Урок №8

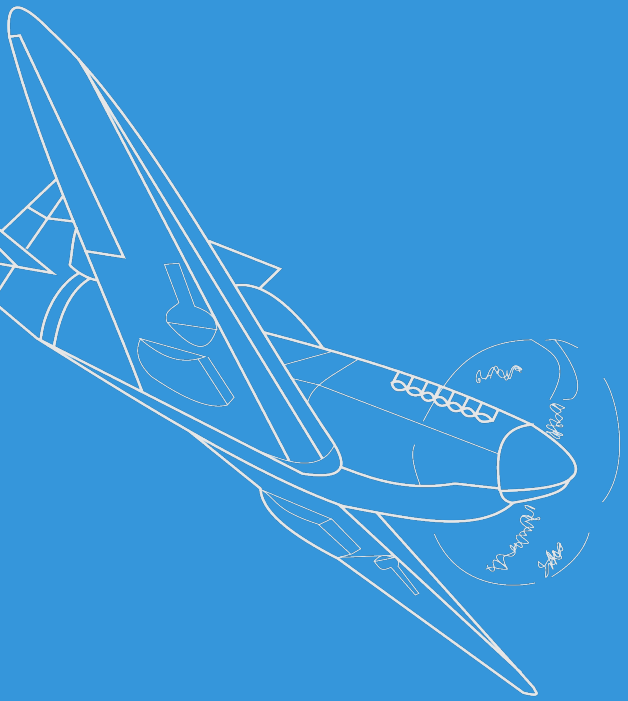
Авторизация в веб-приложениях

09 ноября 2020 года

Антон Кухтичев

Содержание занятия

1. Авторизация и аутентификация
2. Basic-авторизация
3. Cookie
4. Работа с cookie в Django
5. OAuth 2 авторизация



Аутентификация и авторизация

Аутентификация и авторизация

Аутентификация — предоставление доказательств, что вы на самом деле есть тот, кем идентифицировались (от слова “authentic” — истинный, подлинный).

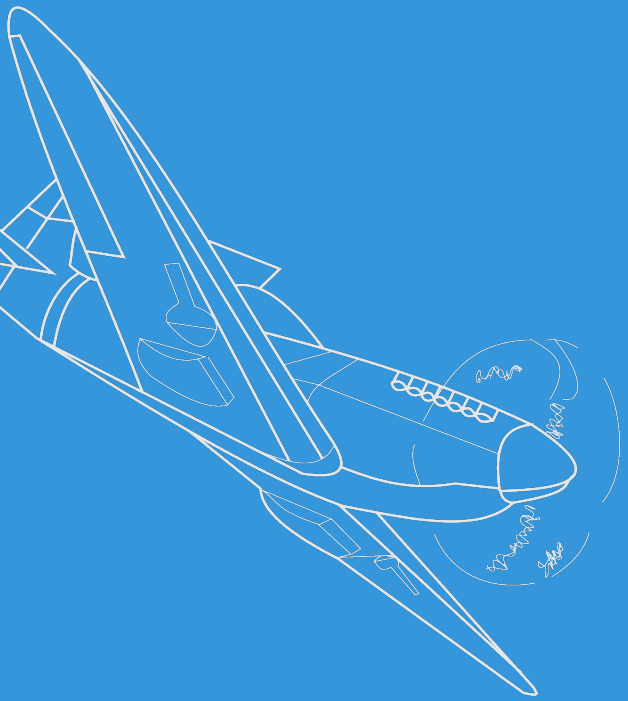
Авторизация — проверка, что вам разрешен доступ к запрашиваемому ресурсу.

Авторизация в веб-приложениях

HTTP — **stateless** протокол, т.е. не предполагает поддержания соединения между клиентом и сервером. Это значит, что сервер не может связать информацию о пользователе с конкретным соединением и вынужден загружать её при каждом запросе.

