数据库原理及安全实验报告模板

姓　　名	丁文兵	学号	2017301500258	班级	信安 3 班
实验名称	Database Web Application			日期	2019/12/18

【实验内容及要求】

搭建一个 Web 服务应用，有如下要求

1. Login, different user Access Control
2. Password Hashed
3. Integrity Check, Trigger
4. Transaction, Lock, Concurrency
5. Anti SQL Injection
6. Some Data encrypted
7. Data Backup & Restore

【实验原理】

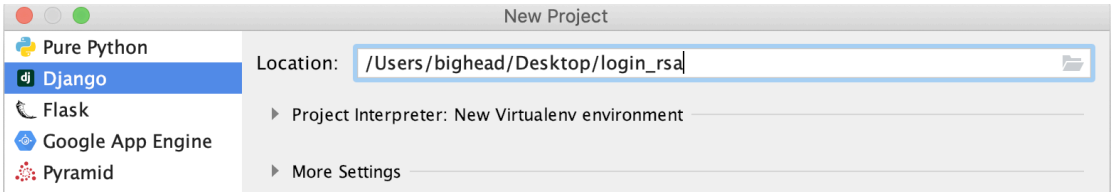
【实验平台】

Mac OSX+Django+mysql+phpmyadmin

【实验步骤】

一．新建工程

1.1 新建 Django 工程，并命名为 login_rsa



1.2 新建 app 并取名 login

```
python3 manage.py startapp login
```

1.3 配置数据库管理

进入 mysql 并新建数据库 rsa

create database rsa;

之后在 django 工程中的 settings 中配置数据库

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        #'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
        'NAME': 'rsa',          #数据库名字
        'USER': 'root',           #账号
        'PASSWORD': '*',        #密码
        'HOST': '127.0.0.1',    #IP
        'PORT': '3306',                   #端口

    }
}
```

1.4 添加新建的 app login

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'login',
]
```

二. 工程开发

2.1 实现登录。

2.1.1 数据库准备:

```
class User(models.Model):
    '''用户表'''

    gender = (
        ('male', '男'),
        ('female', '女'),
    )

    name = models.CharField(max_length=128, unique=True)
    password = models.CharField(max_length=256)
    email = models.EmailField(unique=True)
    sex = models.CharField(max_length=32, choices=gender, default='男')
    c_time = models.DateTimeField(auto_now_add=True)
    types = models.BooleanField(default=False)

    def __str__(self):
        return self.name

    class Meta:
        ordering = ['c_time']
        verbose_name = '用户'
        verbose_name_plural = '用户'
```

新建数据库模型 User，其中

name 表示用户名,并且唯一不重复

password 为用户密码。长度为 256 字节。使用 md5 哈希函数加密存储。

email 为用户邮箱

sex 为性别

c_time 为注册时间

types 为 bool 值，表示用户类型，其中 False 表示学生，True 表示老师。

之后生成迁移文件 python3 manage.py makemigrations

再进行迁移 python3 manage.py migrate

此时可查看数据库生成了 rsa_user 表

#	名字	类型	排序规则	属性	空	默认	注释	额外	操作
1	id	int(11)			否	无		AUTO_INCREMENT	修改 删除 更多
2	name	varchar(128)	utf8mb4_0900_ai_ci		否	无			修改 删除 更多
3	password	varchar(256)	utf8mb4_0900_ai_ci		否	无			修改 删除 更多
4	email	varchar(254)	utf8mb4_0900_ai_ci		否	无			修改 删除 更多
5	sex	varchar(32)	utf8mb4_0900_ai_ci		否	无			修改 删除 更多
6	c_time	datetime(6)			否	无			修改 删除 更多
7	types	tinyint(1)			否	无			修改 删除 更多

