**session**

- session是在服务器上开辟一段空间用于保留浏览器和服务器交互时的重要数据
- 每个客户端都可以在服务器端有一个独立的Session
- http协议是无状态的：每次请求都是一次新的请求，不会记得之前通信的状态
- 客户端与服务器端的一次通信，就是一次会话
- 实现状态保持的方式：在客户端或服务器端存储与会话有关的数据
- 存储方式包括cookie、session，会话一般指session对象
- 使用cookie，所有数据存储在客户端，注意不要存储敏感信息
- 推荐使用sesison方式，所有数据存储在服务器端，在客户端cookie中存储session_id
- 状态保持的目的是在一段时间内跟踪请求者的状态，可以实现跨页面访问当前请求者的数据
- 注意：不同的请求者之间不会共享这个数据，与请求者一一对应
- 什么是session
    - session - 会话
    - 在服务器上开辟一段空间用于保留浏览器和服务器交互时的重要数据
- Django启用Session
    - 在 settings.py 文件中
    - 项INSTALLED_APPS列表中添加：
        - 'django.contrib.sessions',
    - 项MIDDLEWARE_CLASSES列表中添加：
        - 'django.contrib.sessions.middleware.SessionMiddleware',
- session的基本操作:
    - Session对于象是一个 QueryDict 字典, 可以用类拟于字典的方式进行操作
    - 保存 session 的值到服务器
        - `request.session[键] = 值`
        - 如: `request.session['KEY'] = VALUE`
    - 获取session的值
        - `VALUE = request.session['KEY']`
        - 或
        - `VALUE = request.session.get('KEY', 缺省值)`
    - 删除session的值
        - `del request.session['KEY']`

    - 在 settings.py 中有关 session 的设置

        1. SESSION_COOKIE_AGE 作用:指定sessionid在cookies中的保存时长 SESSION_COOKIE_AGE = 60*30
        2. SESSION_EXPIRE_AT_BROWSER_CLOSE = True 设置只要浏览器关闭时,session就失效

- 注: 当使用session时需要迁移数据库,否则会出现错误

```
$ python3 manage.py makemigrations
$ python3 manage.py migrate
```

- session 示例

```
# file : <项目名>/urls.py
from . import  views

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    # 增删改session
    url(r'^add_session', views.add_session),
    url(r'^mod_session/(\d+)', views.mod_session),
    url(r'^del_session', views.del_session),
    url(r'^show_session', views.show_session),
]

# file : <项目名>/views.py
from . import views
from django.http import HttpResponse
def add_session(request):
    request.session['mysession_var'] = 100
    responds = HttpResponse("添加session")
    return responds
def mod_session(request, new_value):
    request.session['mysession_var'] = new_value
    responds = HttpResponse("修改session成功")
    return responds
def del_session(request):
    try:
        del request.session['mysession_var']
        responds = HttpResponse("删除session成功")
    except:
        responds = HttpResponse("删除session失败")
    return responds
def show_session(request):
    mysession_var = request.session.get('mysession_var', '没有值!')
    print("mysession_var = ", mysession_var)
    return HttpResponse("mysession_var = " + str(mysession_var))
```

验证码

## 中间件 Middleware

- 中间件是 Django 请求/响应处理的钩子框架。它是一个轻量级的、低级的“插件”系统，用于全局改变 Django 的输入或输出。
- 每个中间件组件负责做一些特定的功能。例如，Django 包含一个中间件组件 AuthenticationMiddleware，它使用会话将用户与请求关联起来。
- 他的文档解释了中间件是如何工作的，如何激活中间件，以及如何编写自己的中间件。Django 具有一些内置的中间件，你可以直接使用。它们被记录在 built-in middleware reference 中。
- 中间件类:
    - 中间件类须继承自 `django.utils.deprecation.MiddlewareMixin`类
    - 中间件类须实现下列五个方法中的一个或多个:
        - `def process_request(self, request):` 执行视图之前被调用，在每个请求上调用，返回None或HttpResponse对象
        - `def process_view(self, request, callback, callback_args, callback_kwargs):` 调用视图之前被调用，在每个请求上调用，返回None或HttpResponse对象
        - `def process_response(self, request, response):` 所有响应返回浏览器之前被调用，在每个请求上调用，返回HttpResponse对象
        - `def process_exception(self, request, exception):` 当处理过程中抛出异常时调用，返回一个HttpResponse对象
        - `def process_template_response(self, request, response):` 在视图刚好执行完毕之后被调用，在每个请求上调用，返回实现了render方法的响应对象
    - 注： 中间件中的大多数方法在返回None时表示忽略当前操作进入下一项事件，当返回HttpResponese对象时表示此请求结果，直接返回给客户端
- 编写中间件类:

```
# file : middleware/mymiddleware.py
from django.http import HttpResponse, Http404
from django.utils.deprecation import MiddlewareMixin

class MyMiddleWare(MiddlewareMixin):
    def process_request(self, request):
        print("中间件方法 process_request 被调用")

    def process_view(self, request, callback, callback_args, callback_kwargs):
        print("中间件方法 process_view 被调用")

    def process_response(self, request, response):
        print("中间件方法 process_response 被调用")
```

```
        return response

    def process_exception(self, request, exception):
        print("中间件方法 process_exception 被调用")

    def process_template_response(self, request, response):
        print("中间件方法 process_template_response 被调用")
        return response
```

- 注册中间件:

```
# file : settings.py
MIDDLEWARE = [
    ...
    'middleware.mymiddleware.MyMiddleWare',
]
```

- 中间件的执行过程
    - 见verify应用
- 练习
    - 用中间件实现强制某个IP地址只能向/test 发送一次GET请求
    - 提示:
        - request.META['REMOTE_ADDR'] 可以得到远程客户端的IP地址
        - request.path_info 可以得到客户端访问的GET请求路由信息