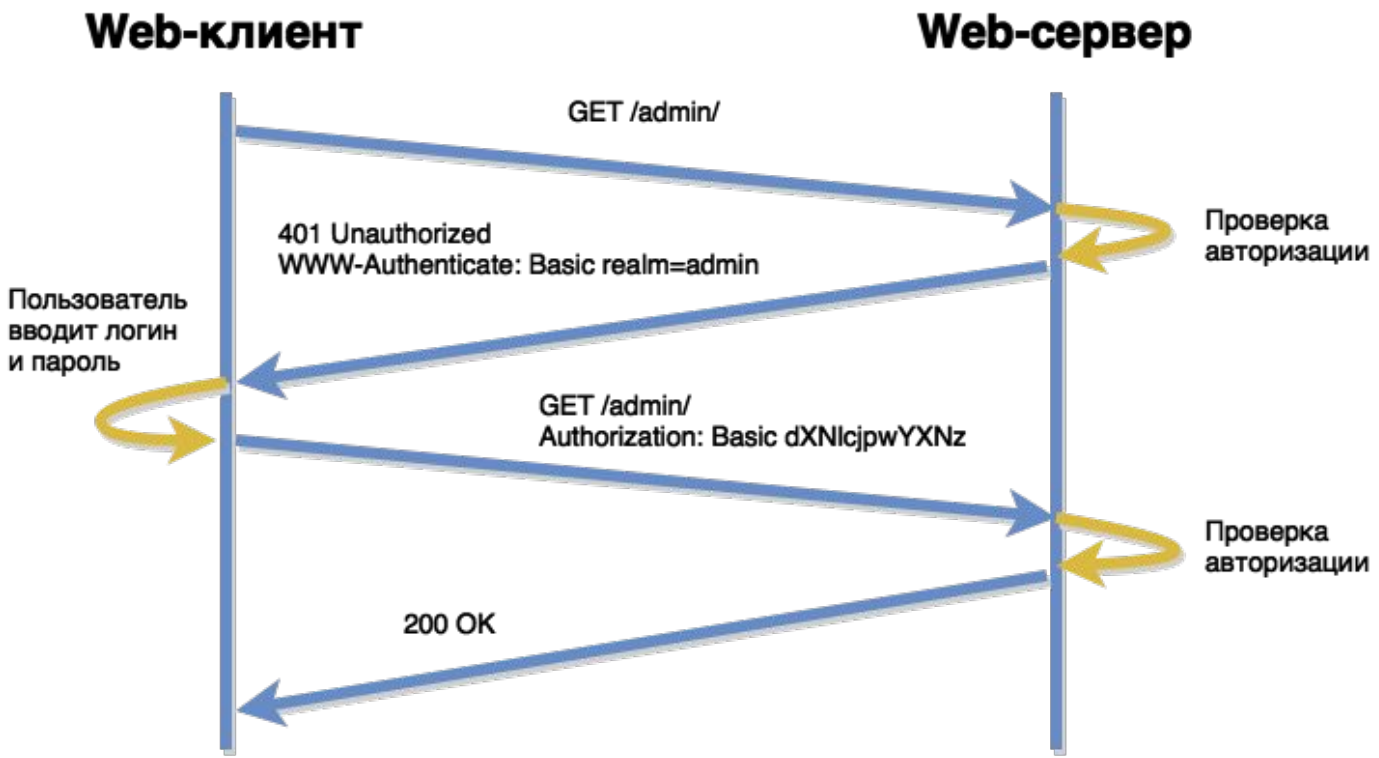
Виды авторизации

- Basic-авторизация
- Авторизация, основанная на куках (cookie-based)
- Авторизация через соц.сети (OAuth2)



