Glimmer Backend Recruit 👑

前言：嗯嗯嗯~...万众瞩目的附加题来了，看来是前两个题收力收猛了，最后一个题了必须放开了写了。光是看第一个ReadMe就知道工程量很大啊……没事我们马上开始！

前置引导:

part1：你了解Spring Framework吗?

嗯嗯...不知道，你知道的接下来是笔记时间！、

笔记part：（前置了解）

（首先是新出现的很多名词，不搞懂那不是我的风格）

事前声明：由于这次的新名词太多，手打的笔记中可能出现错误，不必全然相信。

- **一个汇总**，新出现的名词有：Spring Framework，**面向切面编程**，**控制反转**，**依赖注入**，**解耦**，**Maven**，**Bean**，**XML**，**注解开发**（这个好像了解过？），`IOC`容器，表现层、业务逻辑层、数据访问层三层架构，MyBatis框架，Mapper映射，MVC模式，CRUD操作，apifox。

很好也...不是很多嘛。我们先都初步了解一下，看看能不能有一个大体的印象。(感觉7天学不完呢(^_^)~)

1. Spring Framework：其实readme中介绍的挺多的，主要不理解的是那些个名词了，这个后面查了就ok了。
2. 面向切面编程：传统的编程中业务的功能通常是耦合的，面向切面将一些公有的功能（日志，监控，事务管理）从主体代码中分离出来，称为**切面。**（更深入的了解放在后面吧）
3. 控制反转：看了下，很好理解，具体解释就是当代码A需要用到对象B时如果使用控制反转，它就不在自己的代码中创建B而是向一个外部的容器申请对象B，也是一个**降低代码耦合度**的东西。
4. 依赖注入：在搜索控制反转时有文章提到**控制反转的实现**用到了依赖注入，这里具体来看一下。它就是把控制反转中创建B对象这个过程转移到外部的一个方法。（更深入的了解放在后面吧）
5. 解耦：突然发现这不算个新名词了，哈哈(^_^)。
6. Maven：主要功能有，依赖管理（化繁为简），多模块管理（简化子模块的管理），定义构建生命周期（提高开发效率），插件机制（拓展功能）。看它说的很高大上，但应该用起来会是个很接地气的东向呢~
7. Bean：这就是上文提到的外部容器了，我们可以把任意需要的Bean注入到它里面，帮助实现依赖注入和控制反转。当然这个容器本身也需要管理和配置。
8. XML：看来也是个辅助工具，它可以标记电子文件使其具有结构性，其强大在于客户可以根据需要自定义标记和结构，可以适应不同的文件要求。当然了还有很强的平台不依赖性（自己取的...）
9. 注解开发：这个在以前就有题里要求了解了，后面有用到再相信讲讲。
10. IOC容器就是帮你写好了的控制反转，依赖注入，帮你快速实现这两个步骤。很好理解吧。
11. 表现层、业务逻辑层、数据访问层三层架构：看来是一个常用的编程的框架，表现层负责接收和输出，业务逻辑层负责处理，数据访问层负责与数据库进行交流和操作。那其实它只是一种比较方便的框架，并不是一种必需品？可以让开发的时候更有条理。
12. MyBatis框架：是一个与数据库交流的用的框架而且允许我们自己编写SQL语句，就，灵活性很高吧。当然，它也可以很好的和Spring框架联动，至于怎么联动，emm，还有待学习。
13. Mapper映射：这是MyBatis框架中的一个功能，可以把Java里的方法和对数据库的操作对应起来，方便对数据库的调用。
14. MVC模式：一种设计模式，分为模型，试图，控制器三部分。
15. CRUD操作：时管理数据库时用到的四个操作，create，read，update，delete。

