

期末大作业

这是吴逸豪2018302180060、翁斌2018302180061的数据库期末大作业后端代码

功能介绍

整体代码实现了一个最简单线上论坛，具体功能包括：

1. 用户注册登录，用户信息上传，包含姓名、性别、工作、年龄、兴趣
2. 用户权限控制，在power字段储存，包括 0游客、1注册用户、2高级用户、3管理员、4开发者，权限依次扩大
3. 文章操作，包括创建、修改、删除基本操作，包含创建、修改时间记录，**并特色实现隐私发布功能**
4. 评论操作，包括创建、修改、删除、**点赞操作**，包含**自动更新的创建、最近修改时间记录**

具体实现见Mark 0.1部分

安全考量（突出特色）

本次大作业我们组结合专业特点，没有增加太多功能，而是将关注点放在安全方向。

大作业后端代码有两个版本 Mark 0.1为仅功能实现版本，Mark 0.2为考虑安全性后做出的改进版本，主要改进点有：

1. 将网络劫持及前端解包纳入考虑，采用加密传输与储存
2. 考虑SQL注入威胁，增加防注入措施
3. 使用 视图 和 角色 完善权限控制

具体见Mark 0.2部分

注意

本次数据库大作业使用 django 作为网络框架。

django 是十分优秀的适合文章类网站快速开发的十分优秀的网络框架，其本身具有隐藏SQL细节的数据库操作。

但为了展现课程所学知识，我们仅仅使用了django的路径导航部分（Urls部分），django的模型（Models）及视图（Views）部分也仅仅使用了与Urls连接的部分，数据库操作的主题使用的是python嵌入式sql，数据库使用Mysql完成。

Mark 0.1

该版本仅为功能实现版本，接下来的功能实现细节将提前嵌入式sql中sql语句的简化版本，仅去除python中的字符串操作，方便老师与助教查看

wx/test_app/views.py 登录部分

login:

```
select id,password from auther_user where name = username
```

从前端获取username、password，从数据库取出username相同的项的id、password，与获取的password对比，有匹配项说明已注册且密码正确，返回id

signup:

```
INSERT INTO auther_user (`name`, `password`)
VALUES (username,password);
```

从前端获取username、password，插入数据库用户表，支持同名用户，不考虑同名同密码用户

```
select id,password from auther_user where name = username;
```

为实现注册后直接登录，在插入后紧接查询操作

userinfo:

```
select * from auther_user where name = username;
```

从前端获取username，查询所有信息，并以字典json形式发回用户信息

userchange:

```
update auther_user set name=%s,sex=%s,job=%s,age=%s,like=%s where id=%s;
,userinfo['name'], userinfo['sex'], userinfo['job'],userinfo['age'],
userinfo['like'],id
```

通过id确定用户，从前端获取更改后的用户信息，并更新

userpowerup:

```
select power from auther_user where id=userid;
update auther_user set power%s where id=userid;
,str(power+1)
```

通过userid确定用户，查询权限并加一修改，前端需要搭配认证系统，没有实现

wx/article/views.py 文章部分

article_list:

```
select * from article_post order by updated desc;
```

从数据库文章表中取出文章，以更新日期降序排序

article_Mylist:

```
select * from article_post where author_id=userid order by updated desc;
```

从数据库文章表中取出特定用户文章，以更新日期降序排序

article_detail:

```
select * from article_post where id=articleid
```

通过文章id确定文章，选出文章所有信息，加工后发回

article_create:

```
insert into article_post(title,body,created,updated,author_id,privacy)
values(%s,%s,%s,%s,%s,%s);
,value['title'], value['body'], value['updated'], value['created'], value['id'],
value['privacy']
```

通过前端获取信息，插入新的文章记录

article_delete:

```
delete from article_post where id=%s
```

通过前端发来的id确定文章，并删除

article_update:

```
update article_post set title=%s,body=%s,privacy=%s,updated=%s where id=%s
```

通过前端发来的id确定文章，修改为前端提供的内容

wx/answer/views.py 评论部分

commitAnswer:

```
insert into answer_review(reviewText,writerID_id,questionID_id)
values(%s,%s,%s);
,request.POST.get("review"),request.POST.get("writerID"),request.POST.get("articleID")
```