Basic-авторизация

Basic HTTP Authorization



Заголовки и коды ответа

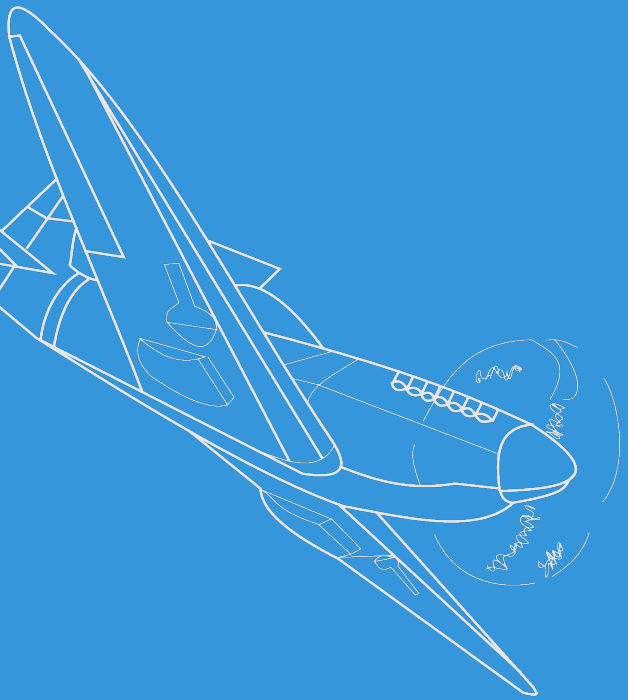
- `401 Unauthorized` — для доступа к ресурсу нужна авторизация
- `WWW-Authenticate: Basic realm="admin"` — запрос логина/пароля для раздела admin
- `Authorization: Basic Z2l2aTpkZXJwYXJvbA==` — передача логина/пароля в виде `base64(login + ':' + password)`
- `403 Forbidden` — логин/пароль не подходят
- `REMOTE_USER` — CGI переменная с именем авторизованного пользователя

Достоинства и недостатки

+ Простота и надежность;
+ Готовые модули для web-серверов;
+ Не требует написания кода;
– Логин/пароль передаются в открытом виде — нужен `https`;
– Невозможно изменить дизайн формы входа;
– Невозможно «сбросить» авторизацию.

Пример Basic-авторизации

Cookies



Cookies

Cookies — небольшие фрагменты данных, которые браузер хранит на стороне клиента и передает на сервер при каждом запросе.

Cookies привязаны к доменам, поэтому при каждом запросе сервер получает только «свои» cookies. Невозможно получить доступ к cookies с другого домена.

Cookies используются для поддержания состояния (state management) в протоколе HTTP и, в частности, для авторизации.

Аттрибуты Cookie

- `name=value` — имя и значение cookie
- `Expires` — время жизни cookie, по умолчанию — до закрытия окна
- `Domain` — домен cookie, по умолчанию — домен текущего URL
- `Path` — путь cookie, по умолчанию — путь текущего URL
- `Secure` — cookie должна передаваться только по https
- `HttpOnly` — cookie не доступна из JavaScript

Установка и удаление Cookies


```
Set-Cookie: sessid=d232rn38jd1023e1nm13r25z;
            Domain=.site.com; Path=/admin/;
            Expires=Mon, 9 Nov 2020 11:38:23 GMT;
            Secure; HttpOnly
Set-Cookie: lang=ru
Set-Cookie: sessid=xxx;
            Expires=Sun, 06 Nov 1994 08:49:37 GMT
```

Для удаления cookie, сервер устанавливает `Expires` в прошлом.

Получение Cookies


```
Cookie: sessid=d232rn38jd1023e1nm13r25z; lang=ru;
        csrftoken=vVqoyo5vzD3hWRHQDRpIHzVmKLfBQIGD;
```

