## 环境

Java 1.8.0\_191

Maven 3.6.3

Node v12.19.0

mysql Ver 8.0.22

# 后端

## 注解

@PathVariable**是spring3.0的一个新功能：接收请求路径中占位符的值**

@RequestBody 前端使用json传递数据 ，将json封装到对象中

@RequestParam接收的参数是来自HTTP请求体或请求url中

Mybatis plus 的注解

[@TableLogic表逻辑处理注解（逻辑删除）](https://www.cnblogs.com/HQ0422/p/10921462.html)

字段自动填充

@TableField(fill = FieldFill.***INSERT***)

@TableField(fill = FieldFill.***INSERT\_UPDATE***)

还需创建自动填充处理类

## Mybatisplus主键策略

* AUTO 数据库ID自增
* INPUT 用户输入ID
* ID\_WORKER 全局唯一ID，Long类型的主键
* ID\_WORKER\_STR 字符串全局唯一ID
* UUID 全局唯一ID，UUID类型的主键
* NONE 该类型为未设置主键类型

如：

@TableId(value = "id", type = IdType.ID\_WORKER\_STR)

private String id;

## 插件

### MyBatisPlus SQL 执行性能分析插件

/\*\*

\* SQL 执行性能分析插件

\* 开发环境使用，线上不推荐。 maxTime 指的是 sql 最大执行时长

\*/

@Bean

@Profile({"dev","test"})// 设置 dev test 环境开启

**public** PerformanceInterceptor performanceInterceptor() {

PerformanceInterceptor performanceInterceptor = **new** PerformanceInterceptor();

performanceInterceptor.setMaxTime(1000);//ms，超过此处设置的ms则sql不执行

performanceInterceptor.setFormat(**true**);

