Python实验报告

实验九：初探tornado

学号：2016326603029 姓名：黄欣悦

学号：2016326603039 姓名：郭 仲

学号：2016326603046 姓名：余泓锷

学号：2016326603071 姓名：王明凯

学号：2016329600130 姓名：王宇舟

实验目的

了解基于python的web应用的相关内容和关键字词含义
了解并掌握tornado包的基本操作及其内部类和方法调用
掌握通过pymysql包结合ORM框架进行数据库的增加和查询

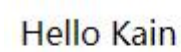
实验内容

在本课目录下,有一个tornado_project.zip,这是另一个工程的例子,里面带有mysql数据库读写的功能,
请仔细分析这个例子,并对这个例子进行修改,增加你觉得需要的功能。

实验过程

对于条目KainHandler的改写

改写前：



改写后：

代码改变部分：

```
class KainHandler(RequestHandler):
    # 该方法会在http方法之前调用
    def initialize(self, item1, item2):
        self.item1 = item1
        self.item2 = item2

    def get(self, *args, **kwargs):
        # print(self.item1, self.itme2)
        str = '<center><h2>Hello '+self.item1
        str += '</h2></center><center><h3>your name is '
        str += self.item1+' '+self.item2+'</h3></center>'
        self.write(str)
```

代码分析：代码从原先的固定输出，变成获取从路由获取服务器传递的参数进行动态输出

对于条目HuckHandler的改写

改写前：

Hello huck

改写后：

Hello huck

your name is kain huck

代码改变部分：

```
class HuckHandler(RequestHandler):
    def initialize(self, item3, item4):
        self.item3 = item3
        self.item4 = item4

    def get(self, *args, **kwargs):
        str = '<center><h2>Hello '+self.item4
        str += '</h2></center><center><h3>your name is '
        str += self.item3+' '+self.item4+'</h3></center>'
        self.write(str)
```

代码分析：展示的效果依然是将原先的固定输出，变成获取动态输出。
不过增加了url反向解析技术

对于条目KainHuckHandler的改写

改写前：

hello kainhuck

改写后：

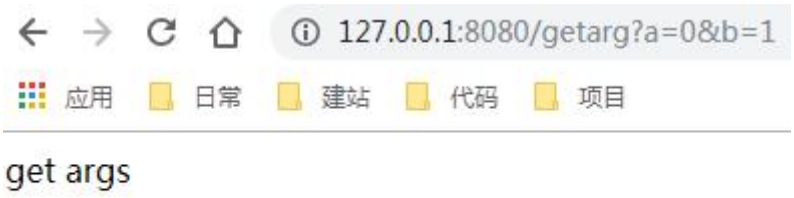
代码改变部分：

```
# 匹配URL中的特定部分
class KainHuckHandler(RequestHandler):
    def get(self, h1, h2, h3, *args, **kwargs):
        print(h1,h2,h3)
        self.write('<center><h2>'+h1+' '+h2+h3+'</h2></cneter>')
```

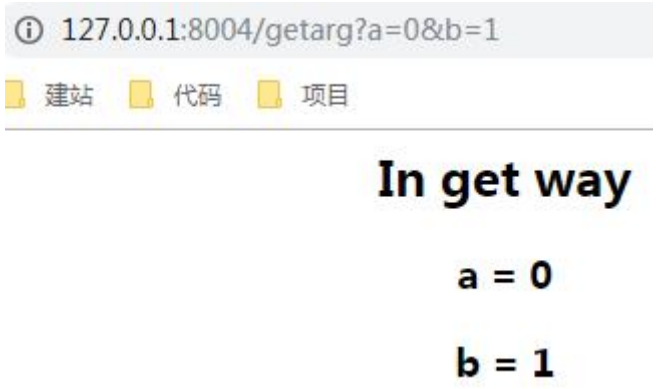
代码分析：代码从原先的固定输出，变成匹配URL特殊位置进行动态输出

对于条目HuckHandler的改写

改写前：



改写后：



代码改变部分：

```
# 获取get方法传递的参数
class GetArgHandler(RequestHandler):
    def get(self, *args, **kwargs):
        # strip过滤左右空格
        a = self.get_query_argument('a',default='not_a',strip=True)
        b = self.get_argument('b',default='not_b',strip=True)
        self.write('<center><h2>In get way</h2><h3>a = '+a+'</h3><h3>b = '+b+'</h3></center>')
        print(a, b)
```

代码分析：代码从原先的固定输出，变成获取get方式传参的参数进行动态输出

本次实验主要实现内容

基于条目PostHandler改写的注册界面

基于条目LoginHandler改写的登录界面

基于条目pymysql包和ORM框架实现的数据库增加和查询操作

注册界面：

登录界面：

用户列表界面：

贪玩蓝月用户列表页面

贪玩蓝月用户列表

userid	username	password	createdAt
1	yuyue_1	c71612924cab34c059756f9d2d3c2d3e	1544610456
2	yuyue_2	6167a09479fac3cd47c9a1abb5ad1c9a	1544610456
3	yuyue_3	634e74e8c856933eb0a5c3344825d6a7	1544610456