其中 id 为 Django 自动生成，默认的主键

2.1.2 前端页面设计：

由于本实验为数据库实验，注重对数据库的设计，本实验中前端界面仅实现功能，不再做美化设计

首先制作 base.html 为本实验中各网页的基础网页，后续的网页都是对其的继承拓展

该界面是用户的基本控制包括登录登出跳转，logo，以及用户在线时显示当前用户

之后进行登录界面的 UI 设计。

使用 django 中的 form 设计，可以用作前端设计，同时在前后端交互式交换数据也有很大的便利

首先是登录界面。表单需要用户输入用户名，二次确认密码，邮箱，性别选择，角色选择

```
class RegisterForm(forms.Form):
    gender = (
        ('male', "男"),
        ('female', "女"),
    )
    types=(
        ('student',"学生"),
        ('teacher','老师'),
    )
    username = forms.CharField(label="用户名", max_length=128, widget=forms.TextInput(attrs={'class': 'form-control'}))
    password1 = forms.CharField(label="密码", max_length=256, widget=forms.PasswordInput(attrs={'class': 'form-control'}))
    password2 = forms.CharField(label="确认密码", max_length=256, widget=forms.PasswordInput(attrs={'class': 'form-control'}))
    email = forms.EmailField(label="邮箱地址", widget=forms.EmailInput(attrs={'class': 'form-control'}))
    sex = forms.ChoiceField(label='性别', choices=gender)
    role=forms.ChoiceField(label='身份', choices=types)
```

做出后前端效果如下：

之后是登录界面设计，需要用户输入用户名和密码即可

```python
from django import forms


class UserForm(forms.Form):
    username = forms.CharField(label="用户名", max_length=128, widget=forms.TextInput(attrs={'class': 'form-control'}))
    password = forms.CharField(label="密码", max_length=2000, widget=forms.PasswordInput(attrs={'class': 'form-control','name':'password'
```

前端效果如下：

同时为了通信安全，前端采用了 md5 哈希以及 RSA 公钥加密传输。

```
btn.onclick = function(){
    var modulus = {{ moddd | safe }};//获得n的base64编码
    var exponent = {{ exppp | safe }};//获得e的base64编码
    var mod=Base64.decode(modulus[0]);//进行base64解码获得n
    var exp=Base64.decode(exponent[0]);//进行base64解码获得e
    var passwd=document.getElementById('id_password');
    var password=hex_md5(passwd.value);//对口令进行md5加密获得十六进制字符串
    var n=bigInt(mod);//n转化为大整数
    var e=bigInt(exp);//e转化为大整数
    var m=bigInt(password,16);//口令哈希值转化为大整数
    var res=Base64.encode(RSA.encrypt(m,e,n).toString());//使用RSA加密并进行Base64编码

    var form1=document.forms[0];
    passwd.value=res;
    form1.submit();
```

如图，前端获取后端发送的公钥（n,e）。将用户输入的密码先使用 md5 哈希，在使用（n，e）公钥对加密

2.1.3 后段设计以及前后端交互

后端则要实现公钥生成，以及登录时的 RSA 解密，并将密码的哈希值跟数据库存储的哈希值进行比对校验。

```
login_form = UserForm()
n, e, d = get_rsa_pair(256)  # 生成公私钥对（n,e）（n,d）。p，q为512位，n为1024位
request.session['n'] = n
request.session['d'] = d
mod = base64.b64encode(str(n).encode('utf-8')).decode()  # base64编码n
exp = base64.b64encode(str(e).encode('utf-8')).decode()  # base64编码e
return render(request, 'page/login.html', {'message':message,"login_form": login_form, "moddd": [mod], "exppp": [exp]})
```

当请求为 get，即用户请求登陆时，后端产生公钥对返回，同时存储私钥方便后续解密。

```
if login_form.is_valid():
    username = login_form.cleaned_data['username']#获得用户名
    password = login_form.cleaned_data['password']#获得密文
    #使用base64解码并使用（n，d）对解码后的密文进行解密
    password=modd(int(base64.b64decode(password).decode()),int(request.session['d']),int(request.session['n']))
    #获得解密后口令哈希值的十六进制字符串
    password=hex(password).replace('0x','')
```

该步骤实现了对传输密文进行解密，获取密码哈希值

```
try:
    user = models.User.objects.get(name=username)
    if request.session.get('is_login', None):
        r = redirect('/index/')
        r.set_cookie('username', username, max_age=1000)
        return r
```