**return** performanceInterceptor;

}

### MyBatisPlus逻辑删除插件

/\*\*

\* 逻辑删除插件

\*/

@Bean

**public** ISqlInjector sqlInjector() {

**return** **new** LogicSqlInjector();

}

### MyBatisPlus分页插件

/\*\*

\* 分页插件

\*/

@Bean

**public** PaginationInterceptor paginationInterceptor() {

**return** **new** PaginationInterceptor();

}

## Swagger 的优势

* 支持 API 自动生成同步的在线文档：
* 提供 Web 页面在线测试 API

常用注解：

- @Api()用于类；

表示标识这个类是swagger的资源

- @ApiOperation()用于方法；

表示一个http请求的操作

- @ApiParam()用于方法，参数，字段说明；

表示对参数的添加元数据（说明或是否必填等）

- @ApiModel()用于类

表示对类进行说明，用于参数用实体类接收

- @ApiModelProperty()用于方法，字段

表示对model属性的说明或者数据操作更改

## 私有构成函数—例如统一返回结果对象中有用到

通过类的私有构造函数来强调类的不可实例化

例如在实际的应用中，工具类，或者整个工程的常量类，在类中只有静态常量

## 依赖传递

B 包含c的依赖 a想用b和c中的类 a引用b的依赖就可以 这就是依赖传递 如果a也引用c 就会出现重复依赖 可能会出问题

## Lombok 不建议使用

1 开发工具安装插件，2 引入依赖

缺点

1**强迫队友**

Lombok插件的使用，要求开发者一定要在IDE中安装对应的插件。不仅自己要安装，任何和你协同开发的人都要安装。

如果有谁未安装插件的话，使用IDE打开一个基于Lombok的项目的话会提示找不到方法等错误，导致项目编译失败。

**2代码可调试性降低**

**3我们需要升级到某个新版本的JDK的时候，若其中的特性在Lombok中不支持的话就会受到影响。**

## 统一异常处理

全局异常处理

指定异常

自定义异常

## 统一日志处理

OFF、FATAL、ERROR、WARN、INFO、DEBUG、ALL

默认情况下，spring boot从控制台打印出来的日志级别只有INFO及以上级别，可以配置日志级别

### Logback

spring boot内部使用Logback作为日志实现的框架。

Logback和log4j非常相似，如果你对log4j很熟悉，那对logback很快就会得心应手。

1控制 控制台的日志级别

2 把日志输出到文件中

日志工具类 ？？？？？？

## 跨越

方法一 解决 在controller 加注解@CrossOrigin

## Nginx

是一个高性能的 HTTP 和 反向代理 服务器

1. 请求转发
2. 负载均衡

常见的算法

第一种：轮询

第二种：随机

第三种：加权轮询 给配置高的配置更多任务

1. 动静分离

Window 版本 在所在目录下 cmd 启动 nginx.exe

重启的命令 nginx -s reload

关闭的命令 nginx -s stop （关闭cmd窗口无效）

## 对象存储oss

为了解决海量数据存储与弹性扩容，项目中我们采用云存储的解决方案- 阿里云OSS。

## 从配置文件读取常量

使用@Value读取application.properties里的配置内容

用spring的 InitializingBean 的 afterPropertiesSet 来初始化配置信息，这个方法将在所有的属性被初始化后调用。

## MultipartFile

如果你想用spring框架来实现项目中的文件上传功能，则MultipartFile可能是最合适的选择

## EasyExcel

写：

1： 引入依赖 ，

2 创建excel映射类 ，

1. EasyExcel.write(文件路径, 映射类.class).sheet("test").doWrite(数据集合);

读：

1： 引入依赖 ，

2 创建excel映射类 ，

3 创建监听器类 extends AnalysisEventListener<映射类> 读的操作都在监听器中执行，一行一行的读

1. EasyExcel.read(文件路径, 映射类.class, new 监听器()).sheet().doRead();

# 课程分类管理模块

主要用到阿里巴巴的EasyExcel实现对Excel操作

## 新增课程

## 课程分类 树图分类

1. [       {
2. id: 1,
3. label: 'Level one 1',
4. children: [{
5. id: 4,
6. label: 'Level two 1-1',
7. }]
8. }, {
9. id: 2,
10. label: 'Level one 2',
11. children: [{
12. id: 5,
13. label: 'Level two 2-1'
14. }, {
15. id: 6,
16. label: 'Level two 2-2'
17. }]
18. }
19. ]

前端所需要的数据格式 ，后端根据格式写个相应的封装类

# 课程管理模块

### 数据库如何选择float，double，decimal

Float double 准确度不高

故价格一般用decimal类型

## 表er图

图示

描述已自动生成

## 整合富文本编辑器Tinymce

图形用户界面, 文本, 应用程序

描述已自动生成

Tinymce中的图片上传功能直接存储的是图片的base64编码，因此无需图片服务器

## 注意vue和后端数据的交付方式

## Maven加载机制 把java文件夹下的java类型进行编译，如何是其他类型，不会加载

文本

中度可信度描述已自动生成

如图 会报错 显示加载不到这个方法 id 就是方法名

解决方案：通过配置的方式

1 pom.xml 配置

<build>

<resources>

<resource>

<directory>src/main/java</directory>

<includes>

<include>\*\*/\*.xml</include>

</includes>

<filtering>false</filtering>

</resource>

</resources>

</build>

2 application.properties

#配置mapper xml文件的路径

mybatis-plus.mapper-locations=classpath:com/matt/edu/mapper/xml/\*.xml

## 阿里视频点播vod

Video on Demand vod

根据视频id 获取视频播放地址

根据视频id 获取视频凭证 凭证 是获取加密的视频播放地址

根据视频id 上传或删除阿里云视频点播

引入依赖时候 某个jar包不能从中央仓库下载 报错

解决方法： 去官网找Jar包后下载到本地　　并用maven命令安装到本地仓库

mvn install:install-file -DgroupId=com.aliyun -DartifactId=aliyun-sdk-vod-upload -Dversion=1.4.11 -Dfile=aliyun-java-vod-upload-1.4.11.jar -Dpackaging=jar -DgeneratePom=true

