cookie和sesssion

1、请求 request

概述：

浏览器发送到服务器被flask接收以后 创建出请求对象 request ， 请求对象使用在视图函数中 来获取请求用户所带的数据

request由flask创建 使用的时候 导入就可以

导入：

from flask import request

request属性

属性	说明
url	完整的请求url地址
base_url	去掉传参的路由地址
host_url	只有协议、主机和端口
host	只有ip和端口号
path	请求路径
full_path	请求路径+参数
method	获取请求方式
remote_addr	客户端IP地址
args	获取GET传参。request.args.get(key)获取单个值 request.args.getlist获取多个值
form	获取POST传递过来的数据。request.form.get(key)获取单个值 request.form.getlist获取多个值
values	可以获取GET或POST传参
files	获取文件上传
headers	获取所有的请求头信息
cookies	获取cookie
json	获取请求过来的json数据

2、响应 response

(1) 使用return直接响应字符串

```
@app.route('/')
def index():
    return 'Hello Flask! '
    return 'Hello Flask! ', 404 #响应内容并响应状态码
```

(2) 通过make_response构造响应

导入：

from flask import make_reponse

```
@app.route('/')
def index():
    return make_response('Hello Flask! ')
    return make_response('Hello Flask! ', 404) #响应内容并响应状态码
```

注意：响应状态码默认为200

3、会话控制 COOKIE

在网站中，http请求是无状态的。也就是说即使第一次和服务器连接后并且登录成功后，第二次请求服务器依然不能知道当前请求是哪个用户。cookie的出现就是为了解决这个问题，第一次登录后服务器返回一些数据（cookie）给浏览器，然后浏览器保存在本地，当该用户发送第二次请求的时候，就会把上次请求存储的cookie数据自动的携带给服务器，服务器通过浏览器携带的数据就能判断当前是哪个用户了。cookie存储的数据量有限，不同的浏览器有不同的存储大小，但一般不超过4kb。因此使用cookie只能存储一些小量的数据。

(1) 设置cookie

格式

response.set_cookie(key,value,max_age,expires,path='/')

参数	说明
key	键
value	值
max_age	设置过期时间，单位秒，如果没有设置过期时间，默认为浏览器关闭后过期
expires	设置过期时间，以时间戳的形式设置
path	主域名，默认是'/',表示在当前站点可用

```
@app.route('/set_cookie/')
def set_cookie():
    res = make_response('设置cookie')
    res.set_cookie('name','zhangsan')
    return res
```

注意：默认存活时间为浏览会话结束（关闭浏览器）

(2) 设置cookie并设置过期时间

```
import time
#设置cookie并设置过期时间
@app.route('/set_cookie_lefttime/')
def set_cookie_lefttime():
    res = make_response('设置cookie并设置过期时间')
    # res.set_cookie('name','zhangsan',max_age=40)
    timeout = time.time()+40
    res.set_cookie('name','zhangsan',expires=timeout)
    return res
```

(3) 获取cookie

```
#获取cookie
@app.route('/get_cookie/')
def get_cookie():
    print(request.cookies)
    return 'cookie的值为{}'.format(request.cookies.get('name'))
```

(4) 删除cookie

```
@app.route('/delete_cookie/')
def delete_cookie():
    res = make_response('删除cookie')
    res.delete_cookie('name')
    # res.set_cookie('name','zhangsan',expires=0) #重新设置cookie 刚开始就死了
    return res
```

4、会话控制 session

session和cookie的作用有点类似，都是为了存储用户相关的信息。不同的是，cookie是存储在本地浏览器，而session存储在服务器。存储在服务器的数据会更加安全，不容易被窃取。但存储在服务器也有一定的弊端，就是会占用服务器的资源，但现在服务器已经发展至今，存储一些session信息还是绰绰有余的。

服务器要区分哪一个用户的请求 会根据cookie携带着的唯一sessionid来进行区分 session将数据存储于服务器端 安全性高于cookie

注意：

sessionid需要根据secret_key 来进行生成（如果没有 则报错）

导入：from flask import session

(1) 设置session

```
@app.route('/set_session/')
def set_session():
    session['name'] = 'zhansgan'
    return '设置session'
```

(2) 设置session并设置过期时间

```
@app.route('/session_lefttime/')
def session_lefttime():
    session.permanent = True #开启session存储持久化
    app.permanent_session_lifetime = timedelta(minutes=1) #存活一分钟
    session['name'] = 'zhansgan'
    return '设置session并设置过期时间'
```

(3) 获取session

```
@app.route('/get_session/')
def get_session():
    return '值为{}'.format(session.get('name'))
```

(4) 删除session

```python
@app.route('/delete_session/')
def delete_session():
    # session.pop(key)
    # session.pop('name')
    session.clear()
    return '删除session'
```

5. 消息闪烁

当用户发出请求后，状态发生了改变，需要系统给出警告、提示等信息时，通常是弹出一条消息，指示用户下一步的操作，用户可以手动关闭提示消息。而且整个的过程不会影响页面原有的显示。

- 配置

```
app.config["SECRET_KEY"] = "qieqw0923-0-klzd//../op[p%^]"
```

- 使用：
  - 在需要弹出消息时，使用`flash`函数保存闪烁消息

```python
@app.route('/')
def index():
    flash('大哥，又换签名了！')
    return render_template('index.html')
```

  - 在模板文件中显示闪烁消息时，可以通过函数`get_flashed_messages`获取
  - 从bootstrap上粘贴一个可消失的警告框

```
{% for message in get_flashed_messages() %}
    <div class="alert alert-warning alert-dismissible"
role="alert">
<button type="button" class="close" data-dismiss="alert" aria-
label="Close"><span aria-hidden="true">&times;</span>
</button>
    {{ message }}
    </div>
{% endfor %}
```