

学成在线-第14天-讲义-媒资管理

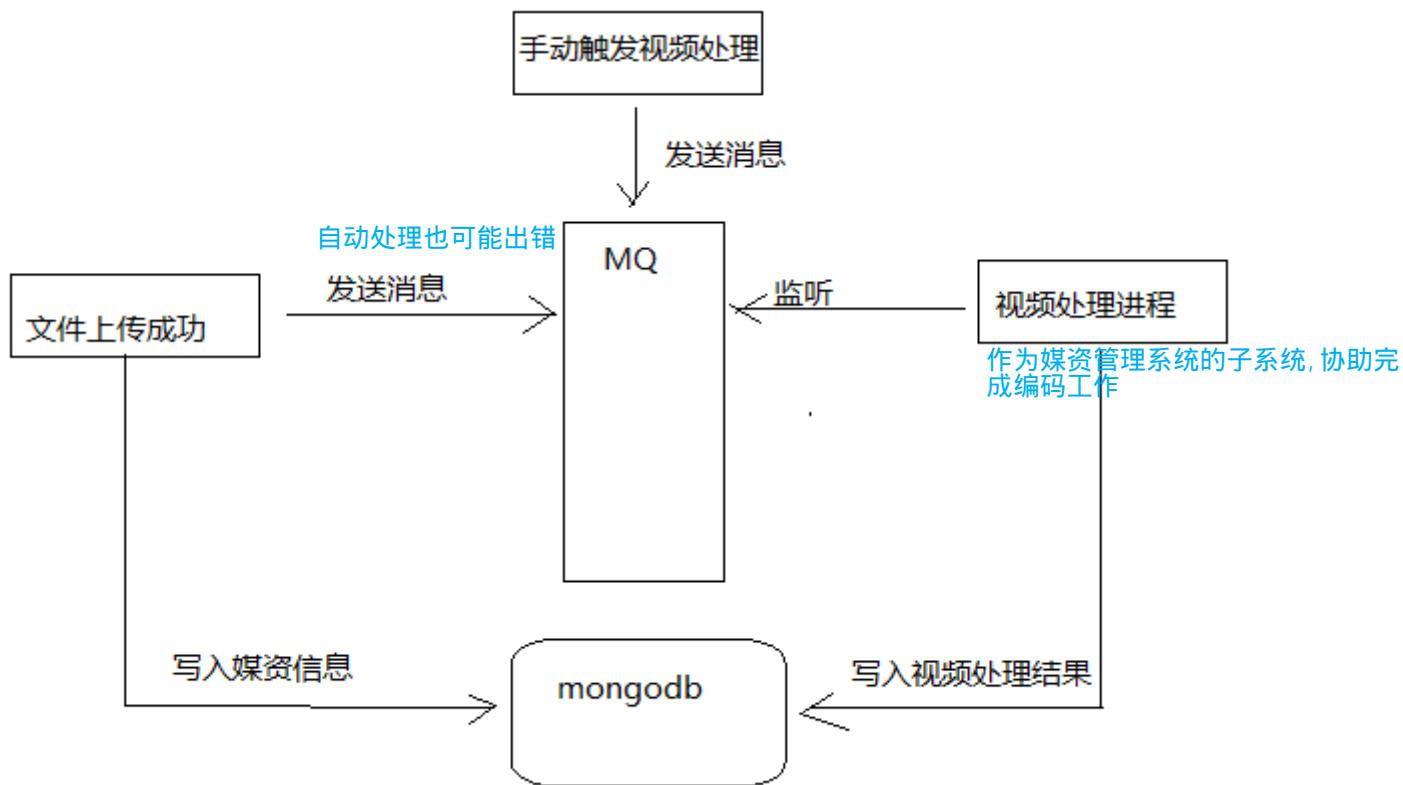
1 视频处理

1.1 需求分析

原始视频通常需要经过编码处理，生成m3u8和ts文件方可基于HLS协议播放视频。通常用户上传原始视频，系统自动处理成标准格式，系统对用户上传的视频自动编码、转换，最终生成m3u8文件和ts文件，处理流程如下：

- 1、用户上传视频成功
- 2、系统对上传成功的视频自动开始编码处理
- 3、用户查看视频处理结果，没有处理成功的视频用户可在管理界面再次触发处理
- 4、视频处理完成将视频地址及处理结果保存到数据库

视频处理流程如下：



视频处理进程的任务是接收视频处理消息进行视频处理，业务流程如下：

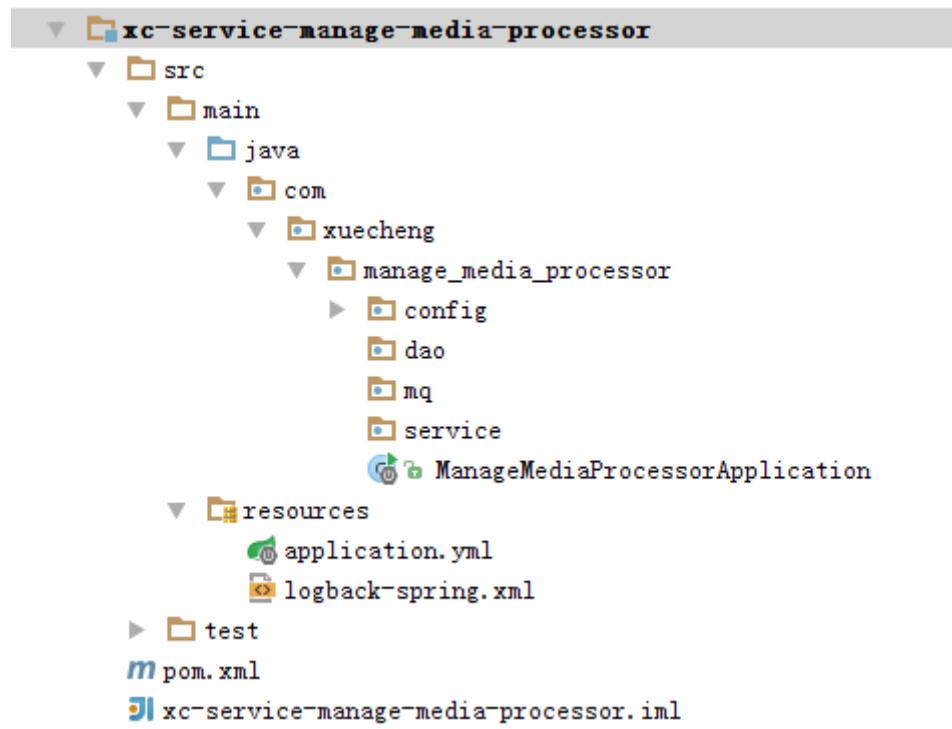
- 1、监听MQ，接收视频处理消息。
- 2、进行视频处理。
- 3、向数据库写入视频处理结果。