数据库结构：

#	名字	类型	排序规则	属性	空	默认	注释	额外	操作
1	userid	int(11)			否	无	用户ID,主键，自增	AUTO_INCREMENT	修改 删除 更多
2	username	varchar(40)	utf8_general_ci		否	无	用户名，字符串		修改 删除 更多
3	password	char(32)	utf8_general_ci		否	无	用MD5加密后密码		修改 删除 更多
4	createdAt	int(11)			否	无	注册时间		修改 删除 更多

【注册验证实现】

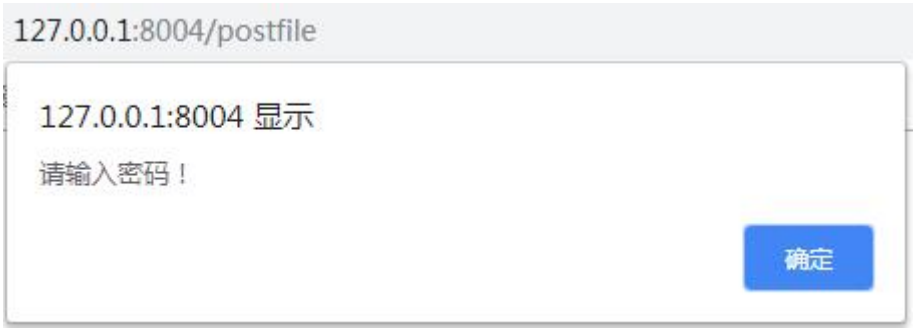
用户名为空：

关键代码：

```
class PostFileHandler(RequestHandler):
    def get(self, *args, **kwargs):
        self.render('register.html')

    def post(self, *args, **kwargs):
        print(self.request.arguments)
        username = self.get_argument('username')
        password = self.get_argument('password')
        re_password = self.get_argument('re_password')
        print(username, password, re_password)
        if username == '':
            self.write('<script>\
                        window.alert("请输入用户名！ ");\
                        </script>')
            self.render('register.html')
            return
```

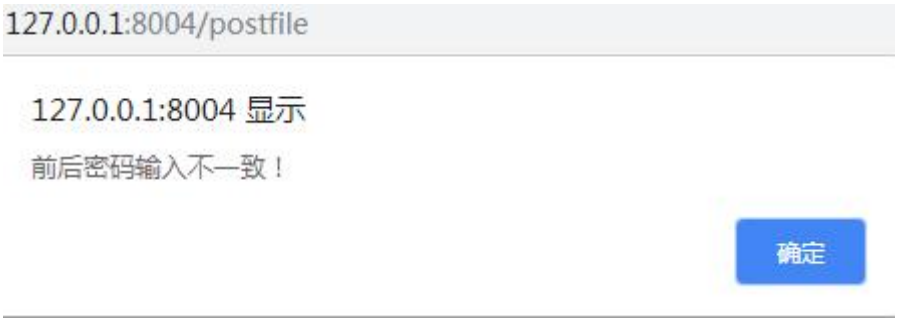
密码为空：



关键代码：

```
if password == '':
    self.write('<script>\
                window.alert("请输入密码！ ");\
                </script>')
    self.render('register.html')
    return
```

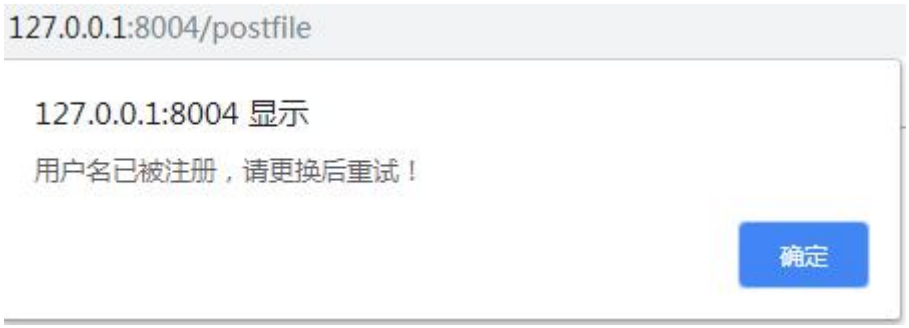
两次密码输入不一致：



关键代码：

```
if password != re_password:
    self.write('<script>\
                window.alert("前后密码输入不一致！ ");\
                </script>')
    self.render('register.html')
```

