# Flask半知半解

**1.**

flask debug模式， 没有引进打包后的文件，需要配置 静态文件所处的文件夹，和模板文件所处的文件夹

app = Flask(\_\_name\_\_,

static\_url\_path='/python', # 静态文件的前缀

static\_folder='static', # 静态文件所处于的文件夹

template\_folder='templates' # 模块文件所处于的文件夹

)

**2. html中引入静态文件**

@app.route('/static/<filename>')

def send\_static(filename):

return send\_from\_directory('build/static/css/', filename)

**3.**

UnicodeDecodeError: 'ascii' codec can't decode byte 0xe7 inposition 0: ordinal not in range(128)

阿斯克码错误

import sys

reload(sys)

sys.setdefaultencoding("utf-8")

**4.Flask获取页面数据的两种方式**

name = request.args.get('name') // GET

name = request.form.get('name') // POST

获取文件： request.files['file']

request.form['abc'] 是获取表单中的数据。

建议使用request.form.get('abc','default value').

理由：

如果表单中没有'abc', form.get('abc','default value') 会返回’default value'.

如果用 form.get('abc'）则返回None;

如果用 form['abc'], 则抛出 400 异常，整个http request 都不可用。

**5.增删改查**

1.增加数据（就相当于增加一个实例对象）

user1 = User(name='long',email='wertyui@qq.com',password='3456789',role\_id=1)

db.session.add(user1)

db.session.commit()

2.修改数据

修改用户表里面的name为long的姓名为：fang

首先查询到名为long的这个用户

user1 = User.query.filter\_by(name='long').first()

赋值／修改

user1.name = 'fang'

提交

db.session.commit()

3.先查询删除

user1 = User.query.filter\_by(name='fang').first()

db.session.delete(user1)

db.session.commit()

4.

Flask中的增删改查

- 查询所有用户数据

User.query.all()

- 查询有多少个用户

User.query.count()

- 查询第1个用户

User.query.first()

- 查询id为4的用户[3种方式]

(1)User.query.get(4)

(2)User.query.filter\_by(id=4).first()

(3)User.query.filter(User.id==4).first()

- 查询名字结尾字符为g的所有数据[开始/包含]

User.query.filter(User.name.endswith('g')).all() --[User:wang, User:zhang, User:tang]

包含：

User.query.filter(User.name.contains('g')).all()　--[<User 1>, <User 2>, <User 5>]

获取第二个对象的名字：

list = User.query.filter(User.name.contains('g')).all()

list[1].name

- 查询名字不等于wang的所有数据[2种方式]

(1)!=: User.query.filter(User.name!='wang').all()

(2)not:User.query.filter(not(User.name=='wang')).all()

- 查询名字和邮箱都以 li 开头的所有数据[2种方式]

(1)and:User.query.filter(and(User.name.startswith('li'), User.email.startswith('li'))).all()

1. 不需要and\_:

User.query.filter(User.name.startswith('li'), User.email.startswith('li')).all()

- 查询password是 123456 或者 email 以 itheima.com 结尾的所有数据

User.query.filter(or\_(User.password=='123456', User.email.endswith('itheima.com'))).all()

- 查询id为 [1, 3, 5, 7, 9] 的用户列表

User.query.filter(User.id.in\_([1,3,5,7,9])).all()

- 查询name为liu的角色数据（重要）

User.query.filter(User.name=='liu').first().role.name

- 查询所有用户数据，并以邮箱排序

User.query.order\_by('email').all()

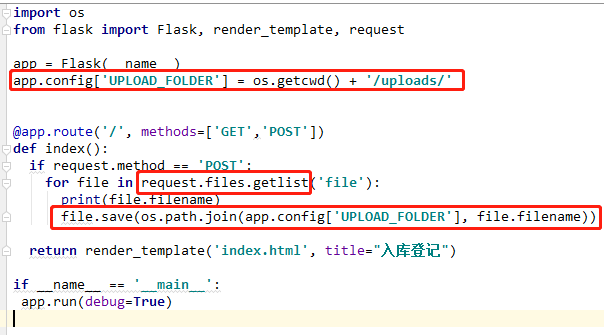
- 每页3个，查询第2页的数据

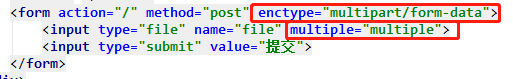
User.query.paginate(2,3,False).items 查询数据

User.query.paginate(2,3,False).page ---当前页

User.query.paginate(2,3,False).pages ---总页数

**6.多个文件上传**





1. **执行多条数据库语句**

**cursor.execute() // execute后需要立即执行一下commit()**

**conn.commit()**

1. **执行动态数据的插入**

**sql = "INSERT INTO `alltype` (`type\_name`,`pass\_num`) VALUES (%s, %s)"**

**cursor.execute(sql, (str(bt), pass\_num))**

**// 当有%d的时候用 sql % (xxx, xxx)**

**sql = "UPDATE `alltype` SET `pass\_num`=%d WHERE `type\_name`= ’%s’ "**

**cursor.execute(sql % (pass\_num, bt))**

1. **获取前台ajax传输的数据（POST方法发送数据）**

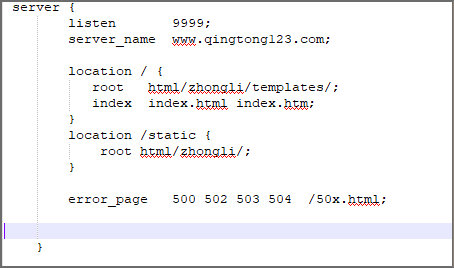
**前台：** var sendData = { 'name':'bob' }; **xmlhttp.send(JSON.stringify(sendData));**

**后台：**

**data\_json=request.get\_data().decode('utf-8')**

**data\_dict=json.loads(data\_json)**

1. **Nginx 配置**
2. 在一个IP下配置多个站点
3. 在服务器新建配置 nginx/vhosts/xxx.conf



xxx.conf

1. 在 nginx/conf/nginx.conf 下添加刚才的配置

include /usr/local/nginx/vhosts/qingtong.conf;

1. nginx重启
   1. ps -ef|grep nginx (查看进程号)
   2. kill -QUIT \*\*\*\*\* （杀死进程）
   3. 在nginx目录下: ./nginx