视频处理进程属于媒资管理系统的一部分，考虑提高系统的扩展性，将视频处理单独定义视频处理工程。

1.2 视频处理开发

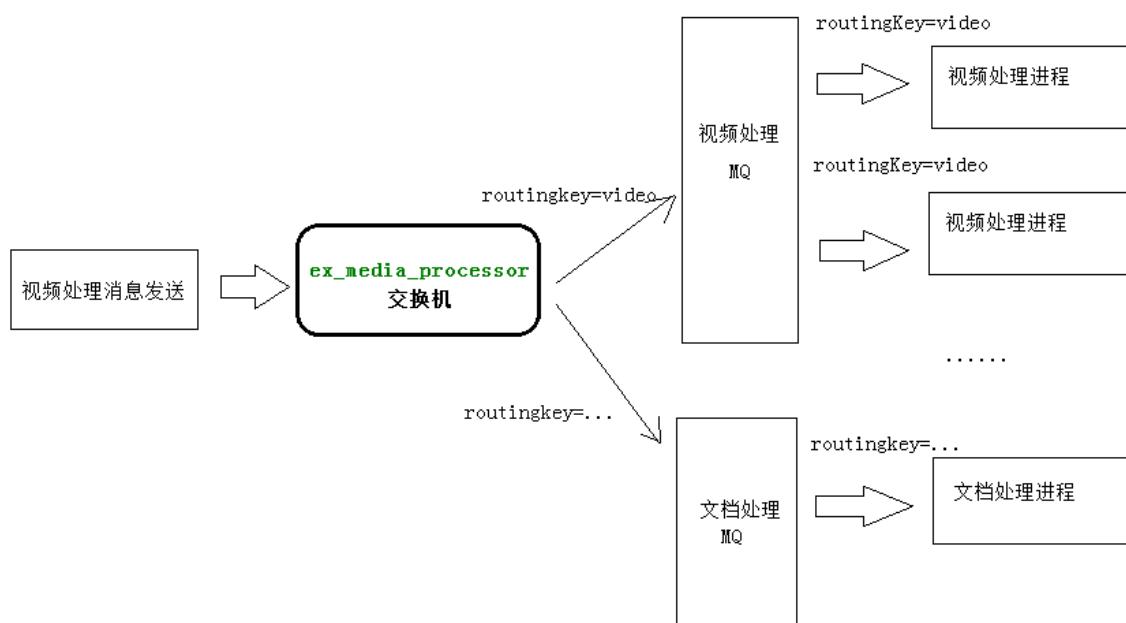
1.2.1 视频处理工程创建

1、导入“资料”下的视频处理工程：xc-service-manage-media-processor



2、RabbitMQ配置

使用rabbitMQ的routing交换机模式，视频处理程序监听视频处理队列，如下图：





RabbitMQ配置如下：

```
@Configuration
public class RabbitMQConfig {
    public static final String EX_MEDIA_PROCESSTASK = "ex_media_processor";

    //视频处理队列
    @Value("${xc-service-manage-media.mq.queue-media-video-processor}")
    public String queue_media_video_processtask;

    //视频处理路由
    @Value("${xc-service-manage-media.mq.routingkey-media-video}")
    public String routingkey_media_video;

    /**
     * 交换机配置
     * @return the exchange
     */
    @Bean(EX_MEDIA_PROCESSTASK)
    public Exchange EX_MEDIA_VIDEOTASK() {
        return ExchangeBuilder.directExchange(EX_MEDIA_PROCESSTASK).durable(true).build();
    }
    //声明队列
    @Bean("queue_media_video_processtask")
    public Queue QUEUE_PROCESSTASK() {
        Queue queue = new Queue(queue_media_video_processtask,true,false,true);
        return queue;
    }
    /**
     * 绑定队列到交换机 .
     * @param queue    the queue
     * @param exchange the exchange
     * @return the binding
     */
    @Bean
    public Binding binding_queue_media_processtask(@Qualifier("queue_media_video_processtask")
    Queue queue, @Qualifier(EX_MEDIA_PROCESSTASK) Exchange exchange) {
        return BindingBuilder.bind(queue).to(exchange).with(routingkey_media_video).noargs();
    }
}
```

在application.yml中配置队列名称及routingkey

```
xc-service-manage-media:
mq:
queue-media-video-processor: queue_media_video_processor
routingkey-media-video: routingkey_media_video
```



1.2.2 视频处理技术方案

如何通过程序进行视频处理？

ffmpeg是一个可行的视频处理程序，可以通过Java调用ffmpeg.exe完成视频处理。

在java中可以使用Runtime类和Process Builder类两种方式来执行外部程序，工作中至少掌握一种。

本项目使用Process Builder的方式来调用ffmpeg完成视频处理。

关于Process Builder的测试如下：

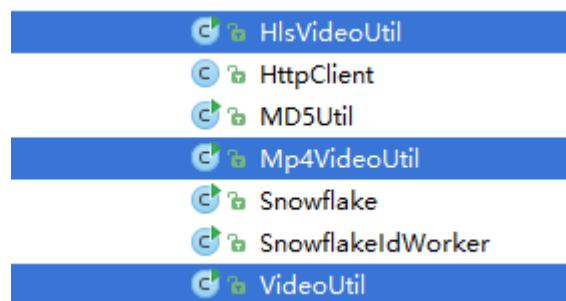
```
@Test
public void testProcessBuilder(){
    ProcessBuilder processBuilder = new ProcessBuilder();
    // processBuilder.command("ping","127.0.0.1");
    processBuilder.command("ipconfig");
    //将标准输入流和错误输入流合并，通过标准输入流读取信息
    processBuilder.redirectErrorStream(true);
    try {
        //启动进程
        Process start = processBuilder.start();
        //获取输入流
        InputStream inputStream = start.getInputStream();
        //转成字符输入流
        InputStreamReader inputStreamReader = new InputStreamReader(inputStream,"gbk");

        int len = -1;
        char[] c = new char[1024];
        StringBuffer outputString = new StringBuffer();
        //读取进程输入流中的内容
        while ((len= inputStreamReader.read(c))!=-1) {
            String s = new String(c,0,len);
            outputString.append(s);
            System.out.print(s);
        }
        inputStream.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
@Test
public void testFFmpeg(){
    ProcessBuilder processBuilder = new ProcessBuilder();
    //定义命令内容
    List<String> command = new ArrayList<>();
    command.add("D:\\Program Files\\ffmpeg-20180227-fa0c9d6-win64-static\\bin\\ffmpeg.exe");
    command.add("-i");
    command.add("E:\\ffmpeg_test\\1.avi");
    command.add("-y");//覆盖输出文件
    command.add("-c:v");
    command.add("libx264");
}
```



```
command.add("-s");
command.add("1280x720");
command.add("-pix_fmt");
command.add("yuv420p");
command.add("-b:a");
command.add("63k");
command.add("-b:v");
command.add("753k");
command.add("-r");
command.add("18");
command.add("E:\\ffmpeg_test\\1.mp4");
processBuilder.command(command);
//将标准输入流和错误输入流合并，通过标准输入流读取信息
processBuilder.redirectErrorStream(true);
try {
    //启动进程
    Process start = processBuilder.start();
    //获取输入流
    InputStream inputStream = start.getInputStream();
    //转成字符输入流
    InputStreamReader inputStreamReader = new InputStreamReader(inputStream,"gbk");
    int len = -1;
    char[] c = new char[1024];
    StringBuffer outputString = new StringBuffer();
    //读取进程输入流中的内容
    while ((len= inputStreamReader.read(c))!=-1) {
        String s = new String(c,0,len);
        outputString.append(s);
        System.out.print(s);
    }
    inputStream.close();
} catch (IOException e) {
    e.printStackTrace();
}
}
```