将解密后的密码同数据库中的密码进行比较。

	id	name	password	email	sex	c_time	types
编辑 复制 删除	2	shybeejd	e10adc3949ba59abbe56e057f20f883e	1@qq.com	male	2019-12-10 13:44:48.521202	0
编辑 复制 删除	12	Theshy	81dc9bdb52d04dc20036dbd8313ed055	12333@qq.com	male	2019-12-17 12:43:14.814000	0
编辑 复制 删除	13	yyy	81dc9bdb52d04dc20036dbd8313ed055	1231414@qq.com	male	2019-12-17 12:44:50.269419	0
编辑 复制 删除	14	yyyy	81dc9bdb52d04dc20036dbd8313ed055	12314141@qq.com	male	2019-12-17 12:46:05.379965	0
编辑 复制 删除	15	1234	202cb962ac59075b964b07152d234b70	111@qq.com	male	2019-12-17 12:49:03.064150	1
编辑 复制 删除	16	teacher	202cb962ac59075b964b07152d234b70	11111@qq.com	male	2019-12-17 13:08:51.018362	1
编辑 复制 删除	17	shy	202cb962ac59075b964b07152d234b70	qqq@qq.com	male	2019-12-20 06:06:08.181741	0

用户表数据样例

2.2 用户主界面

2.2.1 教师用户

教师的课程拥有一个数据库，存储了课程的相关信息。

```
class Course(models.Model):
    c_name=models.CharField(max_length=128)
    c_teacher=models.CharField(max_length=128)
    max_count=models.IntegerField(default=40)
    left_count=models.IntegerField(default=0)

    def __str__(self):
        return self.c_name

    class Meta:
        unique_together=(("c_name","c_teacher"),)
        ordering = ['c_name']
        verbose_name = '课程'
        verbose_name_plural = '课程'
```

c_name:课程名
c_teacher：老师名
max_count：最大人数
left_count：剩余人数
id:课程 id
其中主键为 id

	id	c_name	c_teacher	left_count	max_count
编辑 复制 删除	18	数据库	teacher	1	2
编辑 复制 删除	19	121	teacher	1	2
编辑 复制 删除	22	OS	teacher	1	2
编辑 复制 删除	23	nah	teacher	2	3
编辑 复制 删除	24	111	teacher	1	2

此外还有个学生选课的表

```
class Choice(models.Model):
    course=models.ForeignKey(Course)
    time = models.DateTimeField(auto_now_add=True)
    score = models.IntegerField(default=0)
    stu=models.CharField(max_length=128,default=' ')
    class Meta:
        ordering = ['time']
        verbose_name = '已选课'
        verbose_name_plural = '已选课'
```

course:外键对应课程 id
time:选课时间
stu；选课的学生名
score：学生成绩

	id	time	score	course_id	stu
编辑 复制 删除	7	2019-12-18 07:27:42.098707	100	18	shybeejd
编辑 复制 删除	11	2019-12-18 11:45:25.976761	22	22	shybeejd
编辑 复制 删除	12	2019-12-19 02:35:26.956340	0	23	shybeejd
编辑 复制 删除	17	2019-12-20 06:13:16.540743	0	19	shybeejd
编辑 复制 删除	18	2019-12-20 06:14:16.318225	0	24	shybeejd

用户身份为教师则主页面会显示该老师所拥有的所有课程

My course	Left_num	Max_num	Detial	Manage
111	1	2	详细信息	删除课程
121	1	2	详细信息	删除课程
nah	2	3	详细信息	删除课程
OS	1	2	详细信息	删除课程
数据库	1	2	详细信息	删除课程
			创建课程	

详细信息中会显示该课程的所有学生，并且可对学生的成绩进行管理修改。也可以帮学生撤课，即删除学生

当前课程:111

Stu	Score	Manage	Delete
shybeejd	0	修改	删除

修改后的成绩如下：

Stu	Score	Manage	Delete
shybeejd	100	修改	删除

后端实现：

```
with transaction.atomic():
    new_course = models.Course.objects.create()
    new_course.c_name = new_c
    new_course.c_teacher =username
    new_course.left_count=max_count
    new_course.max_count=max_count
    if new_course.c_name and new_course.c_teacher:
        new_course.save()
```

当教师新建课程时存储课程。

```
c = models.Course.objects.get(c_teacher=username, c_name=course)
exi = models.Choice.objects.filter(course_id=c.id)
if exi:
    message = '存在学生无法删除'
else:
    c = models.Course.objects.get(c_name=course, c_teacher=username)
    c.delete()
```