前端传来文章id, 评论者id, 和评论内容, 将其插入

addHelpful:

```
update answer_review set helpfulVote=helpfulVote+1 where id=id
insert into answer_review_users_like(review_id,user_id) values(id, userid)
```

👍, 更新两个表, 文章点赞数 +- 1; 添加某人对某评论的评价记录。此功能前端没有实现

取消👍同样在这个函数中

```
update answer_review set helpfulVote=helpfulVote-1 where id=id
delete from answer_review_users_like where review_id=id and user_id=userid
```

文章点赞数-1, 删除点赞记录

deleteAnswer:

```
delete from answer_review where id=id
```

通过前端发来的id确定评论, 删除评论

getAnswer:

```
select * from answer_review where id=id
```

通过前端发来的id确定评论, 获取评论的全体内容, 选择性发回

getAnswerList:

```
select * from answer_review where questionID_id=id
```

通过前端发来的id确定文章, 获取文章下所有评论

Mark 0.1 安全分析

Mark 0.1 为功能实现版, 所有代码皆为最简单版本, 因此存在各种各样的漏洞。主要有下面三种:

1. 控制信号在客户端明文储存, 在网路中明文传输, 很容易遭到拦截并修改。具体表现为用户id、文章id、评论id皆明文储存并传输, 他们又在服务器中作为确定用户、文章、评论的控制信号。

大体攻击流程为:

1. 下载网页 (web或使用解包软件解包 (小程序、app) 后获取服务器通讯地址
2. 使用BurpSuite、Charles等皆可对本地运行的客户端进行抓包, 并将各类id修改为想要攻击的
3. 正常接收应答并显示, 便可以实现越过登录验证的查看

2. 数据库及其操作未进行防注入操作，攻击者可轻易对数据库进行sql注入攻击

大体攻击流程为：

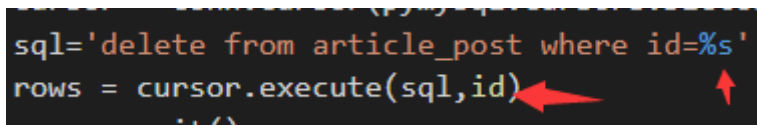
1. 下载网页（web或使用解包软件解包（小程序、app）后获取服务器通讯地址
2. 通过接口设置及后续操作，猜测sql语句及数据库表名、属性名
3. 使用模拟发包软件向返回数据的接口发送构造好的注入语句，如向article_Mylist接口发送id值 '1" or id < 2#' 便可以间接知道2号用户发了那些文章，导致隐私发送失效
4. 使用union联合查找可得到注入更加危险的sql语句，实现查询任意数据，甚至直接操作数据库

3. 服务端明文储存所有信息，对用户隐私无保护，被拖库时隐私泄露严重

Mark 0.2

针对Mark 0.1的安全分析，我们对后端代码进行了优化，主要有：

1. 为了解决控制信号被篡改的问题，我们加入了简单的cookie：
 1. 在user表中加入updated字段储存最后登录时间，登录时自动更新；加入cookie字段储存cookie
 2. 当用户注册或登录时，后端取得updated属性和id属性，连接后进行MD5加密作为cookie，储存进cookie字段并发还前端储存
 3. 前端在其他请求中带上cookie，后端收到cookie与数据库中储存的进行对比，相同才会进行接下来的操作
 4. 为了方便取得id-cookie对并避免所以功能都可以访问user表，创建cookie-view仅显示id-cookie两个属性
2. 为了解决sql注入的问题，我们采取了两种方法：
 1. 关键字匹配：使用正则表达式筛出危险字符，这种方法不适合大段文字，因此我们仅在登录界面使用。规定用户名和密码只能由字母数字和下划线组成，并使用正则表达式匹配，当匹配结果与原字符串完全相同时才继续流程
 2. 预编译sql语句：不使用字符串连接，而使用python中的sql预编译功能，确定某些字符串仅作为条件，可以从根本上避免输入作为独立语句执行，如图：



```
sql='delete from article_post where id=%s'
rows = cursor.execute(sql,id)
```