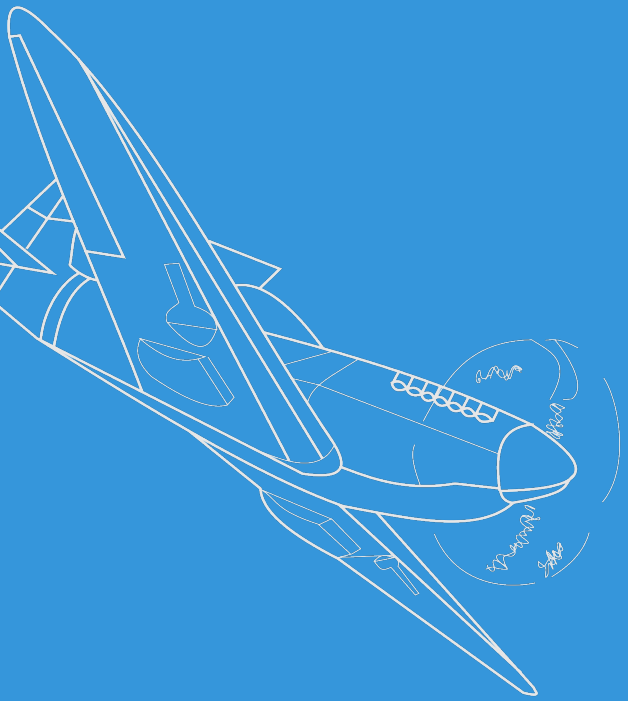
При каждом запросе браузер выбирает подходящие cookies и отправляет только их значения.

Правила выбора Cookies

Пусть URL=`http://my.app.site.com/blog/post/12`

Браузер выберет все cookies, у которых:

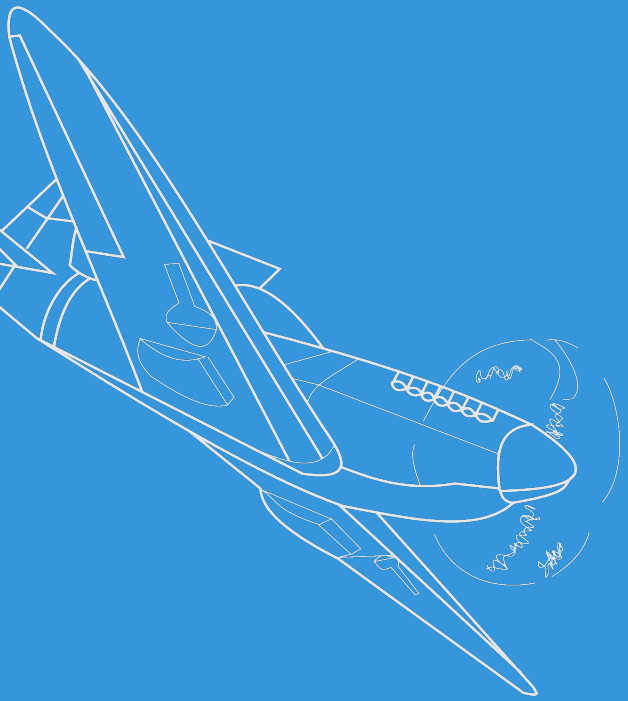
- Не истёк срок `Expires`
- `Domain` совпадает с `my.app.site.com` или явлестся суффиксом, например `Domain=.site.com`
- `Path` является префиксом `/blog/post/12`, например `Path=/blog/`
- Не стоит флага `Secure`