上边的代码已经封装成工具类，参见：



上边的工具类中：

Mp4VideoUtil.java完成avi转mp4



HlsVideoUtil.java完成mp4转hls

分别测试每个工具类的使用方法。

```
public static void main(String[] args) throws IOException {
    //ffmpeg的路径
    String ffmpeg_path = "D:\\Program Files\\ffmpeg-20180227-fa0c9d6-win64-
static\\bin\\ffmpeg.exe";//ffmpeg的安装位置
    //源avi视频的路径
    String video_path = "E:\\ffmpeg_test\\1.avi";
    //转换后mp4文件的名称
    String mp4_name = "1.mp4";
    //转换后mp4文件的路径
    String mp4_path = "E:\\ffmpeg_test\\";
    //创建工具类对象
    Mp4VideoUtil videoUtil = new Mp4VideoUtil(ffmpeg_path,video_path,mp4_name,mp4_path);
    //开始视频转换，成功将返回success
    String s = videoUtil.generateMp4();
    System.out.println(s);
}
```

1.2.3 视频处理实现

1.2.3.1 确定消息格式

MQ消息统一采用json格式，视频处理生产方会向MQ发送如下消息，视频处理消费方接收此消息后进行视频处理：

```
{ "mediaId":XXX }
```

1.2.3.2 处理流程

- 1) 接收视频处理消息
- 2) 判断媒体文件是否需要处理（本视频处理程序目前只接收avi视频的处理）
当前只有avi文件需要处理，其它文件需要更新处理状态为“无需处理”。
- 3) 处理前初始化处理状态为“未处理”
- 4) 处理失败需要在数据库记录处理日志，及处理状态为“处理失败”
- 5) 处理成功记录处理状态为“处理成功”

1.2.3.3 数据模型

在MediaFile类中添加mediaFileProcess_m3u8属性记录ts文件列表，代码如下：



```
//处理状态
private String processStatus;
//hls处理
private MediaFileProcess_m3u8 mediaFileProcess_m3u8;
```

```
@Data
@Override
public class MediaFileProcess_m3u8 extends MediaFileProcess {
    //ts列表
    private List<String> tslist;
}
```

1.2.3.4 视频处理生成Mp4

1、创建Dao

视频处理结果需要保存到媒资数据库，创建dao如下：

```
public interface MediaFileRepository extends MongoRepository<MediaFile, String> {
}
```

2、在application.yml中配置ffmpeg的位置及视频目录的根目录：

```
xc-service-manage-media:
  video-location: F:/develop/video/
  ffmpeg-path: D:/Program Files/ffmpeg-20180227-fa0c9d6-win64-static/bin/ffmpeg.exe
```

3、处理任务类

在mq包下创建MediaProcessTask类，此类负责监听视频处理队列，并进行视频处理。

整个视频处理内容较多，这里分两部分实现：生成Mp4和生成m3u8，下边代码实现了生成mp4。

```
@Component
public class MediaProcessTask {
    private static final Logger LOGGER = LoggerFactory.getLogger(MediaProcessTask.class);

    //ffmpeg绝对路径
    @Value("${xc-service-manage-media.ffmpeg-path}")
    String ffmpeg_path;

    //上传文件根目录
    @Value("${xc-service-manage-media.upload-location}")
    String serverPath;
```



```
@Autowired
MediaFileRepository mediaFileRepository;

@RabbitListener(queues = "${xc-service-manage-media.mq.queue-media-processtask}")
public void receiveMediaProcessTask(String msg) throws IOException {
    Map msgMap = JSON.parseObject(msg, Map.class);
    LOGGER.info("receive media process task msg :{} ",msgMap);
    //解析消息
    //媒资文件id
    String mediaId = (String) msgMap.get("mediaId");
    //获取媒资文件信息
    Optional<MediaFile> optional = mediaFileRepository.findById(fileMd5);
    if(!optional.isPresent()){
        return ;
    }
    MediaFile mediaFile = optional.get();
    //媒资文件类型
    String fileType = mediaFile.getFileType();
    if(fileType == null || !fileType.equals("avi")){//目前只处理avi文件
        mediaFile.setProcessStatus("303004");//处理状态为无需处理
        mediaFileRepository.save(mediaFile);
        return ;
    }else{
        mediaFile.setProcessStatus("303001");//处理状态为未处理
        mediaFileRepository.save(mediaFile);
    }
    //生成mp4
    String video_path = serverPath + mediaFile.getPath()+mediaFile.getName();
    String mp4_name = mediaFile.getId()+"mp4";
    String mp4folder_path = serverPath + mediaFile.getPath();
    Mp4VideoUtil videoUtil = new
Mp4VideoUtil(ffmpeg_path,video_path,mp4_name,mp4folder_path);
    String result = videoUtil.generateMp4();
    if(result == null || !result.equals("success")){
        //操作失败写入处理日志
        mediaFile.setProcessStatus("303003");//处理状态为处理失败
        MediaFileProcess_m3u8 mediaFileProcess_m3u8 = new MediaFileProcess_m3u8();
        mediaFileProcess_m3u8.setErrorMsg(result);
        mediaFile.setMediaFileProcess_m3u8(mediaFileProcess_m3u8);
        mediaFileRepository.save(mediaFile);
        return ;
    }
    //生成m3u8...
}

}
```

说明：

1、原始视频转成mp4如何判断转换成功？



根据视频时长来判断，取原视频和转换成功视频的时长（时分秒），如果相等则相同。

1.2.3.5 视频处理生成m3u8

下边是完整的视频处理任务类代码，包括了生成m3u8及生成mp4的代码。

```
@Component
public class MediaProcessTask {
    private static final Logger LOGGER = LoggerFactory.getLogger(MediaProcessTask.class);

    //ffmpeg绝对路径
    @Value("${xc-service-manage-media.ffmpeg-path}")
    String ffmpeg_path;

    //上传文件根目录
    @Value("${xc-service-manage-media.upload-location}")
    String serverPath;

    @Autowired
    MediaFileRepository mediaFileRepository;

    @RabbitListener(queues = "${xc-service-manage-media.mq.queue-media-processtask}")
    public void receiveMediaProcessTask(String msg) throws IOException {
        Map msgMap = JSON.parseObject(msg, Map.class);
        LOGGER.info("receive media process task msg :{} ",msgMap);
        //解析消息
        //媒资文件id
        String mediaId = (String) msgMap.get("mediaId");
        //获取媒资文件信息
        Optional<MediaFile> optional = mediaFileRepository.findById(fileMd5);
        if( ! optional.isPresent()){
            return ;
        }
        MediaFile mediaFile = optional.get();
        //媒资文件类型
        String fileType = mediaFile.getFileType();
        if(fileType == null || !fileType.equals("avi")){//目前只处理avi文件
            mediaFile.setProcessStatus("303004");//处理状态为无需处理
            mediaFileRepository.save(mediaFile);
            return ;
        }else{
            mediaFile.setProcessStatus("303001");//处理状态为未处理
            mediaFileRepository.save(mediaFile);
        }
        //生成mp4
        String video_path = serverPath + mediaFile.getPath()+mediaFile.getName();
        String mp4_name = mediaFile.getId()+"."+"mp4";
        String mp4folder_path = serverPath + mediaFile.getPath();
        Mp4VideoUtil videoUtil = new
        Mp4VideoUtil(ffmpeg_path,video_path,mp4_name,mp4folder_path);
        String result = videoUtil.generateMp4();
```



```
if(result == null || !result.equals("success")){
    //操作失败写入处理日志
    mediaFile.setProcessStatus("303003");//处理状态为处理失败
    MediaFileProcess_m3u8 mediaFileProcess_m3u8 = new MediaFileProcess_m3u8();
    mediaFileProcess_m3u8.setErrormsg(result);
    mediaFile.setMediaFileProcess_m3u8(mediaFileProcess_m3u8);
    mediaFileRepository.save(mediaFile);
    return ;
}
//生成m3u8
video_path = serverPath + mediaFile.getFilePath()+mp4_name;//此地址为mp4的地址
String m3u8_name = mediaFile.getFileId()+"_m3u8";
String m3u8folder_path = serverPath + mediaFile.getFilePath()+"hls/";
HlsVideoUtil hlsVideoUtil = new
HlsVideoUtil(ffmpeg_path,video_path,m3u8_name,m3u8folder_path);
result = hlsVideoUtil.generateM3u8();
if(result == null || !result.equals("success")){
    //操作失败写入处理日志
    mediaFile.setProcessStatus("303003");//处理状态为处理失败
    MediaFileProcess_m3u8 mediaFileProcess_m3u8 = new MediaFileProcess_m3u8();
    mediaFileProcess_m3u8.setErrormsg(result);
    mediaFile.setMediaFileProcess_m3u8(mediaFileProcess_m3u8);
    mediaFileRepository.save(mediaFile);
    return ;
}
//获取m3u8列表
List<String> ts_list = hlsVideoUtil.get_ts_list();
//更新处理状态为成功
mediaFile.setProcessStatus("303002");//处理状态为处理成功
MediaFileProcess_m3u8 mediaFileProcess_m3u8 = new MediaFileProcess_m3u8();
mediaFileProcess_m3u8.setTslist(ts_list);
mediaFile.setMediaFileProcess_m3u8(mediaFileProcess_m3u8);
//m3u8文件url
mediaFile.setFileUrl(mediaFile.getFilePath()+"hls/"+m3u8_name);
mediaFileRepository.save(mediaFile);

}
}
```