16. apifox：是一个方便的接口的协作平台，可以方便的进行接口设计，接口调试，自动化测试接口等等。

好了，我确实很喜欢这样整体的学习方式，至少这整个框架是什么样大体是搞清楚了。

当然每个东西具体怎么去使用还是一点都不知道，我们且行且学吧。

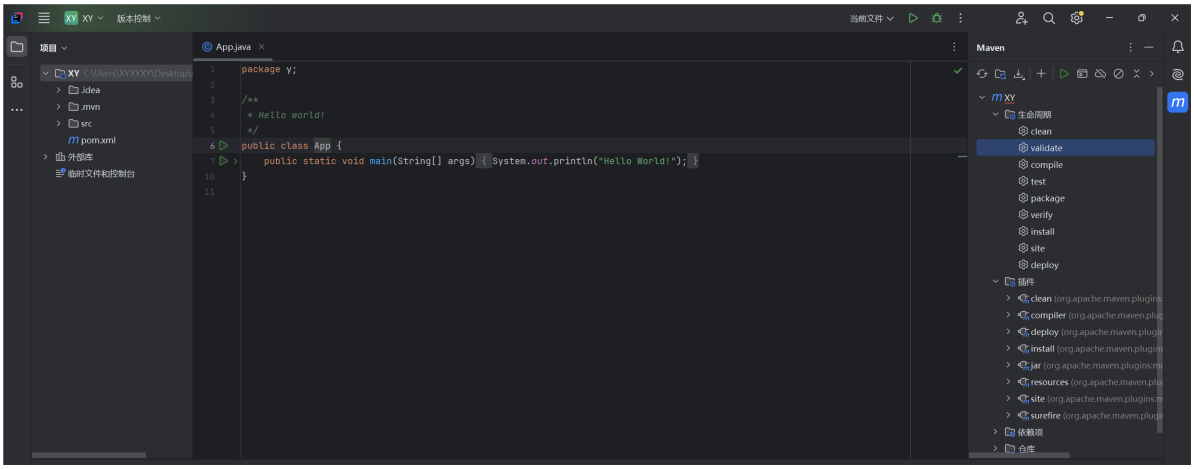
在Maven中引入Spring的相关依赖，推荐使用注解开发，可以试试用`IOC`容器创建和管理Bean哦！

很好，开头就遇到了第一个问题，确实了解了Maven和Spring framework的功能，但它们到底是什么，以一个什么底层形态来实现自己的功能？

不过也不是什么大问题，查阅了下，Maven需要去安装，然后引入Spring依赖就可以了。

首先是下载安装，配置下环境嗯嗯(跟着视频学一下MAVEN怎么用ing)

经过长时间的尝试，终于是创建了第一个maven项目，其中遇到的问题包括但不限于generate时找不到pow文件，Idea中的maven更新设置，不过也是稍微变通一下就没问题了。接下来是一些maven使用方面的笔记。（附上截图）



笔记part：

1. 生命周期的功能：
    1. clean：**清理项目**，将编译生成的字节码文件和jar包文件删除
    2. vaildate：**验证**项目是否正确，并且所有必要的信息是可用的
    3. compile：**编译**项目源代码，生成字节码文件
    4. test：单元**测试**，会执行我们test目录下的test用例
    5. package：**打包**项目，把编译生成的字节码文件和其他的资源文件一起打包生成jar包或是war包（其实在执行package时也会先编译和测试一下，没问题后才会打包）
    6. verify：**检查**打包生成的jar包是否正确
    7. install：把打包生成的jar包或是war包**安装**到本地仓库
    8. deplay：把打包好的jar包**上传**到远程仓库里
    9. site：**生成**项目站点文档
2. 依赖管理：
    1. provided：编译时需要，运行时不需要
    2. test：依赖只在测试时需要（不会被打包到jar包中）
    3. compile：编译和运行时都需要
    4. runtime：运行时需要，编译时并不需要

5. system：本地提供的依赖，此时还需要一个systempath来指定该去哪里找到这个依赖（不过会导致可移植性变差，最好还是上传私服）
6. import：导入其它pom文件里的依赖，但不会实际引入依赖

3. 依赖添加：

   搜索然后添加：在官网找到下载路径，复制到pom的dependency中就可以了（第一次下会爆红，刷新一下，它会帮你下载下来）

4. 依赖传递：

   好像不是一个我需要关心的问题，别人已经帮你搞好了（只有是compile的依赖会被传递）

5. 依赖冲突：

   如果我们的两个依赖分别依赖了一个依赖的不同版本，这个时候idea会选择最短路径优先，然后是先声明优先。

   当然也可以手动控制依赖，用exclusions标签来排除不需要的依赖，用optional标签来标记一个依赖是可选的

好多东西，好多/(ㄒoㄒ)/~~,现在我们来尝试导入Spring相关依赖吧

问题在于我也不知道到底需要用到多少spring系列的依赖，所以引入了挺多，反正含有那个Bean就OK，然后又到了我们的学习时间，就学怎么用IOC容器创建和管理Bean。