3. 为了防止出现新型sql跨表查询攻击，我们通过创建不同用户进行单独授权，限定了每个功能的权限。以登录系统为例，共创建5个用户，对应5种功能：

```
CREATE USER 'login' IDENTIFIED BY 'login_password';
CREATE USER 'signup' IDENTIFIED BY 'signup_password';
CREATE USER 'userinfo' IDENTIFIED BY 'userinfo_password';
CREATE USER 'userchange' IDENTIFIED BY 'userchange_password';
CREATE USER 'userpowerup' IDENTIFIED BY 'userpowerup';

GRANT select,update ON auther_user TO 'login';
GRANT insert ON auther_user TO 'signup';
GRANT select ON auther_user TO 'userinfo';
GRANT update ON auther_user TO 'userchange';
GRANT update ON auther_user TO 'userpowerup';
```

同时还可以创建不同的角色和视图对不同功能的权限做更加详细的限定，但我们的项目功能简单，在其他安全性措施下已经没有必要进行这些限定。

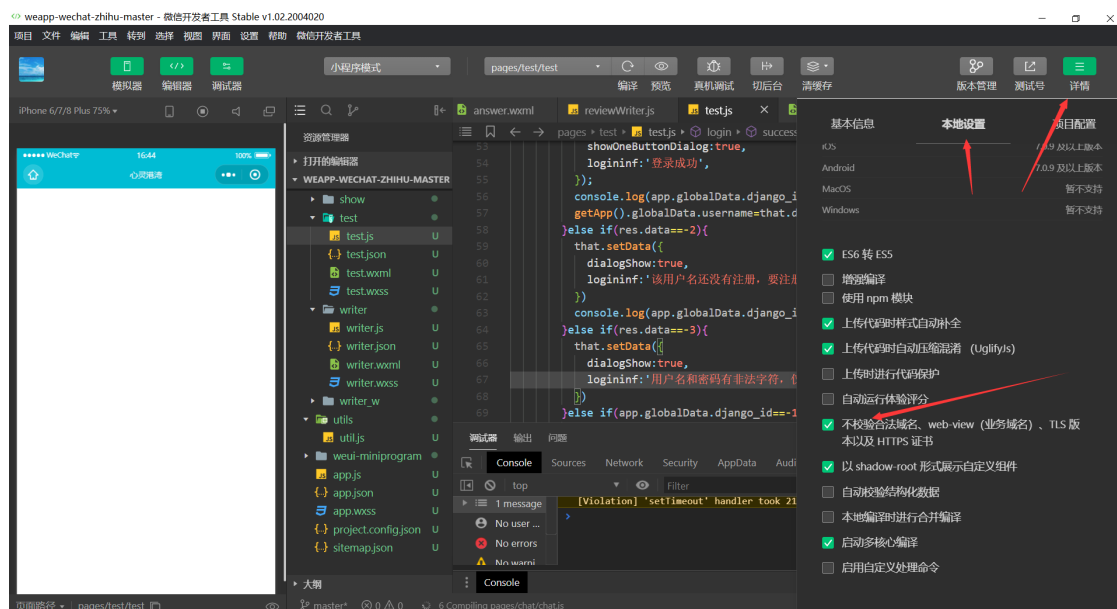
4. 为了解决遭到拖库时密码隐私的泄露问题，我们采用加盐加密的方法：

1. 用户注册时，后端自动生成随机字符串称为salt，salt与密码运算后进行MD5加密，将加密后密文连同salt储存进数据库
 2. 用户登录时，后端从数据库中取得salt和密文，对用户输入的密码进行同样运算得到密文，两个密文进行对比，只有相同时验证通过
 3. 通过这样的方法，可以避免拖库时密码明文泄露，同时也可以防止只进行单纯MD5加密被使用彩虹表破解的情况
5. 对于需要明文查看的其他隐私，如电话等，需要使用密钥进行可逆加密，并将密钥放在单独文件中防止拖库时一并泄露，这样的操作和数据库关系不大，同时不好演示，我们没有实现

具体代码及使用方法

- 前端代码放在front文件夹中，是小程序工程文件，需要使用微信小程序开发工具导入并进行测试。请注意如不能正确连接本地服务器，大概率是要设置忽略HTTPS，如图：