说明：

mp4转成m3u8如何判断转换成功？

第一、根据视频时长来判断，同mp4转换成功的判断方法。

第二、最后还要判断m3u8文件内容是否完整。

1.3 发送视频处理消息



当视频上传成功后向MQ发送视频处理消息。

修改媒资管理服务的文件上传代码，当文件上传成功向MQ发送视频处理消息。

1.3.1 RabbitMQ配置

1、将media-processor工程下的RabbitmqConfig配置类拷贝到media工程下。

2、在media工程下配置mq队列等信息

修改application.yml

```
xc-service-manage-media:  
  mq:  
    queue-media-video-processor: queue_media_video_processor  
    routingkey-media-video: routingkey_media_video
```

1.3.2 修改Service

在文件合并方法中添加向mq发送视频处理消息的代码：

```
//向MQ发送视频处理消息  
public ResponseResult sendProcessVideoMsg(String mediaId){  
    Optional<MediaFile> optional = mediaFileRepository.findById(fileMd5);  
    if( ! optional.isPresent()) {  
        return new ResponseResult(CommonCode.FAIL);  
    }  
    MediaFile mediaFile = optional.get();  
    //发送视频处理消息  
    Map<String, String> msgMap = new HashMap<>();  
    msgMap.put("mediaId", mediaId);  
    //发送的消息  
    String msg = JSON.toJSONString(msgMap);  
    try {  
  
        this.rabbitTemplate.convertAndSend(RabbitMQConfig.EX_MEDIA_PROCESSTASK, routingkey_media_video,  
        msg);  
        LOGGER.info("send media process task msg:{}" ,msg);  
    } catch (Exception e) {  
        e.printStackTrace();  
        LOGGER.info("send media process task error, msg is:{} ,error:{}" ,msg, e.getMessage());  
        return new ResponseResult(CommonCode.FAIL);  
    }  
    return new ResponseResult(CommonCode.SUCCESS);  
}
```

在mergechunks方法最后调用sendProcessVideo方法。



```
.....  
    //状态为上传成功  
    mediaFile.setFileStatus("301002");  
    mediaFileRepository.save(mediaFile);  
    String mediaId = mediaFile.getFileId();  
    //向MQ发送视频处理消息  
    sendProcessVideoMsg(mediaId);  
.....
```

1.4 视频处理测试

测试流程：

- 1、上传avi文件
- 2、观察日志是否发送消息
- 3、观察视频处理进程是否接收到消息进行处理
- 4、观察mp4文件是否生成
- 5、观察m3u8及 ts文件是否生成

1.5 视频处理并发设置

代码中使用@RabbitListener注解指定消费方法，默认情况是单线程监听队列，可以观察当队列有多个任务时消费端每次只消费一个消息，单线程处理消息容易引起消息处理缓慢，消息堆积，不能最大利用硬件资源。

可以配置mq的容器工厂参数，增加并发处理数量即可实现多线程处理监听队列，实现多线程处理消息。

- 1、在RabbitmqConfig.java中添加容器工厂配置：

```
//消费者并发数量  
public static final int DEFAULT_CONCURRENT = 10;  
  
@Bean("customContainerFactory")  
public SimpleRabbitListenerContainerFactory  
containerFactory(SimpleRabbitListenerContainerFactoryConfigurer configurer, ConnectionFactory  
connectionFactory) {  
    SimpleRabbitListenerContainerFactory factory = new SimpleRabbitListenerContainerFactory();  
    factory.setConcurrentConsumers(DEFAULT_CONCURRENT);  
    factory.setMaxConcurrentConsumers(DEFAULT_CONCURRENT);  
    configurer.configure(factory, connectionFactory);  
    return factory;  
}
```

- 2、在@RabbitListener注解中指定容器工厂



```
//视频处理方法
```

```
@RabbitListener(queues = {"${xc-service-manage-media.mq.queue-media-video-processor}"},  
    containerFactory="customContainerFactory")
```

再次测试当队列有多个任务时消费端的并发处理能力。

2 我的媒资

2.1 需求分析

通过我的媒资可以查询本教育机构拥有的媒资文件，进行文件处理、删除文件、修改文件信息等操作，具体需求如下：

1、分页查询我的媒资文件

2、删除媒资文件

远程请求？

3、处理媒资文件

4、修改媒资文件信息

The screenshot shows a web-based media management system. On the left, there's a sidebar with 'Course Management' and 'Media Management'. Under 'Media Management', there are sections for 'Upload File' and 'My Media'. The main area is titled 'My Media' and contains a search bar with fields for 'Label', 'Original Name', and 'Processing Status' (set to 'All'). Below the search bar are two buttons: 'Search' (highlighted in red) and 'Upload File'. A table lists four media files:

...	Original File Name	File Name	Access URL	Label	File Size	Processing Status	Creation Time	Start Processing
1	bos_v2.0_chapter08_01第八章主要内容概述.avi	0529de17a0d54163a77d24c2f2b421cc.avi	0/5/0529de17a0d54163a77d24c2f2b421cc/hls/0529de17a0d54163a77d24c2f2b421cc.m3u8		12313600	Processing Success	2018-07-02 15:35:10	<button>Start Processing</button>
2	bos_v2.0_chapter08_02_运单快速录入业务需求分析.avi	5cc2fbf638442d6fe2d08ed688f2484/hls/5cc2fbf638442d6fe2d08ed688f2484.m3u8			44721152	Processing Success	2018-07-02 15:36:44	<button>Start Processing</button>
3	bos_v2.0_chapter08_15_ElasticSearch下载安装运行.avi	e19e872904aaafc45b6509b8128f590a.avi	e/1/e19e872904aaafc45b6509b8128f590a/hls/e19e872904aaafc45b6509b8128f590a.m3u8		40341504	Processing Success	2018-07-02 15:38:02	<button>Start Processing</button>
4	bos_v2.0_chapter08_04_dagrid行编辑插入空行实现.avi	0b47219cbd92aae1d558f7116d6d57dca9.avi	0/b/0b47219cbd92aae1d558f7116d6d57dca9/hls/0b47219cbd92aae1d558f7116d6d57dca9.m3u8		25270784	Processing Success	2018-07-02 15:42:30	<button>Start Processing</button>

