后端代码规范

Controller层

1. 每个Controller的接口必须加上@ApiOperation，并且在里面写上接口的用处，用于Swagger自动生成接口，也可以让其他人更好看懂接口的作用。
2. 接口统一返回CommonResult类，通过返回CommonResult中的静态方法success或者failed指示是否调用是否成功。
3. 不许在Controller的代码中try catch异常，实在想try ，要在catch代码块里把捕获到的异常再抛出去给统一异常处理类。
4. **尽量写遵循Restful风格的接口：**
    - 简单的增删改查（参数比较少的情况）使用@PathVariable映射接口：如GET /user/{id} 表示获取指定用户的信息；POST /user/{id} + 传入VO类 UserParam 修改指定用户的信息，
    - 如果POST请求传入的参数比较多，一定要写一个VO来接收这些参数，在接口的参数前加上@RequestBody注解
    - 一定要考虑前端瞎传参数和安全的情况，所以最好用@Validated修饰接口的参数，以及在VO类中用@NotEmpty等参数验证注解来保证传入的参数合法
    - 接口传入的VO类该项目统一以XxxParam命名
5. 返回分页数据的接口如果成功，统一返回 CommonResult.success(CommonPage.restPage(**xxxpage**));（其中的xxxpage是从service层返回的Page 对象）

Service层

1. 所有的service类必须加上以javadoc形式的注释（/** xxx **/这样的），表示该方法的作用
2. 接口方法命名尽量合理即可，但是不要用到sql的关键字作为开头，比如select、insert这两个
3. 实现类放在对应模块的impl包中
4. 如果你要实现的服务类前后台都适用，比如redis服务类、邮件服务类、短信服务类、文件上传下载服务类等等你能想到的通用的业务，可以写在common模块中，但要记住在common模块的中的service不用加@Service注解，也不会被spring扫描到，要使用这个服务，要在前台或后台的配置类中手动@Bean，实例化那个服务类。

Dao层

1. 该项目所有的Dao接口类以Mapper作为后缀，且都有对应的xml文件，xml文件位于resource的mapper中，是由application.yml中配置的扫描路径决定的，如果没有必要就按这个路径存放，以模块分文件夹。

```
mybatis-plus:
 mapper-locations:
   classpath:/mapper/**/*.xml
```

2. 接口的命名以sql的关键字作为前缀：
   查——selectXxxByXxxAndXxx()
   改——updateXxxByXxxOrXxx()
   增——insertXxx()
   删——deleteXxxByXxx()
3. 查询的sql语句不许写select * ，必须写入字段名，比如select id,name,age这样

DTO

1. sql语句返回的内容如果比较复杂，用实体类包裹不起来，请声明一个DTO类封装那些内容，如果觉得两个类的重复字段很多，可以用 BeanUtils.copyProperties(类1,类2)来帮你封装重复的字段。
2. 不要觉得建DTO类麻烦，之后的开发肯定会修改数据库表，而改完表，DTO是不一定要改的，所以改动的地方会很少，更利于维护。