当老师要删除课程时，首先检验课程是否有学生选课，如果有则删除失败。

2.2.2 学生用户

数据库设计：

与学生相关的表有 choice 即之前介绍的选课信息存储表，此外还有 inf 表，存储了学生的历史选课记录（包括老师撤课）

```
class Inf(models.Model):
    name=models.CharField(max_length=128)
    course=models.ForeignKey(Course)
    time = models.DateTimeField(auto_now_add=True)
    def __str__(self):
        return self.name

    class Meta:
        ordering = ['time']
        verbose_name = '选课记录'
        verbose_name_plural = '选课记录'
```

name:学生名
course:课程信息
time：选课时间

前端展示：

课程记录

Course	Teacher	Left	Max	option	Score
111	teacher	1	2	已选	0
121	teacher	1	2	已选	0
nah	teacher	2	3	已选	0
OS	teacher	1	2	已选	22
数据库	teacher	1	2	已选	100

第一个是学生可选课以及已选课程的信息，如果课程没有选，则可以进行选

课，已经选课则可以查看课程分数。

还有个选课历史记录，可以查看自己选课历史

历史记录

Course	Teacher	time
数据库	teacher	Dec. 18, 2019, 7:27 a.m.
121	teacher	Dec. 18, 2019, 7:29 a.m.
OS	teacher	Dec. 18, 2019, 11:37 a.m.
OS	teacher	Dec. 18, 2019, 11:44 a.m.
nah	teacher	Dec. 19, 2019, 2:35 a.m.
121	teacher	Dec. 19, 2019, 2:36 a.m.
121	teacher	Dec. 19, 2019, 2:38 a.m.
121	teacher	Dec. 20, 2019, 6:09 a.m.
121	teacher	Dec. 20, 2019, 6:13 a.m.
111	teacher	Dec. 20, 2019, 6:14 a.m.

后端实现：

```
with transaction.atomic():
    c = models.Course.objects.select_for_update().get(c_name=course, c_teacher=teacher)  # 加锁
    c.left_count=c.left_count-1
    choice=models.Choice(course_id=c.id,stu=username)
    choice.save()
    c.save()
```

学生的后端主要为学生选课，在确定剩余人数大于 0 时进行选课并将剩余人数减 1.

三，需求实现

3.1 Login, different user Access Control

本实验实现了简单的教务管理，适用对象为老师和学生。注册时自己选择身份，登录后会自动判断用户的身份并返回不同的界面。对于老师可以发布课程，修改学生分数，帮学生撤课，删除课程。

对于学生用户可以查看课程信息并进行选课，查看自己的已选课程的分数，并查看自己的历史选课记录。

3.2 Password Hashed

用户和学生的密码存储在 user 表中，适用 md5 加密

←T→	id	name	password	email	sex	c_time	types
编辑 复制 删除	2	shybeejd	e10adc3949ba59abbe56e057f20f883e	1@qq.com	male	2019-12-10 13:44:48.521202	0
编辑 复制 删除	12	Theshy	81dc9bdb52d04dc20036dbd8313ed055	12333@qq.com	male	2019-12-17 12:43:14.814000	0
编辑 复制 删除	13	yyy	81dc9bdb52d04dc20036dbd8313ed055	1231414@qq.com	male	2019-12-17 12:44:50.269419	0
编辑 复制 删除	14	yyyy	81dc9bdb52d04dc20036dbd8313ed055	12314141@qq.com	male	2019-12-17 12:46:05.379965	0
编辑 复制 删除	15	1234	202cb962ac59075b964b07152d234b70	111@qq.com	male	2019-12-17 12:49:03.064150	1
编辑 复制 删除	16	teacher	202cb962ac59075b964b07152d234b70	11111@qq.com	male	2019-12-17 13:08:51.018362	1
编辑 复制 删除	17	shy	202cb962ac59075b964b07152d234b70	qqq@qq.com	male	2019-12-20 06:06:08.181741	0