在这期间遇到了一个令我为之惊叹的问题，因为创建新项目的时候它不会给你把啥的打出来，我去抄格式的时候一直还再套了一层然后一直报错说springframework不可识别，很好的错误，想抽自己两巴掌。

然后的问题就是：该把方法的实现写在哪里，怎么创建Beans，创建了Beans后怎么调用，怎么告诉电脑哪里有Bean需要从创建，一一解决吧。

这里总结一下语法，配置，负责扫描创建Bean

```
package config;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
@Configuration
@ComponentScan
public class AppConfig {
}
```

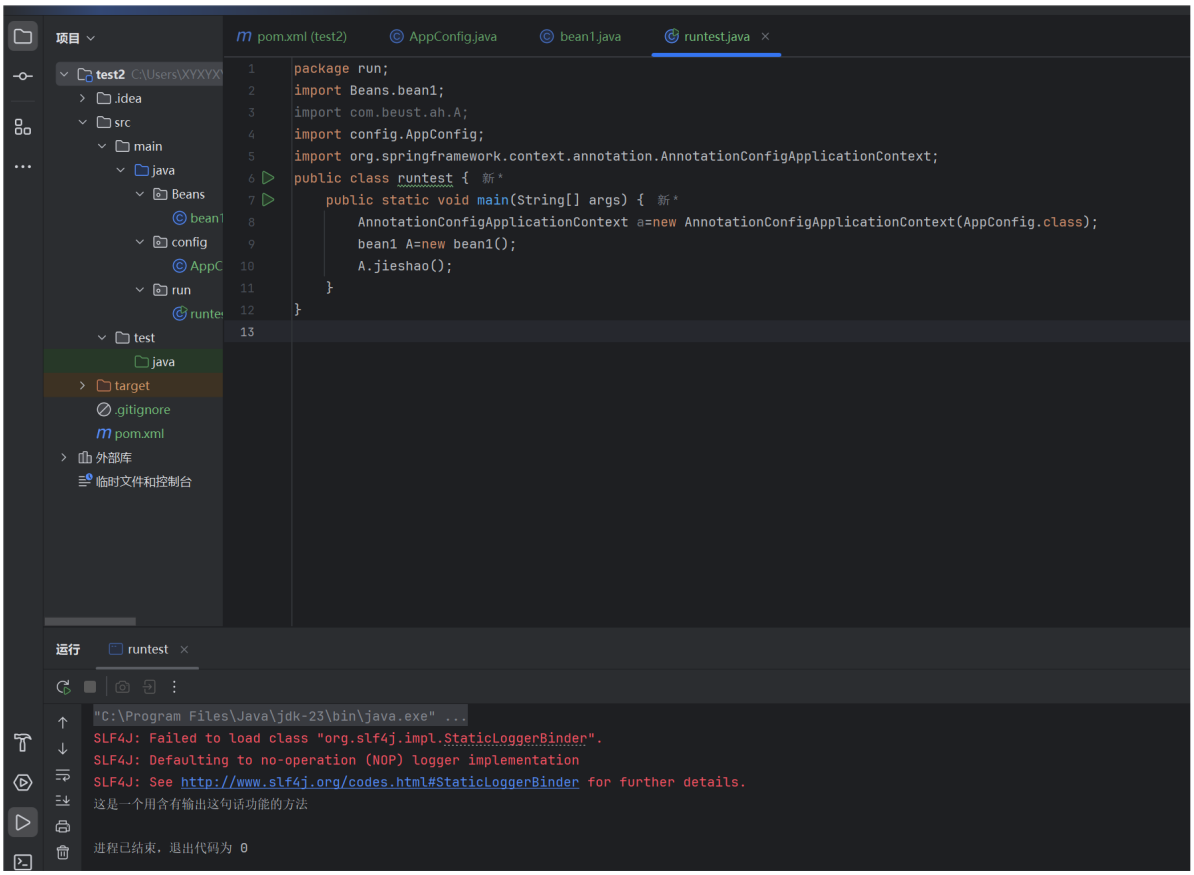
bean中具体的功能

```
package Beans;
import org.springframework.stereotype.Service;
@Service
public class bean1 {
    public void jieshao(){
        System.out.println("这是一个用含有输出这句话功能的方法");
    }
}
```