Работа с cookie в Django

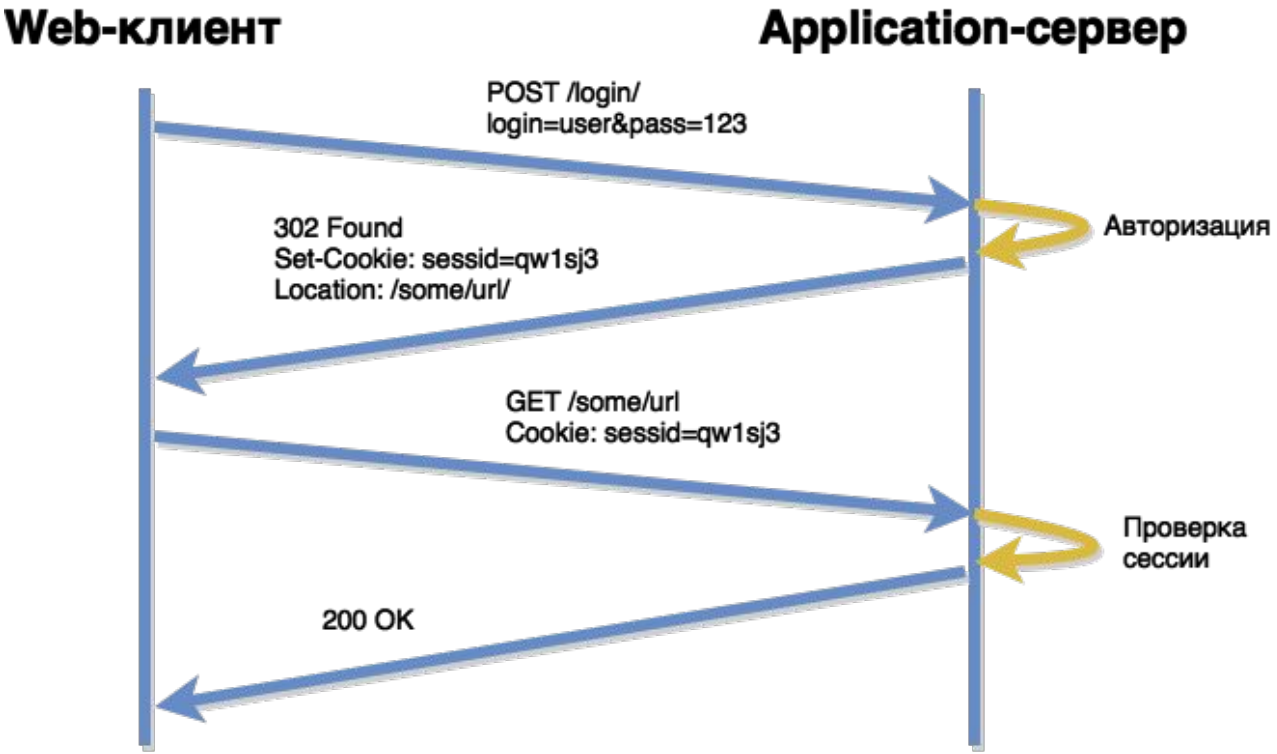
Работа с cookie в Django


```
# установка
response.set_cookie('sessid','asde132dk13d1')
response.set_cookie('sessid','asde132dk13d1',
domain='.site.com', path='/blog/', expires=(datetime.now() +
timedelta(days=30)))
# удаление
response.delete_cookie('another')
# получение
request.COOKIES # все cookies
request.COOKIES.get('sessid') # одна cookie
```



Cookie-based авторизация

Cookie-based авторизация



Встроенная авторизация Django

django.contrib.sessions

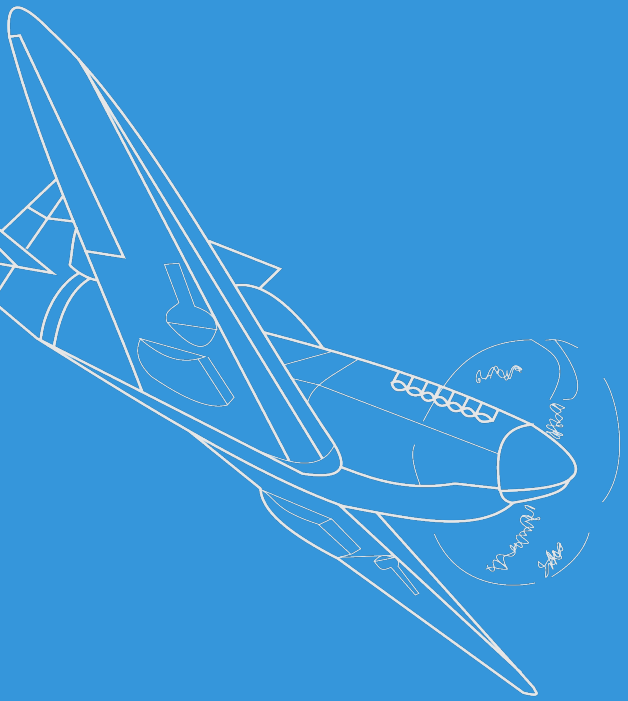
Предоставляет поддержку сессий, в том числе анонимных. Позволяет хранить в сессии произвольные данные, а не только ID пользователя. Позволяет хранить сессии в различных хранилищах, например Redis или Memcached.

```
def some_view(request):
    val = request.session['some_name']
    request.session.flush()
    request.session['some_name'] = 'val2'
```

