## 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 ，讲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

## 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

## Nignx

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

## 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。

# 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