运行

```java
package run;
import Beans.bean1;
import com.beust.ah.A;
import config.AppConfig;
import
org.springframework.context.annotation.AnnotationConfigApplicationContext;
public class runtest {
    public static void main(String[] args) {
        AnnotationConfigApplicationContext a=new
AnnotationConfigApplicationContext(AppConfig.class);
        //此语句完成Bean创建
        bean1 A=new bean1();
        A.jieshao();
    }
}
```

当然Bean中还有很多其它用法，暂时了解到这这里。附截图（这是我唯一一个能上传的代码了，虽然也不成样子，但有总比没有强！）（文件夹名叫Bean创建与管理）



part2：在Maven中引入MyBatis与数据库驱动的相关依赖，并用MyBatis框架完成对数据库的CRUD操作吧！

等等，这个选做任务，试着写代码就完全变成对着抄了，感觉还不是很能理解我们先跳过吧...如果有机会....我们再了解。

好了恭喜恭喜我们可以开始愉快的task1了，结果还有个task0.....前摇太太太长了！

task0：前置知识掌握

- 网络的知识：我记得初中微机课还讲过，但...忘完了

  1. **ip**，不简称是互联网协议的地址，功能类似于你的家庭地址（而且从某种意义上它还真能当家庭地址），可以把数据准确的传到你的设备。它也分两种类型，IPv4和IPv6，前者由四个数字组成，每个数字在0到255之间，后者由八组16进制的数字组成（192.168.1.1；2001：0db8：85a3：0000：0000：8a2e：0370：7334）
  2. **端口号**：是用于区分不同应用程序或者服务的一串数字。可以帮你准确找到特定的应用，比如通常80端口（HTTP）443端口（HTTPS）...
  3. **URL**:通俗来讲是网址，也是统一资源定位符。结构包含：协议（http，https），主机名（域名或者IP地址），端口号(这个视情况)，路径，查询参数等部分。
  4. **HTTP**:不缩写为超文本传输协议，使用于在Web上传输超文本的一种协议。它规定了很多怎么传输的方法。其中，状态码200代表请求成功，400代表客户端请求有错，500代表服务器内部错误。

- 下面的准备有不少可以结合着apifox上写的内容来理解

  1. **接口**：在软件开发里呢，接口定义了一组方法，参数和返回值，使得不同的模块可以用接口实现不同的功能。所以，接口亦是不同软件交互的一个约定。在前端和后端来讲，接口就是进行数据交互的通道。

  2. **请求参数**是什么：是客户端发给服务器，用于进行特定操作的数据。

     请求头（header）：是一个装了关于请求的信息的载体

     请求头可以包含的信息有：Content-Type（指定了请求体的格式），Authorization（身份验证信息），User-Agent（客户端信息）等。

     请求体的格式主要有：**JSON**:类似于key，value,这种形式，一般用于数据交互；**form-data格式**：主要用于上传文件。

  3. **响应参数**：是服务器返回给客户端的数据，响应客户端的请求。

     响应头header可以包含：Content-Type（同上），Content-Lenth（响应体长度），Set-Cookie（设置cookie，~~cookie：存储在用户计算机上的小文件，通常包含一些标识性的数据，可以带来一些方便~~）等信息

     相应的格式：JSON格式，binary格式（照片，音频，视频的二进制数据）

  4. emm...它是统一的，但统一不应该是有助于开发分离的吗？前端只需照接口的要求从接口获取数据，后端只需要专注于逻辑和稳定性，本质上没有什么冲突。

  5. 先说说什么是**rest风格接口**，是一种以资源为核心的框架风格，每个资源都有一个对应的URL，且使用统一的接口，标准的HTTP方法，以及（200，400，500）状态码...（~~虽说子解子跟没子解的区别就是子解子~~）好在了解了就知道它是怎么规范的了

     - 资源用URL表示，每个资源对应唯一的URL
     - 使用HTTP方法（get，post，put，delete）
     - 状态码反映结果
     - 返回数据多为JSON