2.2 API

本节讲解我的媒资文件分页查询、处理媒资文件，其它功能请学员自行实现。



```
@Api(value = "媒体文件管理",description = "媒体文件管理接口",tags = {"媒体文件管理接口"})
public interface MediaFileControllerApi {

    @ApiOperation("查询文件列表")
    public QueryResponseResult findList(int page, int size, QueryMediaFileRequest
queryMediaFileRequest) ;

}
```

2.3 服务端开发

2.3.1 Dao

```
@Repository
public interface MediaFileDao extends MongoRepository<MediaFile, String> {

}
```

2.3.2 Service

定义findList方法实现媒资文件查询列表。

```
@Service
public class MediaFileService {

    private static Logger logger = LoggerFactory.getLogger(MediaFileService.class);

    @Autowired
    MediaFileRepository mediaFileRepository;

    //文件列表分页查询
    public QueryResponseResult findList(int page,int size,QueryMediaFileRequest
queryMediaFileRequest){
        //查询条件
        MediaFile mediaFile = new MediaFile();
        if(queryMediaFileRequest == null){
            queryMediaFileRequest = new QueryMediaFileRequest();
        }
        //查询条件匹配器
        ExampleMatcher matcher = ExampleMatcher.matching()
            .withMatcher("tag", ExampleMatcher.GenericPropertyMatchers.contains())//tag字段
        模糊匹配
            .withMatcher("fileOriginalName",
        ExampleMatcher.GenericPropertyMatchers.contains())//文件原始名称模糊匹配
            .withMatcher("processStatus", ExampleMatcher.GenericPropertyMatchers.exact());//处理状态精确匹配(默认)
        //查询条件对象
        if(StringUtils.isNotEmpty(queryMediaFileRequest.getTag())){

```



```
        mediaFile.setTag(queryMediaFileRequest.getTag());
    }
    if(StringUtils.isNotEmpty(queryMediaFileRequest.getFileOriginalName())){
        mediaFile.setFileOriginalName(queryMediaFileRequest.getFileOriginalName());
    }
    if(StringUtils.isNotEmpty(queryMediaFileRequest.getProcessStatus())){
        mediaFile.setProcessStatus(queryMediaFileRequest.getProcessStatus());
    }
    //定义example实例
    Example<MediaFile> ex = Example.of(mediaFile, matcher);

    page = page-1;
    //分页参数
    Pageable pageable = new PageRequest(page, size);
    //分页查询
    Page<MediaFile> all = mediaFileRepository.findAll(ex,pageable);
    QueryResult<MediaFile> mediaFileQueryResult = new QueryResult<MediaFile>();
    mediaFileQueryResult.setList(all.getContent());
    mediaFileQueryResult.setTotal(all.getTotalElements());
    return new QueryResponseResult(CommonCode.SUCCESS,mediaFileQueryResult);
}

}
```

2.3.3 Controller

```
@RestController
@RequestMapping("/media/file")
public class MediaFileController implements MediaFileControllerApi {

    @Autowired
    MediaFileService mediaFileService;
    @Autowired
    MediaUploadService mediaUploadService;

    @Override
    @GetMapping("/list/{page}/{size}")
    public QueryResponseResult findList(@PathVariable("page") int page, @PathVariable("size")
    int size, QueryMediaFileRequest queryMediaFileRequest) {
        //媒资文件查询
        return mediaFileService.findList(page,size,queryMediaFileRequest);
    }

}
```

2.4 前端开发



2.4.1 API方法

在media模块定义api方法如下：

```
import http from '../../../../../base/api/public'
import querystring from 'querystring'
let sysConfig = require('@/../../config/sysConfig')
let apiUrl = sysConfig.xcApiUrlPre;
/*页面列表*/
export const media_list = (page,size,params) => {
    //params为json格式
    //使用querystring将json对象转成key/value串
    let querys = querystring.stringify(params)
    return http.requestQuickGet(apiUrl+'/media/file/list/'+page+'/'+size+'/?'+querys)
}
/*发送处理消息*/
export const media_process = (id) => {
    return http.requestPost(apiUrl+'/media/file/process/'+id)
}
```

2.4.2 页面

在media模块创建media_list.vue，可参考cms系统的page_list.vue来编写此页面。

1、视图

```
<template>
<div>
    <!--查询表单-->
    <el-form :model="params">
        标签：
        <el-input v-model="params.tag" style="width:160px"></el-input>
        原始名称：
        <el-input v-model="params.fileOriginalName" style="width:160px"></el-input>
        处理状态：
        <el-select v-model="params.processStatus" placeholder="请选择处理状态">
            <el-option
                v-for="item in processStatusList"
                :key="item.id"
                :label="item.name"
                :value="item.id">
            </el-option>
        </el-select>
        <br/>
        <el-button type="primary" v-on:click="query" size="small">查询</el-button>
        <router-link class="mui-tab-item" :to="{path:'/media/upload'}">
            <el-button type="primary" size="small" >上传文件</el-button>
        </router-link>
    </el-form>
    <!--列表-->
    <el-table :data="list" highlight-current-row v-loading="listLoading" style="width: 100%;>
        <el-table-column type="index" width="30">
```



```
</el-table-column>
<el-table-column prop="fileOriginalName" label="原始文件名称" width="220">
</el-table-column>
<el-table-column prop="fileName" label="文件名称" width="220">
</el-table-column>
<el-table-column prop="fileUrl" label="访问url" width="260">
</el-table-column>
<el-table-column prop="tag" label="标签" width="100">
</el-table-column>
<el-table-column prop="fileSize" label="文件大小" width="120">
</el-table-column>
<el-table-column prop="processStatus" label="处理状态" width="100"
:formatter="formatProcessStatus">
</el-table-column>
<el-table-column prop="uploadTime" label="创建时间" width="110"
:formatter="formatCreatetime">
</el-table-column>
<el-table-column label="开始处理" width="100" >
<template slot-scope="scope">
<el-button
size="small" type="primary" plain @click="process(scope.row fileId)">开始处理
</el-button>
</template>
</el-table-column>

</el-table>
<!--分页-->
<el-col :span="24" class="toolbar">
<el-pagination background layout="prev, pager, next" @current-change="changePage" :page-size="this.params.size"
:total="total" :current-page="this.params.page"
style="float:right;">
</el-pagination>
</el-col>
</div>
</template>
```

2、数据对象

```
import * as mediaApi from '../api/media'
import utilApi from '@/common/utils';
export default{
  data(){
    return {
      params:{
        page:1,//页码
        size:2,//每页显示个数
        tag:'',
        fileName:'',
        processStatus:'',//处理状态
      }
    }
  }
}
```



```
},
listLoading:false,
list:[],
total:0,
processStatusList:[]
}
}
...
...
```