django.contrib.auth

Предоставляет готовую модель `User` , готовую систему разделения прав, view для регистрации / входа / выхода. Используется другими приложениями, например `django.contrib.admin`

```
def some_view(request):
    user = request.user # Определено всегда!
    if user.is_authenticated():
        pass # обычный пользователь
    else:
        pass # анонимный пользователь
```



OAuth 2 авторизация

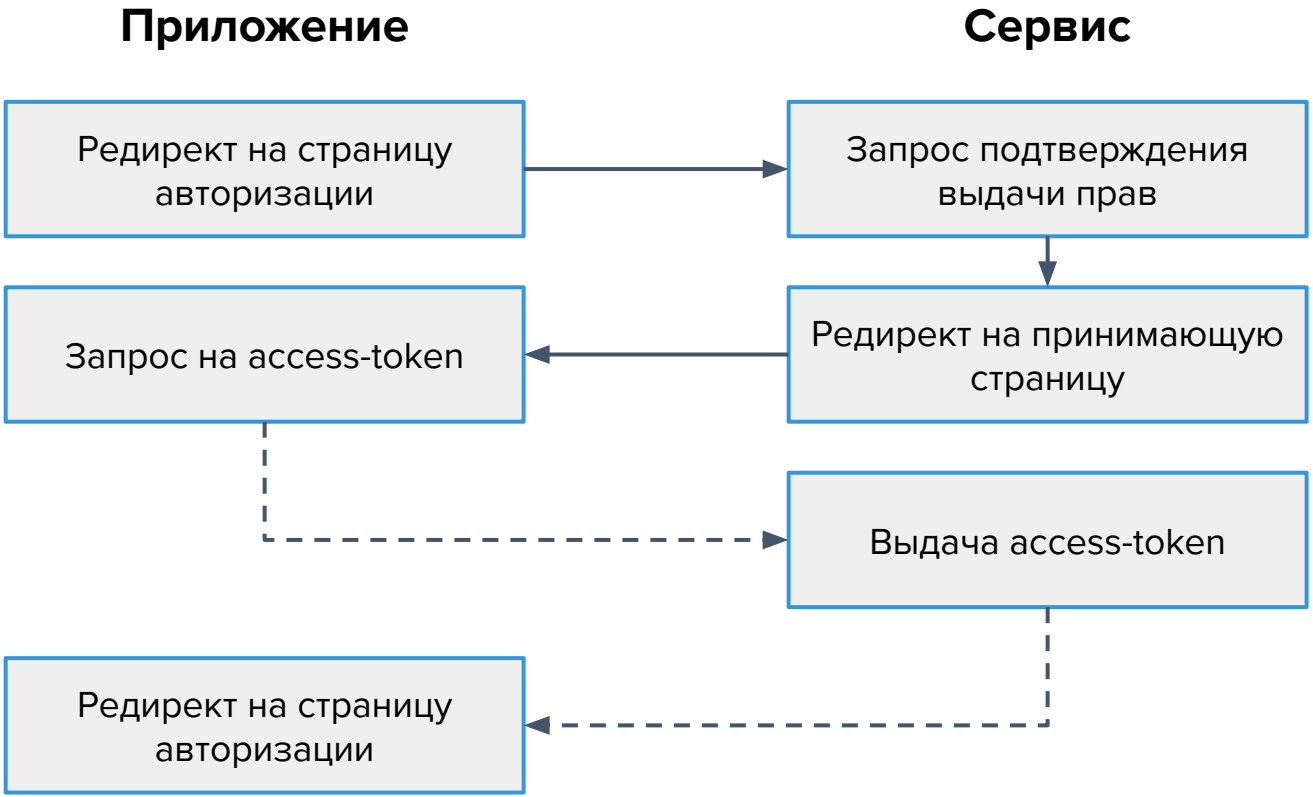
OAuth(2.0) авторизация

OAuth 2.0 — протокол авторизации, позволяющий выдать одному сервису (приложению) права на доступ к ресурсам пользователя на другом сервисе. Протокол избавляет от необходимости доверять приложению логин и пароль, а также позволяет выдавать ограниченный набор прав, а не все сразу.

OAuth(2.0) авторизация

Результатом авторизации является `access token` — некий ключ, предъявление которого является пропуском к защищенным ресурсам. Обращение к ним в самом простом случае происходит по HTTPS с указанием в заголовках или в качестве одного из параметров полученного `access token`.

OAuth2


Приложение
Сервис
Редирект на страницу авторизации
Запрос подтверждения выдачи прав
Запрос на access-token
Редирект на принимающую страницу
Выдача access-token
Редирект на страницу авторизации

Code time!

Пишем авторизацию через VK!

Шаг #1. Устанавливаем и мигрируем


```
(venv) pip install social-auth-app-django

(venv) ./manage.py migrate social_django
```

Шаг #2. Прописываем всё необходимое в settiongs.py


```
#settings.py
INSTALLED_APPS = [
    ...
    'social_django',
]

AUTHENTICATION_BACKENDS = [
    'social_core.backends.vk.VKOAuth2',