  6. **怎么处理异常反馈给前端**：响应参数中给出特定的的错误信息和状态码。（上文都提到了）

- unix时间戳的概念

  1. unix时间戳是什么：是从格林威治时间1970年01月01日00时00分00秒（也叫协调世界时）起至现在的总秒数

     unix时间戳和我们平时的时间加时区的方式有什么关系：它们都可以用来表示时间，但总的功能有区别，一个是为了让不同时区的人们更好理解时间，一个是为了更好的统一时间。

     unix时间戳有时区的概念吗？：你看这个时间戳的定义就知道肯定是没有的。

- 相关注解的作用

  1. @Slf4j：由Lombok提供的注解，会在编译时自动生成一个名字log，类型为org.slf4j.Logger的日志对象，简化了日志的代码编写

2. @Configuration：标注会告诉系统这是一个Spring的配置类，可以在类中使用@Bean标记Bean容器，以及配置Spring框架。
3. Controller用@Controller或者@RestController，前者处理HTTP请求返回视图，后者直接返回数据（JSON或是XML格式的）

   Service用@Service标记，即可将服务类交给Spring容器管理

   Mapper用@Mapper注解（使用MyBatis的话），标记访问数据对象类
4. 这个注解相当于@Controller和@ResponseBody，其功能就是直接返回数据，而不需要每处都添加@ResponseBody来注明返回的是数据而不是试图。
5. @RequestMapping，用于映射请求的URL到一个方法上，它既可以用在类级别上，也可以用在方法级别上，但在后者上还细化了具体的请求方法（GET，POST）等，有了这个东西，Spring框架可以将HTTP的请求分发到正确的处理方法上。
6. 有四个相关注解：

   @GetMapping:处理HTTP GET请求，读取

   @PostMapping：处理HTTP POST请求，创建

   @PutMapping：处理HTTP PUT请求，更新

   @DeleteMappong：处理HTTP DELETE，删除
7. @Autowired：字面意思是自动组装，那确实，是**自动装配注解**的作用，它可以自动查找被标注的属性类型相匹配的Bean，并注入到该属性中（有点没get到是什么意思），一般呢放在成员变量，构造函数上。

- 了解一下SHA256加密方式
  1. **好处**是安全性高，和难反推，因为他加密转换到的哈希值很难撞，且很难逆转。**为什么用它**，因为它安全性高，且我们需要保护数据安全。

问题很多，具体怎么使用用到了再具体学习吧。

---

task1（弃坑了）

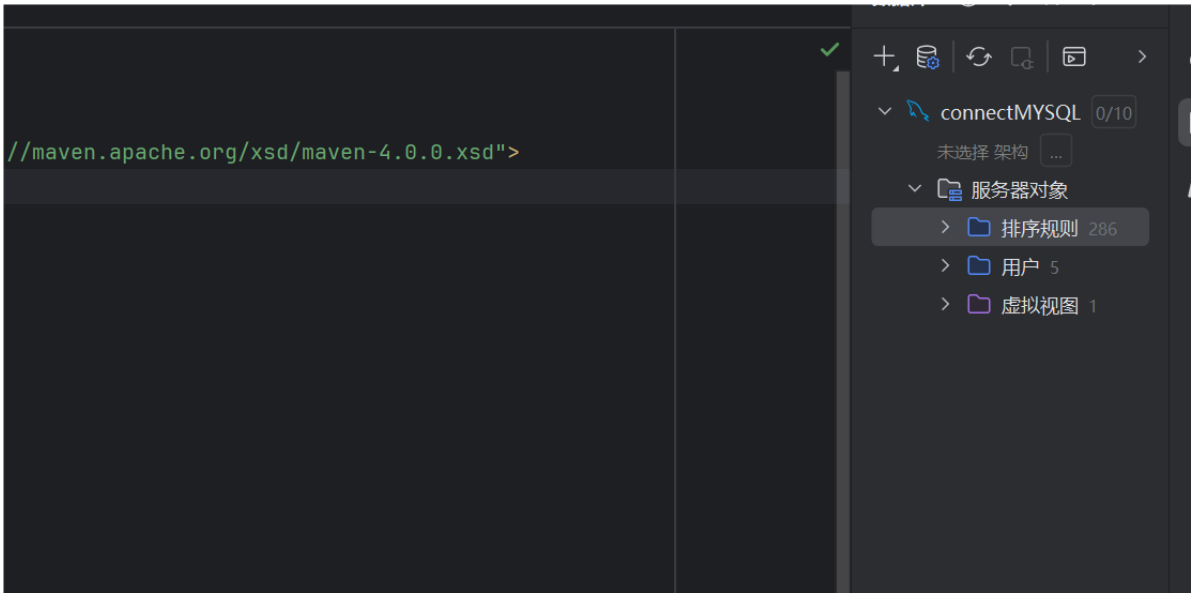
到这里才开始task1...