用户在前端登录时，就已经将密码进行 md5 加密，后端使用私钥解密 rsa 密码后得到的也是哈希后的口令，再同数据库存储的口令哈希值比较，以此判断用户的合法性。

3.3 Integrity Check, Trigger

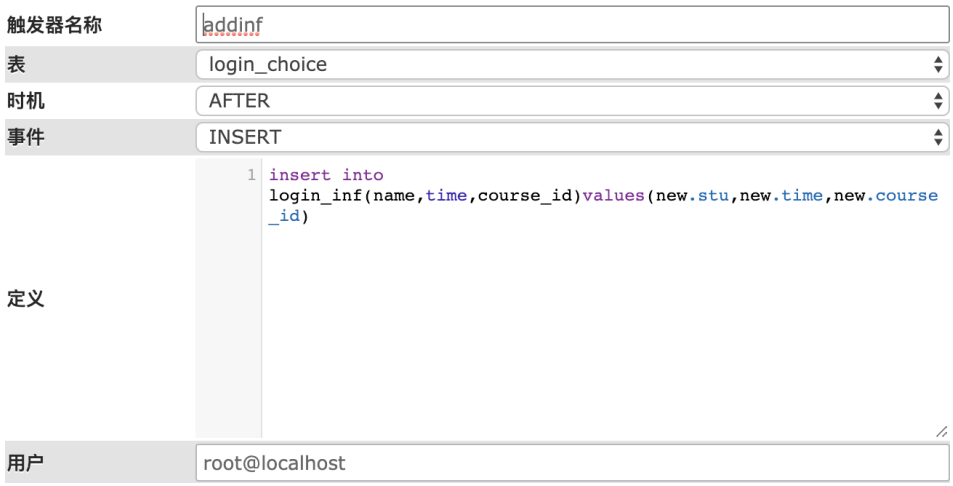
课程 id 为课程的主键，同学的选课依赖于课程，当老师删除课程时，如果仍然有同学选课就会提示无法删除课程，即此时要进行完整性校验。

```
c = models.Course.objects.get(c_teacher=username, c_name=course)
exi = models.Choice.objects.filter(course_id=c.id)
if exi:
    message = '存在学生无法删除'
else:
    c = models.Course.objects.get(c_name=course, c_teacher=username)
    c.delete()
```

同时数据库表 choice 的外键也是课程 id，在有学生选课的情况下无法直接删除课程

#	名字	类型	排序规则	属性	空	默认	注释	额外	操作
1	id	int(11)			否	无		AUTO_INCREMENT	修改 删除 更多
2	time	datetime(6)			否	无			修改 删除 更多
3	score	int(11)			否	无			修改 删除 更多
4	course_id	int(11)			否	无			修改 删除 更多
5	stu	varchar(128)	utf8mb4_0900_ai_ci		否	无			修改 删除 更多

此外为了保存选课的历史记录，数据库中添加了一个触发器，当学生进行了一次选课操作，则自动向表 inf 中添加选课记录。



如果老师私自撤学生的课，学生则可以以此为记录进行维权。

3.4 Transaction, Lock, Concurrency

本实验中事务，锁共同实现。Django 已经支持并发，不再赘述，但是为了处理并发时保证数据库的正确，需要处理并发事物。

其中最为重要的是学生并发选课。当选课开放时，会有大量学生选课。

学生选课进行的主要操作有：1.课程的剩余人数减 1。2.添加学生选课记录。二者要同时完成或者同时失败，此处即将二者作为一个事务进行处理。同时对于课程的剩余人数需要进行加锁处理，防止在多数学生选课时读取错误的数据信息，导致剩余人数写入错误

```
with transaction.atomic():
    c = models.Course.objects.select_for_update().get(c_name=course, c_teacher=teacher)  # 加锁
    c.left_count=c.left_count-1
    choice=models.Choice(course_id=c.id,stu=username)
    choice.save()
    c.save()
```

3.5Anti SQL Injection

Django 自带反 SQL 注入机制。其中 ORM 查询数据库已经避免了 SQL 注入的风险。

3.6Some Data encrypted

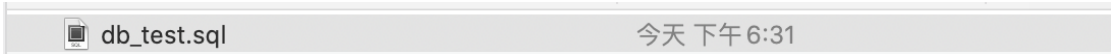
本实验中最重要的数据用户密码已经哈希储存，考虑到信道的安全，本实验中的加密也是对通信信道的加密。如前所述。用户在输入密码后先使用 md5 哈希再使用后端传输的公钥进行 RSA 加密。请求发到后端后使用私钥解密并同数据库进行比对校验