3、方法

```
methods:{
    formatCreatetime(row, column){
        var createTime = new Date(row.uploadTime);
        if (createTime) {
            return utilApi.formatDate(createTime, 'yyyy-MM-dd hh:mm:ss');
        }
    },
    formatProcessStatus(row,column){
        var processStatus = row.processStatus;
        if (processStatus) {
            if(processStatus == '303001'){
                return "处理中";
            }else if(processStatus == '303002'){
                return "处理成功";
            }else if(processStatus == '303003'){
                return "处理失败";
            }else if(processStatus == '303004'){
                return "无需处理";
            }
        }
    },
    changePage(page){
        this.params.page = page;
        this.query()
    },
    process (id) {
//        console.log(id)
        mediaApi.media_process(id).then((res)=>{
            console.log(res)
            if(res.success){
                this.$message.success('开始处理，请稍后查看处理结果');
            }else{
                this.$message.error('操作失败，请刷新页面重试');
            }
        })
    },
    query(){
        mediaApi.media_list(this.params.page,this.params.size,this.params).then((res)=>{
            console.log(res)
            this.total = res.queryResult.total
            this.list = res.queryResult.list
        })
    }
},
```



```
        })
    }
}
```

4、钩子方法

```
created(){
    //默认第一页
    this.params.page = Number.parseInt(this.$route.query.page||1);
},
mounted() {
    //默认查询页面
    this.query()
    //初始化处理状态
    this.processStatusList = [
        {
            id:'',
            name:'全部'
        },
        {
            id:'303001',
            name:'处理中'
        },
        {
            id:'303002',
            name:'处理成功'
        },
        {
            id:'303003',
            name:'处理失败'
        },
        {
            id:'303004',
            name:'无需处理'
        }
    ]
}
```

3 媒资与课程计划关联

3.1 需求分析

到目前为止，媒资管理已完成文件上传、视频处理、我的媒资功能等基本功能。其它模块已可以使用媒资管理功能，本节要讲解课程计划在编辑时如何选择媒资文件。



操作的业务流程如下：

1、进入课程计划修改页面

2、选择视频

打开媒资文件查询窗口，找到该课程章节的视频，选择此视频。

点击“选择媒资文件”打开媒资文件列表

... 1	原始文件名称 f9d80d12-3571-4fb8-a8ee-a30716bb8991.png	文件名称 0be04ada6a63dbbc1cb25407da08215.png	访问url 0/b/0be04ada6a63dbbc1cb25407da08215/0be04ada6a63dbbc1cb25407da08215.png	标签 java基础	文件大小 181019	处理状态 无需处理	创建时间 2018-05-12 00:12:41	选择
2	5128K_mp4_2.m3u8	38b6c61516e119c2dffaf5fe4b939efaa.m3u8	3/8/38b6c61516e119c2dffaf5fe4b939efaa/38b6c61516e119c2dffaf5fe4b939efaa.m3u8	spring cloud从入门到精通	2068	无需处理	2018-05-12 00:14:38	选择
3	1.avi	809694a6a974c35e3a36f36850837d7c.avi			1968640	处理成功	2018-05-12 15:54:07	选择
4	3.avi	53ac4cca3ddf386c21f4f1cbb4dc9876.avi	5/3/53ac4cca3ddf386c21f4f1cbb4dc9876/53ac4cca3ddf386c21f4f1cbb4dc9876.m3u8		15138560	处理成功	2018-05-13 21:51:47	选择

3、选择成功后，将在课程管理数据库保存课程计划对应的课程视频地址。

在课程管理数据库创建表 teachplan_media 存储课程计划与媒资关联信息，如下：

Field	Type	Comment
teachplan_id	varchar(32) NOT NULL	课程计划id
media_id	varchar(32) NOT NULL	媒资文件id
media_fileoriginalname	varchar(128) NOT NULL	媒资文件的原始名称
media_url	varchar(256) NOT NULL	媒资文件访问地址
courseid	varchar(32) NOT NULL	课程Id