其他问题还请联系团队成员



- 后端代码放在wx文件夹中，是django工程文件，需要 python3.6.x 和 django2.x 环境运行，直接在wx/目录下运行命令：

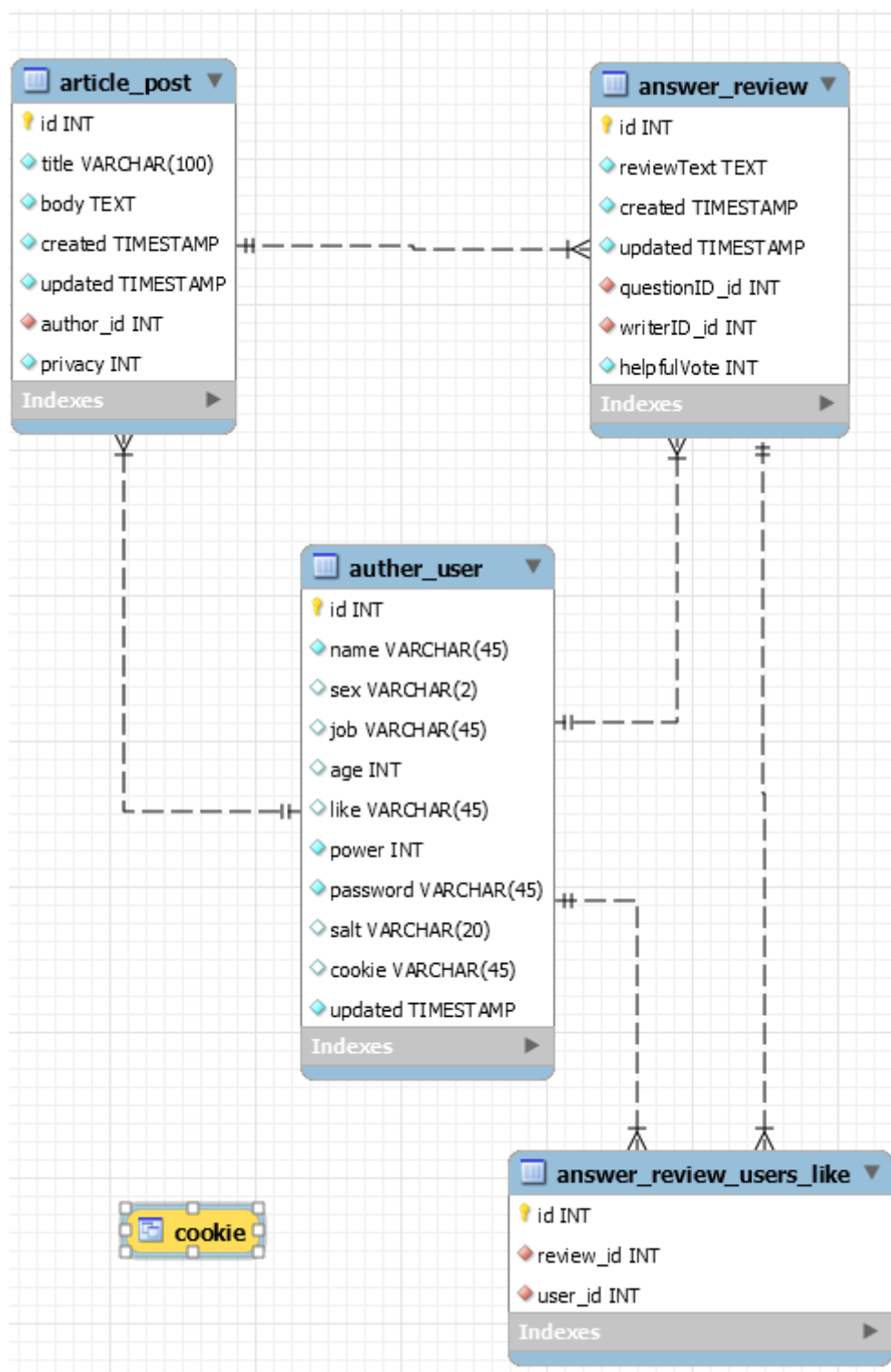
```
python manage.py runserver
```

即可开启服务器。

有其他问题请联系团队成员。

- 数据库代码为DB.sql，是一个sql清单文件，可以直接在Mysql8环境下运行，创建所有需要的数据库结构

数据库ER图如下：



演示视频请见：