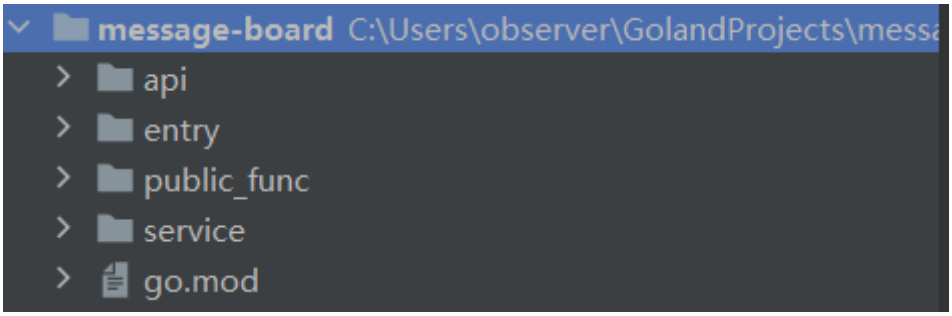
后端第七次作业

说在前面：

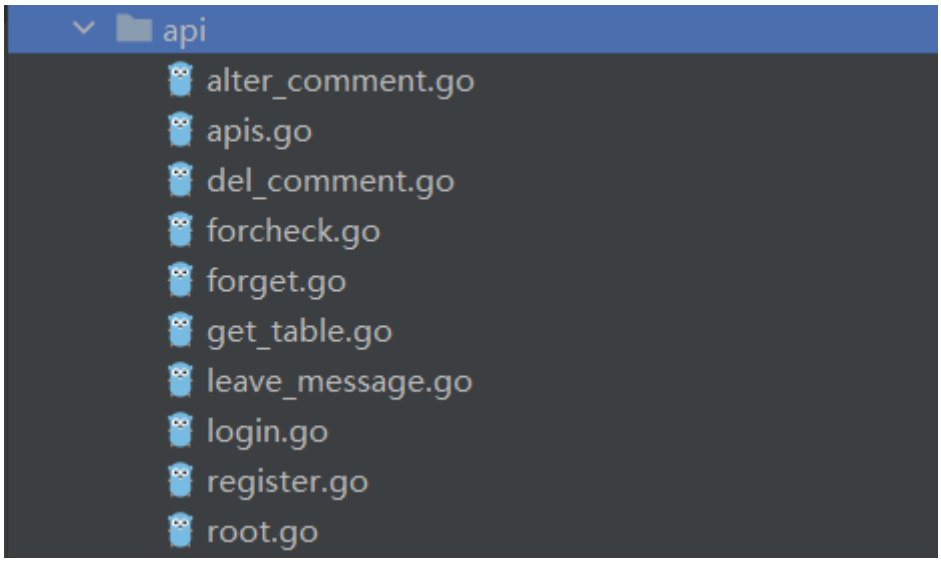
还有很多有意思的小功能，没时间整了，就先鸽了

等有时间学了前端后，考虑把它们一一实现，如果只有后端，效果不明显，也很抽象

项目结构



一.api



api模块是项目的重头，注册了项目所使用的主要接口

包括：

- 注册
- 登录
- 主页
- 密保问题获取
- 密保答案检查
- 留言

- 修改评论
- 删除评论（加标记）

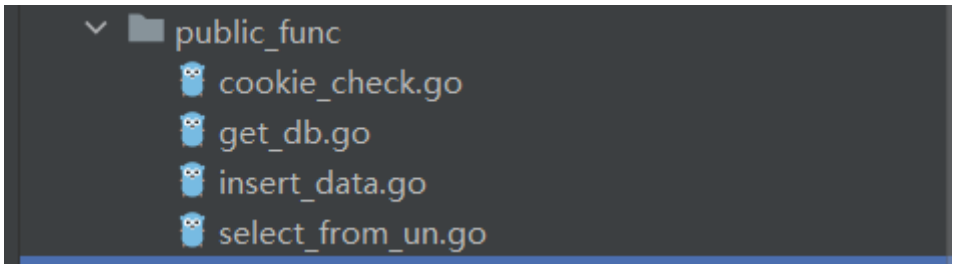
二.entry

项目的启动入口，将包装后的模块集成在这里供启动

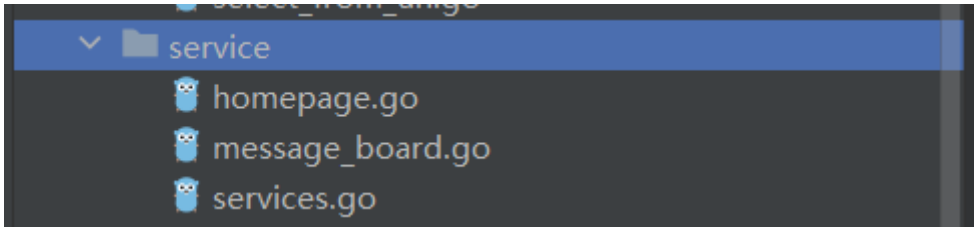
三.public_func

编写了一系列公共方法，声明在此处供其他包调用

- 检查cookie
- 获取数据库
- 插入数据
- 查表



四.service



将api的功能分区集成后存放在这里，算是entry下的另一个分发的入口

数据库的架构

使用两张表完成

一.user表

```
CREATE TABLE `user` (
`id` INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
`username` varCHAR(16) NOT NULL UNIQUE,
`password` varchar(20) NOT NULL,
`question` varchar(20) NOT NULL,
`question_content` varchar(50) NOT NULL
);
```

二.message表

```
CREATE TABLE `message` (
`id` INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
`username` varCHAR(16) NOT NULL,
`parentid` int NOT NULL,
`message`  varchar(1000) NOT NULL default "no message",
`isdelete` int default 0
);
```

三.嵌套实现思路：

在插入表时：

- 如果是根评论，则parentid记为114514，id主键递增
- 如果是下属评论，指定parentid发布

在遍历表，获取内容时

- 为了不给前端留下压力，这里先把数据进行一次处理，而不是直接将整张表返回给前端
- 首先通过查表获取parentid为114514的根评论，作为第一层
- 从这些根评论开始向下查找，parentid为这些根评论的评论。即为根评论的子评论，如此向下递归，每查到一个就在该根评论的结构体切片下append一个，直到不存在parentid为该值的项
- 最后将上一条执行的每一个结构体切片再append成一个切片，即结构体切片切片
- 向前端返回已经处理过的切片