3.2 选择视频

3.2.1 Vue父子组件通信

上一章已实现了我的媒资页面，所以媒资查询窗口页面不需要再开发，将“我的媒资页面”作为一个组件在修改课程计划页面中引用，如下图：

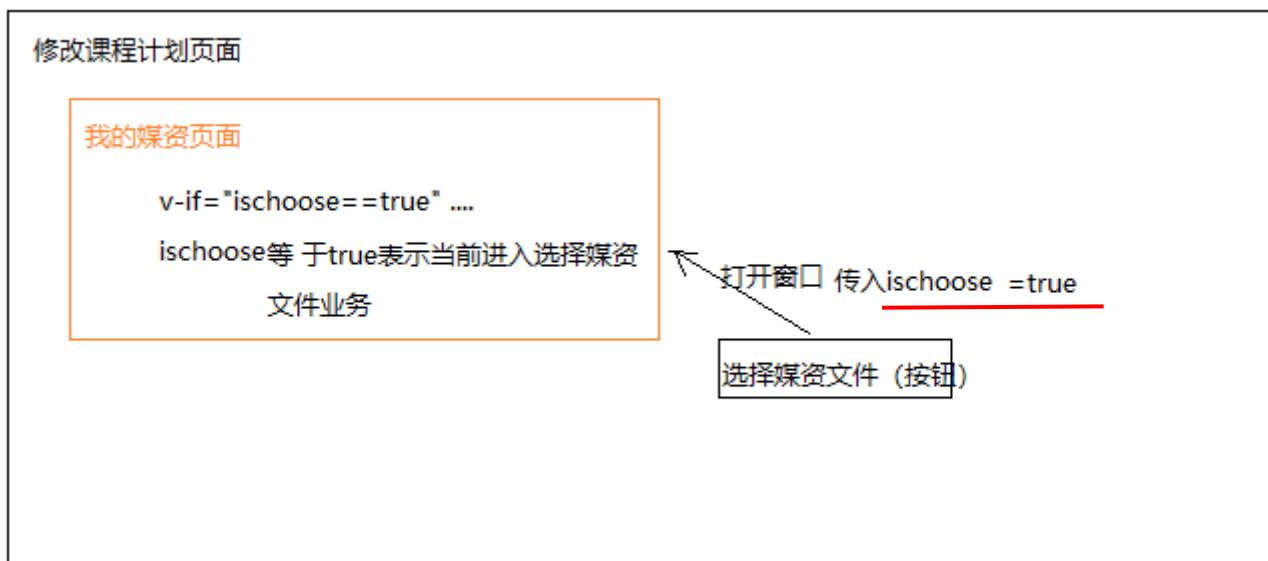


修改课程计划页面为父组件，我的媒资查询页面为子组件。

问题1：

我的媒资页面在选择媒资文件时不允许显示，比如“视频处理”按钮，该如何控制？

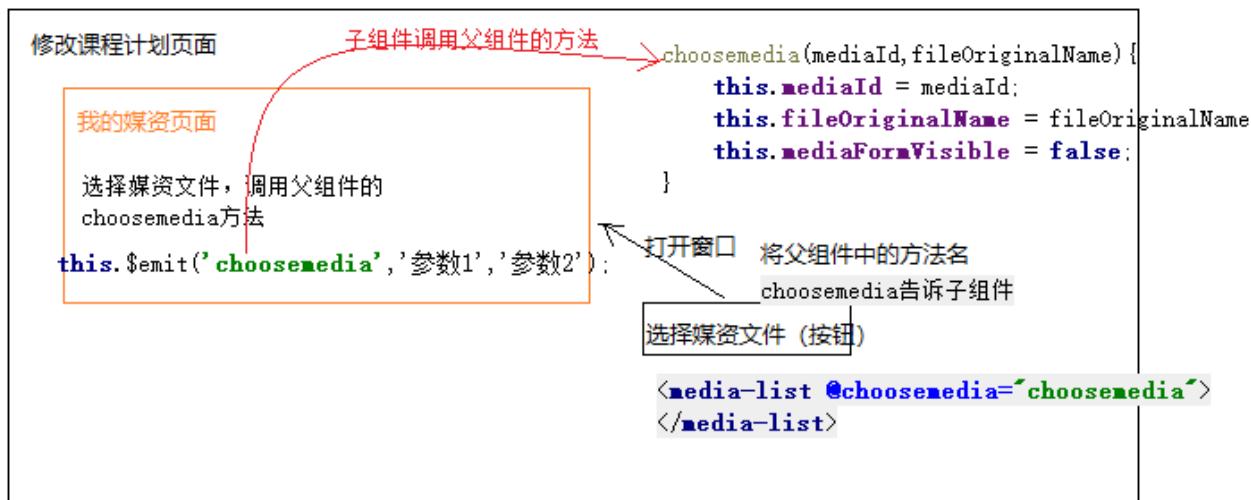
这时就需要父组件（修改课程计划页面）向子组件（我的媒资页面）传入一个变量，使用此变量来控制当前是否进入选择媒资文件业务，从而控制哪些元素不显示，如下图：



问题2：

在我的媒资页面选择了媒资文件，如何将选择的媒资文件信息传到父组件？

这时就需要子组件调用父组件的方法来解决此问题，如下图：



3.2.2 父组件（修改课程计划）

本节实现功能：在课程计划页面打开我的媒资页面。

1、引入子组件

```
import mediaList from '@/module/media/page/media_list.vue';
export default {
  components: {
    mediaList
  },
  data() {
    ....
  }
}
```

2、使用子组件

在父组件的视图中使用子组件，同时传入变量`ischoose`，并指定父组件的方法名为`choosemedia`

这里使用el-dialog 实现弹出窗口。

```
<el-dialog title="选择媒资文件" :visible.sync="mediaFormVisible">
  <media-list v-bind:ischoose="true" @choosemedia="choosemedia"></media-list>
</el-dialog>
```

3、choosemedia方法:接收子组件调用

在父组件中定义`choosemedia`方法，接收子组件调用，参数包括：媒资文件id、媒资文件的原始名称、媒资文件url

```
choosemedia(mediaId, fileOriginalName, mediaUrl){
}
```



4、打开子组件窗口

1) 打开子组件窗口按钮定义

添加“选择视频”按钮：

```
<el-button style="font-size: 12px;" type="text" on-click={() => this.querymedia(data.id)}>选择  
视频</el-button>
```

效果如下：

2、打开子组件窗口方法

定义querymedia方法：

```
methods: {  
    //打开查询媒资文件窗口，传入课程计划id  
    querymedia(teachplanId){  
        this.activeTeachplanId = teachplanId;  
        this.mediaFormVisible = true;  
    },
```

3.2.3 子组件（我的媒资查询）

1、定义ischoose变量，接收父组件传入的ischoose

```
export default{  
    props: ['ischoose'],  
    data(){
```

2、父组件传的ischoose变量为 true时表示当前是选择媒资文件业务，需要控制页面元素是否显示

1) ischoose=true，选择按钮显示

```
<el-table-column label="选择" width="80" v-if="ischoose == true">  
    <template slot-scope="scope">  
        <el-button  
            size="small" type="primary" plain @click="choose(scope.row)">选择</el-button>  
    </template>  
</el-table-column>
```



2) ischoose=false , 视频处理按钮显示

```
<el-table-column label="开始处理" width="100" v-if="ischoose != true">
  <template slot-scope="scope">
    <el-button
      size="small" type="primary" plain @click="process(scope.row fileId)">开始处理
    </el-button>
  </template>
</el-table-column>
```

3、选择媒资文件方法

用户点击“选择”按钮将向父组件传递媒资文件信息

```
choose(mediaFile){
  if(mediaFile.processStatus != '303002' && mediaFile.processStatus != '303004'){
    this.$message.error('该文件未处理，不允许选择');
    return ;
  }
  if(!mediaFile.fileUrl){
    this.$message.error('该文件的访问url为空，不允许选择');
    return ;
  }
  //调用父组件的choosemedia方法
  this.$emit('choosemedia',mediaFile.fileId,mediaFile.fileOriginalName);
}
```

3.3 保存视频信息

3.3.1 需求分析

用户进入课程计划页面，选择视频，将课程计划与视频信息保存在课程管理数据库中。

用户操作流程：

- 1、进入课程计划，点击“选择视频”，打开我的媒资查询页面
- 2、为课程计划选择对应的视频，选择“选择”
- 3、前端请求课程管理服务保存课程计划与视频信息。

3.3.2 数据模型

在课程管理数据库创建表 teachplan_media 存储课程计划与媒资关联信息，如下：



Field	Type	Comment
teachplan_id	varchar(32) NOT NULL	课程计划id
media_id	varchar(32) NOT NULL	媒资文件id
media_fileoriginalname	varchar(128) NOT NULL	媒资文件的原始名称
media_url	varchar(256) NOT NULL	媒资文件访问地址
courseid	varchar(32) NOT NULL	课程Id

创建teachplanMedia 模型类：

```
@Data
@ToString
@Entity
@Table(name="teachplan_media")
@GenericGenerator(name = "jpa-assigned", strategy = "assigned")
public class TeachplanMedia implements Serializable {
    private static final long serialVersionUID = -916357110051689485L;
    @Id
    @GeneratedValue(generator = "jpa-assigned")
    @Column(name="teachplan_id")
    private String teachplanId;

    @Column(name="media_id")
    private String mediaId;

    @Column(name="media_fileoriginalname")
    private String mediaFileOriginalName;

    @Column(name="media_url")
    private String mediaUrl;

    @Column(name="courseid")
    private String courseId;
}
```

3.3.3 API接口

此接口作为前端请求课程管理服务保存课程计划与视频信息的接口：

在课程管理服务增加接口：

```
@ApiOperation("保存媒资信息")
public ResponseResult savemedia(TeachplanMedia teachplanMedia);
```

3.3.4 服务端开发

3.3.3.1 DAO



创建TeachplanMediaRepository用于对TeachplanMedia的操作。

```
public interface TeachplanMediaRepository extends JpaRepository<TeachplanMedia, String> {  
}
```

3.3.3.2 Service

```
//保存媒资信息  
public ResponseResult savemedia(TeachplanMedia teachplanMedia) {  
    if(teachplanMedia == null){  
        ExceptionCast.cast(CommonCode.INVALIDPARAM);  
    }  
    //课程计划  
    String teachplanId = teachplanMedia.getTeachplanId();  
  
    //查询课程计划  
    Optional<Teachplan> optional = teachplanRepository.findById(teachplanId);  
    if(!optional.isPresent()){  
        ExceptionCast.cast(CourseCode.COURSE_MEDIA_TEACHPLAN_ISNULL);  
    }  
    Teachplan teachplan = optional.get();  
    //只允许为叶子结点课程计划选择视频  
    String grade = teachplan.getGrade();  
    if(StringUtils.isEmpty(grade) || !grade.equals("3")){  
        ExceptionCast.cast(CourseCode.COURSE_MEDIA_TEACHPLAN_GRADEERROR);  
    }  
    TeachplanMedia one = null;  
    Optional<TeachplanMedia> teachplanMediaOptional =  
teachplanMediaRepository.findById(teachplanId);  
    if(!teachplanMediaOptional.isPresent()){  
        one = new TeachplanMedia();  
    }else{  
        one = teachplanMediaOptional.get();  
    }  
    //保存媒资信息与课程计划信息  
    one.setTeachplanId(teachplanId);  
    one.setCourseId(teachplanMedia.get courseId());  
    one.setMediaFileOriginalName(teachplanMedia.getMediaFileOriginalName());  
    one.setMediaId(teachplanMedia.getMediaId());  
    one.setMediaUrl(teachplanMedia.getMediaUrl());  
    teachplanMediaRepository.save(one);  
    return new ResponseResult(CommonCode.SUCCESS);  
}
```

