implements VideoService {

    @Autowired
    private VodService vodService;

    //根据课程id删除小节
    @Override
    public void removeVideoByCourseId(Long id) {
        //1 删除小节中的视频
        //根据课程id获取课程里面所有小节
        QueryWrapper<Video> wrapper = new QueryWrapper<>();
        wrapper.eq("course_id",id);
        List<Video> videoList = baseMapper.selectList(wrapper);
        //遍历获取每个小节中的视频id
        for(Video video:videoList) {
            String videoSourceId = video.getVideoSourceId();
            //如果视频id不为空，调用方法删除
            if(!StringUtils.isEmpty(videoSourceId)) {
                vodService.removeVideo(videoSourceId);
            }
        }
        //2 根据课程id删除小节
        baseMapper.delete(wrapper);
    }

    //根据小节id删除小节删除视频
    @Override
    public void removeVideoById(Long id) {
        //1 删除视频
        Video video = baseMapper.selectById(id);
        //获取视频id
        String videoSourceId = video.getVideoSourceId();
        //如果视频id不为空，调用方法删除
        if(!StringUtils.isEmpty(videoSourceId)) {
            vodService.removeVideo(videoSourceId);
        }
        //2 删除小节
        baseMapper.deleteById(id);
    }
}
```