04-Flask进阶

Flask插件

flask-caching

安装
```
pip install flask-caching
```

初始化
```
from flask_cache import Cache
cache = Cache(config={
    'CACHE_TYPE': 'simple',
})
cache.init_app(app=app)
```

使用
在视图函数上添加缓存
```
@blue.route('/')
@cache.cached(timeout=30)
def home():
    print('加载数据')
    return 'home'
```

钩子

什么是钩子（中间件Middleware）
钩子或叫钩子函数，是指在执行函数和目标函数之间挂载的函数，框架开发者给调用方提供一个point-挂载点，是一种AOP切面编程思想.

常用的钩子函数
```
before_first_request: 处理第一次请求之前执行.
before_request: 在每次请求之前执行. 通常使用这个钩子函数预处理一些变量, 实现反爬等.
after_request: 注册一个函数, 如果没有未处理的异常抛出, 在每次请求之后运行.
teardown_appcontext: 当APP上下文被移除之后执行的函数,可以进行数据库的提交或者回滚.
```

AOP反爬策略
```
# 利用缓存反爬,相同ip地址1秒内不允许重复访问
key = request.remote_addr + "before"
value = cache.get(key)
if value:
    return '小伙子, 别爬了'
else:
    cache.set(key, 'aa', timeout=1)

# 反爬,防止非浏览器访问
```

```
    ua = request.user_agent  # 用户代理
    if not ua:
        return "hello"
        # abort(400)  # 可以抛出错误给用户
```

Flask内置对象

```
g:
  global全局对象
  g对象是专门用来保存用户的数据的
  g对象在一次请求中的所有的代码的地方，都是可以使用的
  突破变量存储位置限制，为数据传递添加了新的方式，比如我们在before_request产生一个数据在后面需要使用，可以保存在g对象中，在其他视图函数中就可以使用这个数据.

request:
  请求对象，可以获取客户端提交过来的所有请求信息

session:
  会话技术,服务端会话技术的接口

current_app:
    app的配置信息，app对象获取， current_app
    使用获取当前app需要注意，一定要在程序初始化完成之后
```

配置templates和static

如果想要修改templates模板目录或static静态目录,可以自己配置

在settings.py文件中添加BASE_DIR:

```
    import os
    BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
```

在__init__.py文件中添加static路径和templates路径:

```
    static_path = os.path.join(settings.BASE_DIR, 'static')
    template_path = os.path.join(settings.BASE_DIR, 'templates')
    app = Flask(__name__, static_folder=static_path, template_folder=template_path)
```

在views.py文件中访问模板:

```
    @blue.route('/hello/')
    def hello():
        return render_template('hello.html')
```

在模板中使用静态资源:

```
    <link rel="stylesheet" href="{{ url_for('static', filename='css/hello.css') }}">
```

前后端分离

json序列化： jsonify

```
json数据
  可以前后端数据交互时使用

jsonify():
    将字典转换成json
        @blue.route('/json/')
        def json():
            data = {
                'status': 1,
                'msg': 'ok',
            }
            return jsonify(data)
```