什么？连接自己的数据库，我哪来的数据库...不确定，再看看。等等这个不会是要Ultimate的吧，忍痛试用30天...

(30分钟的升级后)

开始申请MYSOL数据库，一开始下错了只下载了server，又陆陆续续搞了很久反正最后是成功了

最后的最后又进行了些莫名其妙的尝试，然后应该是成功了，附上截图 



跟着教程clean然后compile了一下，好像是缺少哪个依赖嘛，里面的用不上外面pom得依赖，不过莫名加上就好了，没太懂...

接下来是正式开始。

（又是很长一段时间之后）

额...学长如果是你那时候刚看到这个东向能搞的明白吗?

```
@RequestMapping("/alert")
public class AlertTransactionController {
    @Autowired
    private AlertTransactionService alertTransactionService;

    /*
    GetAlert 函数用于获取报警信息，注册的路由url为: /alert
    | 请求参数   |          |                                          |
    | ---------- | -------- | ---------------------------------------- |
    | 参数名     | 数据类型 | 说明                                     |
    | startDate  | string   | 需要搜索的开始日期，全部搜索输入""       |
    | endDate    | string   | 需要搜索的结束日期，全部搜索输入""       |
    | cameraName | list     | 需要搜索的摄像头名称列表，全部搜索输入[""] |
    | type       | list     | 需要搜索的检测类型列表，全部搜索输入[""]   |
    返回参数：
      data里面除了标准的status和message外，使用infos字段返回报警信息列表，infos里面包含了报警信息的所有字段(主键id除外),这些字段来源于Alert结构体
    有["caId"]:是alert.CaId，["type"]:是alert.Type，["alertTime"]:是alert.AlertTime，["videoPath"]:是alert.PathVideo，["photoPath"]:是alert.PathPhoto
    status 0:成功 1:请求失败 9.系统错误
    */
    @GetMapping 新*
    public Result<StatusVO> GetAlert(@RequestBody GetAlertDTO getAlertDTO){


    }
```

我试着去弄明白这些东西是什么，也不算太失败，大约也是搞明白了。

唯有**很多点不懂**，infos字段怎么用？这个方法是怎么完成搜索这个步骤得？怎么在同时返回status，message的时候一起返回infos字段...虽说不应该随便放弃的，但估计现在继续搞这个性价比很低，先搁置在这里吧。、

其实到这了还挣扎着去学了下Apifox，但比起没学区别就在于学了。

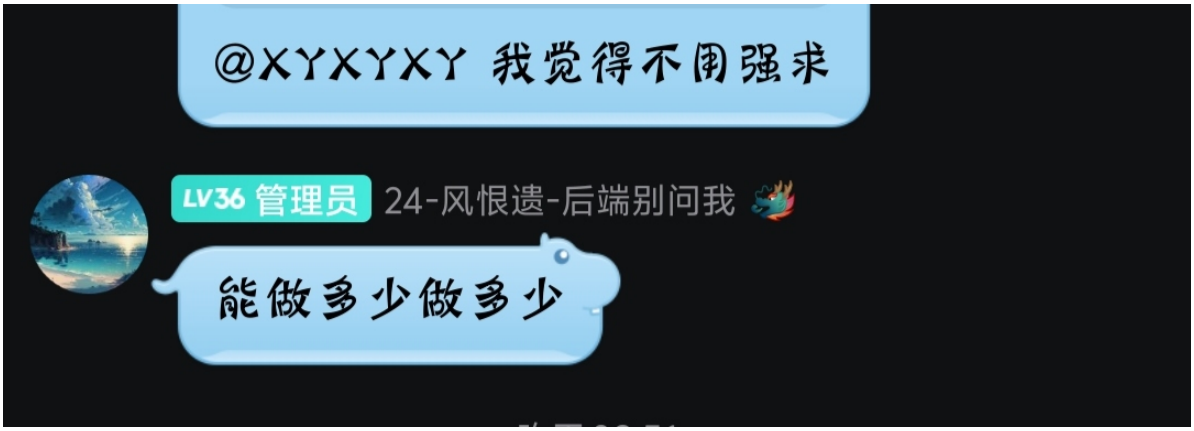
题也没做啥，我都不知道代码那边能传什么上去/(ㄒoㄒ)/~~

.....

.....

...

..



投降了，很难绷啊...