3.7 Data Backup & Restore

导出整个数据库

mysqldump -u 用户名 -p 数据库名 > 导出的文件名

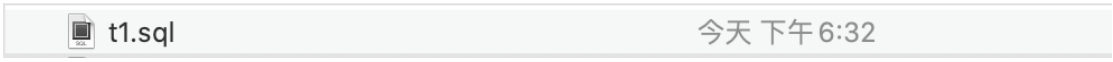
```
bighead@dingwenbingdeMacBook-Pro bin % /usr/local/mysql-8.0.15-macos10.14-x86_64
/bin/mysqldump -u root -p db_test >db_test.sql
```



导出一个表

mysqldump -u 用户名 -p 数据库名 表名> 导出的文件名

```
bighead@dingwenbingdeMacBook-Pro bin % /usr/local/mysql-8.0.15-macos10.14-x86_64
/bin/mysqldump -u root -p db_test t1 >t1.sql
```



导入数据库

方法一：

（1）选择数据库

mysql>use abc;

（2）设置数据库编码

mysql>set names utf8;

（3）导入数据（注意 sql 文件的路径）

mysql>source /home/abc/abc.sql;

方法二：

mysql -u 用户名 -p 密码 数据库名 < 数据库名.sql

#mysql -uabc_f -p abc < abc.sql

【实验结果】

教师登陆：

创建课程

My course	Left_num	Max_num	Detial	Manage
111	1	2	详细信息	删除课程
121	1	2	详细信息	删除课程
nah	2	3	详细信息	删除课程
OS	1	2	详细信息	删除课程
数据库	1	2	详细信息	删除课程
密码学		5	创建课程	

创建后如下

My course	Left_num	Max_num	Detial	Manage
111	1	2	详细信息	删除课程
121	1	2	详细信息	删除课程
nah	2	3	详细信息	删除课程
OS	1	2	详细信息	删除课程
密码学	5	5	详细信息	删除课程
数据库	1	2	详细信息	删除课程
			创建课程	

查看学生选课信息：

Stu	Score	Manage	Delete
shybeejd	0	修改	删除

修改分数：

Stu	Score	Manage	Delete
shybeejd	100	修改	删除

删除学生：

当前课程:111

Stu	Score	Manage	Delete

学生登录：

课程记录

Course	Teacher	Left	Max	option	Score
111	teacher	2	2	选课	
121	teacher	1	2	已选	0
nah	teacher	2	3	已选	0
OS	teacher	1	2	已选	22
密码学	teacher	5	5	选课	
数据库	teacher	1	2	已选	100

选课：

Course	Teacher	Left	Max	option	Score
111	teacher	1	2	已选	0
121	teacher	1	2	已选	0
nah	teacher	2	3	已选	0
OS	teacher	1	2	已选	22
密码学	teacher	5	5	选课	
数据库	teacher	1	2	已选	100

查看选课历史记录：

历史记录

Course	Teacher	time
数据库	teacher	Dec. 18, 2019, 7:27 a.m.
121	teacher	Dec. 18, 2019, 7:29 a.m.
OS	teacher	Dec. 18, 2019, 11:37 a.m.
OS	teacher	Dec. 18, 2019, 11:44 a.m.
nah	teacher	Dec. 19, 2019, 2:35 a.m.
121	teacher	Dec. 19, 2019, 2:36 a.m.
121	teacher	Dec. 19, 2019, 2:38 a.m.
121	teacher	Dec. 20, 2019, 6:09 a.m.
121	teacher	Dec. 20, 2019, 6:13 a.m.
111	teacher	Dec. 20, 2019, 6:14 a.m.
111	teacher	Dec. 27, 2019, 5:19 a.m.

【实验小结】

本次 web 开发实战对数据库进行了一次应用。将数据库的理论知识运用到了实际应用。同时数据库常遇到的问题也在实际应用中体现了出来。对存在问题的处理进一步加深了对数据库的理解。为未来的数据库开发积累了宝贵的知识经验财富。•

评语：	**成绩：**
	签名：
	日期：　　年　月　日