用户名已存在：



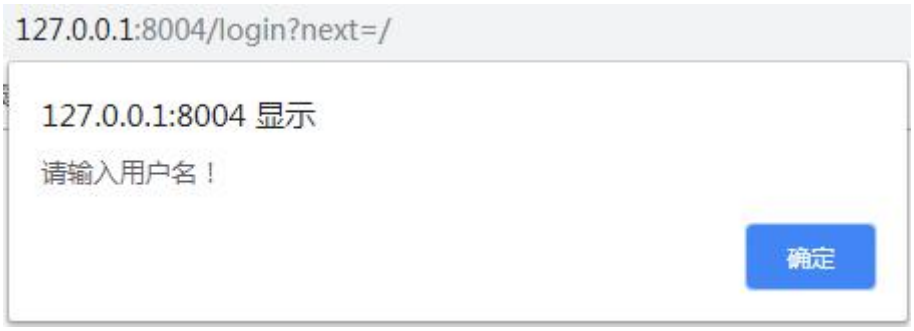
关键代码：

```
else:
    user = Users(username, password)
    jilu_one = user.select_one()
    if (jilu_one==None):
        user = Users(username, password)
        user.save()

        self.write('<script>\
                    window.alert("注册成功！ ");\
                    </script>')
        next = self.get_argument('next', '/')
        url = 'login?next=' + next
        self.render('login.html',url=url)
    else:
        self.write('<script>\
                    window.alert("用户名已被注册，请更换后重试！ ");\
                    </script>')
        self.render('register.html')
```

【登录验证实现】

用户名为空：

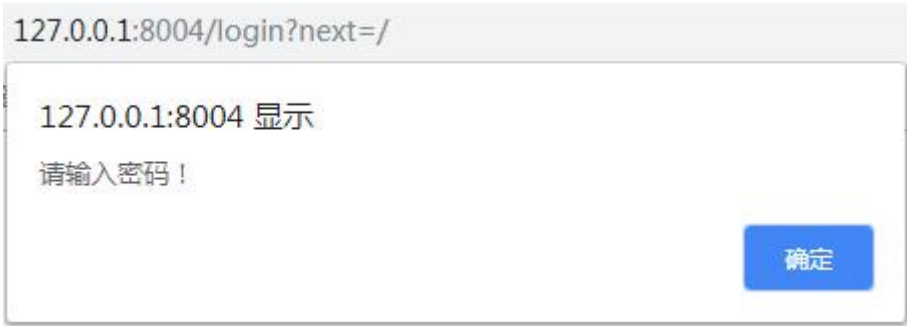


关键代码：

```
class LoginHandler(RequestHandler):
    def get(self, *args, **kwargs):
        next = self.get_argument('next', '/')
        url = 'login?next=' + next
        self.render('login.html', url=url)

    def post(self, *args, **kwargs):
        name = self.get_argument('username')
        passwd = self.get_argument('password')
        next = self.get_argument('next', '/')
        url = 'login?next=' + next
        if name == '':
            self.write('<script>\
                        window.alert("请输入用户名！ ");\
                        </script>')
            self.render('login.html', url=url)
            return
```

密码为空：



关键代码：

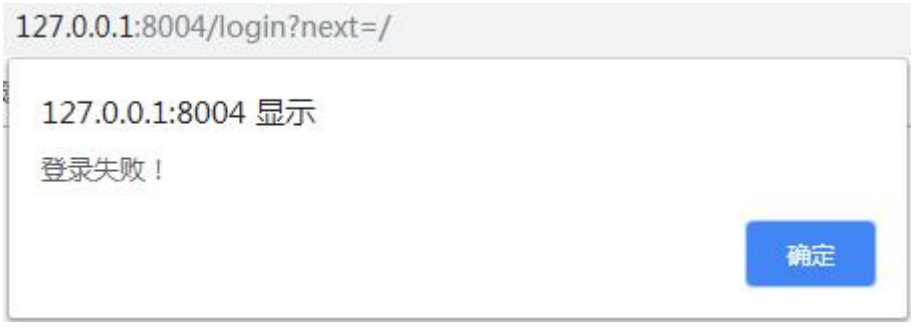
```
if passwd == '':
    self.write('<script>\
                window.alert("请输入密码！ ");\
                </script>')
    self.render('login.html', url=url)
    return
```

用户名不存在：

关键代码：

```
user = Users(name, passwd)
jilu_one = user.select_one()
if (jilu_one==None):
    self.write('<script>\
                window.alert("该用户名不存在，请注册！ ");\
                </script>')
    self.render('login.html', url=url)
    return
```

密码不正确：



关键代码：

```
if common.md5(passwd)==jilu_one[2]:
    self.write('<script>\
                window.alert("登录成功！");\
                </script>')
    next = self.get_argument('next', '/')
    users = Users.all()
    self.render('user.html', users=users)
else:
    self.write('<script>\
                window.alert("登录失败！");\
                </script>')
    self.render('login.html', url=url)
```

【其他实现】

MD5加密：

```
import hashlib

def md5(str):
    m = hashlib.md5()
    m.update(str.encode("utf8"))
    print(m.hexdigest())
    return m.hexdigest()
```

通过用户名查询用户信息：

```
def select_one(self):
    # "select * from users where username=''"
    tableName = self.__class__.__name__.lower()
    fieldStr = ''
    fieldStr += "username=" + "'" + self.__dict__['username'] + "'"
    sql = "select * from " + tableName + ' where ' + fieldStr
    print(sql)
    tempMysql = KainMysql()
    return tempMysql.get_one(sql)
```

代码分析： 通过pymysql包连接本地数据库，需要设置host、db、user和passwd，
而且需要建立相关的数据库以及和模型类名对应的数据表。