# 微服务

微服务：把一个项目按业务划分为多个服务，每个服务独立运行，占用独立线程

## 微服务与单体架构区别

（1）单体架构所有的模块全都耦合在一块，代码量大，维护困难。

         微服务每个模块就相当于一个单独的项目，代码量明显减少，遇到问题也相对来说比较好解决。

（2）单体架构所有的模块都共用一个数据库，存储方式比较单一。

         微服务每个模块都可以使用不同的存储方式（比如有的用redis，有的用mysql等），数据库也是单个模块对应自己的数据库。

（3）单体架构所有的模块开发所使用的技术一样。

         微服务每个模块都可以使用不同的开发技术，开发模式更灵活。

## Nacos  服务注册

不支持yml格式的配置文件，只能有properties格式的配置文件

Nacos 是阿里巴巴推出来的一个新开源项目

（2）常见的注册中心：

1. Eureka（原生，2.0遇到性能瓶颈，停止维护）

2. Zookeeper（支持，专业的独立产品。例如：dubbo）

3. Nacos

### Nacos的使用

（1）下载地址和版本

下载地址：<https://github.com/alibaba/nacos/releases>

下载版本：nacos-server-1.1.4.tar.gz或nacos-server-1.1.4.zip，解压任意目录即可

（2）启动nacos服务

- Linux/Unix/Mac

启动命令(standalone代表着单机模式运行，非集群模式)

启动命令：sh startup.sh -m standalone

- Windows

启动命令：cmd startup.cmd 或者双击startup.cmd运行文件。

访问：<http://localhost:8848/nacos>

用户名密码：nacos/nacos

（3）服务注册

1加依赖

<!--服务注册-->

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>

</dependency>

2配置# nacos服务地址 application.properties

spring.cloud.nacos.discovery.server-addr=127.0.0.1:8848

3 **添加Nacos客户端注解**

在客户端微服务启动类中添加注解

@EnableDiscoveryClient

最后nacos的web可视化页面 服务列表可以看到被注册的微服务

## Feign 服务调用

1、需求

删除课时的同时删除云端视频

2、在service模块添加pom依赖

<!--服务调用-->

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-openfeign</artifactId>

</dependency>

3、在调用端的启动类添加注解

@EnableFeignClients

4、创建包和接口

创建client包

@FeignClient注解用于指定从哪个服务中调用功能 ，名称与被调用的服务名保持一致。

@GetMapping注解用于对被调用的微服务进行地址映射。

@PathVariable注解一定要指定参数名称，否则出错

@Component注解防止，在其他位置注入CodClient时idea报错

package com.guli.edu.client;

@FeignClient("service-vod")

@Component

public interface VodClient {

@DeleteMapping(value = "/eduvod/vod/video/{videoId}")

public R removeVideo(@PathVariable("videoId") String videoId);

}

5、调用微服务

在调用的地方注入 VodClient

即可调用vodClient.removeVideo(videoSourceId);

# 前端

两个入口 index.html 和main.js

## debugger 设置断点调试

## 前端遇到的问题

### 布局里调用方法时一定要加（）



不加这个 url后缀变成 [object%20PointerEvent]/10 一直访问不到后台

加了才会把参数传过去

# Bug

## 有些文件未找到引起报错问题

可能是编译问题 原因是 springboot正常只会编译java文件到classes文件下

需要再pom文件加下面配置

<resource>

<directory>src/main/java</directory>

<includes>

<include>\*\*/\*.xml</include>

<include>\*\*/\*.yml</include>

<include>\*\*/\*.properties</include>

</includes>

<filtering>false</filtering>

</resource>

<resource>

<directory>src/main/resources</directory>

<includes>

<include>\*\*/\*.yml</include>

<include>\*\*/\*.properties</include>

<include>\*\*/\*.xml</include>

</includes>

<filtering>false</filtering>