3.3.3.3 Controller



```
@Override
@PostMapping("/savemedia")
public ResponseResult savemedia(@RequestBody TeachplanMedia teachplanMedia) {
    return courseService.savemedia(teachplanMedia);
}
```

3.3.4 前端开发

3.3.4.1 API方法

定义api方法，调用课程管理服务保存媒资信息接口

```
/*保存媒资信息*/
export const savemedia = teachplanMedia => {
    return http.requestPost(apiUrl+'/course/savemedia',teachplanMedia);
}
```

3.3.4.2 API调用

在课程视频方法中调用api：

```
choosemedia(mediaId,fileOriginalName,mediaUrl){
    this.mediaFormVisible = false;
    //保存课程计划与视频对应关系
    let teachplanMedia = {};
    teachplanMedia.teachplanId = this.activeTeachplanId;
    teachplanMedia.mediaId = mediaId;
    teachplanMedia.mediaFileOriginalName = fileOriginalName;
    teachplanMedia.mediaUrl = mediaUrl;
    teachplanMedia.courseId = this.courseid;
    //保存媒资信息到课程数据库
    courseApi.savemedia(teachplanMedia).then(res=>{
        if(res.success){
            this.$message.success("选择视频成功")
        }else{
            this.$message.error(res.message)
        }
    })
},
```

3.3.4 测试

1、向叶子结点课程计划保存媒资信息

操作结果：保存成功

2、向非叶子结点课程计划保存媒资信息

操作结果：保存失败



3.4 查询视频信息

3.4.1 需求分析

课程计划的视频信息保存后在页面无法查看，本节解决课程计划页面显示相关联的媒资信息。

解决方案：

在获取课程计划树结点信息时将关联的媒资信息一并查询，并在前端显示，下图说明了课程计划显示的区域。

The screenshot shows the 'Course Plan' section of a web application. At the top, there's a navigation bar with links: 'Course Home', 'Basic Information', 'Course Pictures', 'Course Marketing', 'Course Plan' (which is currently selected), 'Teacher Information', and 'Publish Course'. Below the navigation is a tree view of course plans. The tree has two main branches: 'Java Foundation' and 'Java Advanced'. Under 'Java Foundation', there are nodes for '面向对象' and 'Java O'. Under 'Java Advanced', there are nodes for 'Java Collection' and 'Network Programming'. To the right of the tree, there's a table with several rows, each representing a video media item. The table has columns for '选择视频' (Select Video), '修改' (Modify), and '删除' (Delete). The text '可以在哪个区域显示课程计划对应的视频' (Where can the corresponding video of the course plan be displayed) is overlaid on the tree area.

3.4.2 Dao

修改课程计划查询的Dao:

1、修改模型

在课程计划结果信息中添加媒资信息

```
@Data
@Override
public class TeachplanNode extends Teachplan {

    List<TeachplanNode> children;

    //媒资信息
    private String mediaId;
    private String mediaFileOriginalName;
}
```

2、修改sql语句，添加关联查询媒资信息

添加mediaId、 mediaFileOriginalName

```
<resultMap type="com.xuecheng.framework.domain.course.ext.TeachplanNode" id="teachplanMap" >
    <id property="id" column="one_id"/>
    <result property="pname" column="one_name"/>
    <result property="grade" column="one_grade"/>

    <collection property="children"
    ofType="com.xuecheng.framework.domain.course.ext.TeachplanNode">
```



```
<id property="id" column="two_id"/>
<result property="pname" column="two_name"/>
<result property="grade" column="two_grade"/>
<collection property="children"
ofType="com.xuecheng.framework.domain.course.ext.TeachplanNode">
    <id property="id" column="three_id"/>
    <result property="pname" column="three_name"/>
    <result property="grade" column="three_grade"/>
    <result property="mediaId" column="mediaId"/>
    <result property="mediaFileOriginalName" column="mediaFileOriginalName"/>
</collection>
</collection>
</resultMap>
<select id="selectList" resultMap="teachplanMap" parameterType="java.lang.String" >
    SELECT
        a.id one_id,
        a.pname one_name,
        a.grade one_grade,
        a.orderby one_orderby,
        b.id two_id,
        b.pname two_name,
        b.grade two_grade,
        b.orderby two_orderby,
        c.id three_id,
        c.pname three_name,
        c.grade three_grade,
        c.orderby three_orderby,
        media.media_id mediaId,
        media.media_fileoriginalname mediaFileOriginalName
    FROM
        teachplan a LEFT JOIN teachplan b
        ON a.id = b.parentid
        LEFT JOIN teachplan c
        ON b.id = c.parentid
        LEFT JOIN teachplan_media media
        ON c.id = media.teachplan_id
    WHERE a.parentid = '0'
    <if test="_parameter!=null and _parameter!=''">
        and a.courseid=#{courseId}
    </if>

    ORDER BY a.orderby,
        b.orderby,
        c.orderby
</select>
```

3.4.3 页面查询视频

课程计划结点信息已包括媒资信息，可在页面获取信息后显示：



```
<el-button style="font-size: 12px;" type="text" on-click={ () => this.querymedia(data.id) }>
{data.mediaFileOriginalName}&nbsp;&nbsp;&nbsp;选择视频</el-button>
```

效果如下：

The screenshot shows the 'Teach Plan' tab selected in a course management interface. On the left, a tree view lists course modules: 'java进阶' (including 'Java集合' and '网络编程'), 'Java基础' (including '面向对象' and 'JavaIO'). On the right, a list of media files is shown, with the first item, '01Vue基础- MVVM模式讲解.avi', highlighted by a red box. Each media item has three actions: '选择视频' (Select Video), '修改' (Edit), and '删除' (Delete).

选择视频后立即刷新课程计划树，在提交成功后，添加查询课程计划代码：this.findTeachplan()，完整代码如下：

```
choosemedia(mediaId,fileOriginalName,mediaUrl){
    this.mediaFormVisible = false;
    //保存课程计划与视频对应关系
    let teachplanMedia = {};
    teachplanMedia.teachplanId = this.activeTeachplanId;
    teachplanMedia.mediaId = mediaId;
    teachplanMedia.mediaFileOriginalName = fileOriginalName;
    teachplanMedia.mediaUrl = mediaUrl;
    teachplanMedia.courseId = this.courseid;
    //保存媒资信息到课程数据库
    courseApi.savemedia(teachplanMedia).then(res=>{
        if(res.success){
            this.$message.success("选择视频成功")
            //查询课程计划
            this.findTeachplan()
        }else{
            this.$message.error(res.message)
        }
    })
},
```