    #'social_core.backends.instagram.InstagramOAuth2',
    #'social_core.backends.facebook.FacebookOAuth2',
    'django.contrib.auth.backends.ModelBackend',
]
```

Шаг #3


```
# settings.py
TEMPLATES = [
    {
          ...
        'OPTIONS': {
            'context_processors': [
                ...
                'social_django.context_processors.backends',
            ],
        },
    },
]
```

Шаг #4


```
# settings.py
SOCIAL_AUTH_VK_OAUTH2_KEY = 'XXXXXXX'
SOCIAL_AUTH_VK_OAUTH2_SECRET = 'XXXXXXXXXXXXXXXXXXX'
SOCIAL_AUTH_URL_NAMESPACE = 'social'
SOCIAL_AUTH_VK_OAUTH2_SCOPE = ['email']

LOGIN_URL = 'login'
LOGIN_REDIRECT_URL = 'home'
LOGOUT_URL = 'logout'
LOGOUT_REDIRECT_URL = 'login'
```

Шаг #5


```
# urls.py
from django.contrib.auth import views as auth_views


urlpatterns = [
    ...
    path('login/', login_view, name='login'),
    path('logout/', auth_views.LogoutView.as_view(), name='logout'),
    path('social_auth/', include('social_django.urls',
namespace='social')),
]
```

Шаг #6


```
# login.html
...
<button>
  <a href="{% url 'social:begin' 'vk-oauth2' %}">Login via VK</a>
</button>
...
```

Домашнее задание № 8

1. Реализовать OAuth2-авторизацию для двух любых социальных сетей;
2. Навесить декоратор, проверяющий авторизацию при вызовах API;
3. Изменить запросы и код API так, чтобы учитывался текущий пользователь;

Рекомендуемая литература

[Авторизация в Django]
[Список сервисов для OAuth 2 авторизации в python-social-auth]

Для саморазвития (опционально)
[Чтобы не набирать двумя пальчиками]

Спасибо за внимание!

Антон Кухтичев

a.kukhtichev@mail.ru

@toshunster