Session

Session-Cookie机制

网景公司发明了Cookie技术，为了解决浏览器端数据存储问题。
每一次request请求时，会把此域名相关的Cookie发往服务器端。服务器端也可以使用response中的set-cookie来设置cookie值。

动态网页技术，也需要知道用户身份，但是HTTP是无状态协议，无法知道。必须提出一种技术，让客户端提交的信息可以表明身份，而且不能更改。这就是Session技术。
Session开启后，会为浏览器端设置一个Cookie值，即SessionID。
这个放置SessionID的Cookie是会话级的，浏览器不做持久化存储只放在内存中，并且浏览器关闭自动清除。
浏览器端发起HTTP请求后，这个SessionID会通过Cookie发到服务器端，服务器端就可以通过这个ID查到对应的一个字典结构。如果查无此ID，就为此浏览器重新生成一个SessionID，为它建立一个SessionID和空字典的映射关系。
可以在这个ID对应的Session字典中，存入键值对来保持与当前会话相关的信息。

- Session会定期过期清除
- Session占用服务器端内存
- Session如果没有持久化，如果服务程序崩溃，那么所有Session信息丢失
- Session可以持久化到数据库中，如果服务程序崩溃，那么可以从数据库中恢复

开启session支持

Django可以使用Session

- 在settings中，MIDDLEWARE设置中，启用'django.contrib.sessions.middleware.SessionMiddleware'。
- INSTALLED_APPS设置中，启用'django.contrib.sessions'。它是基于数据库存储的Session。
- Session不使用，可以关闭上述配置，以减少开销。
- 在数据库的表中的django_session表，记录session信息。但可以使用文件系统或其他cache来存储

登录登出实现

登录实现

原来登录成功，会使用Jwt的token发往客户端。现在不需要了，只需要在Session中记录登录信息即可。

```
def login(request:HttpRequest):
    # post json
    try:
        payload = simplejson.loads(request.body)
        email = payload['email']

        user = User.objects.get(email=email) # only one

        # 验证密码
        if bcrypt.checkpw(payload['password'].encode(), user.password.encode()):
            # 注释掉原来的jwt token代码
            # token = gen_token(user.id)
```

```
            # res = JsonResponse({
            #     'user':{
            #         'user_id':user.id,
            #         'name':user.name,
            #         'email':user.email,
            #     }, 'token':token
            # })
            # res.set_cookie('jwt', token)
            s = request.session
            # 只有登录成功才会在session中保存信息
            s.set_expiry(300) # 设置过期时长
            s['user_id'] = user.id # 用于判断是否登录
            s['user_info'] = '[**{} {} {}**]'.format(user.id, user.name, user.email)
            #s['user'] = user # user不可以json序列化，会报错
            return JsonResponse({
                'user':{
                    'user_id':user.id,
                    'name':user.name,
                    'email':user.email,
                }, 'user_info':s['user_info']
            })
        else:
            return HttpResponseBadRequest()

    except Exception as e:
        print(e)
        return HttpResponseBadRequest()
```

建议不要在Session中保存太多数据，也不要保存过于复杂的类型。

认证实现

取消原有判断HTTP header中是否提供了JWT信息，改为判断该SessionID是否能找到一个字典，这个字典中是否有登录成功后设置的user_id键值对信息。

```
def authenticate(viewfunc):
    def wrapper(request:HttpRequest):
        # 认证检测 "HTTP_JWT"
        try:
            # auth = request.META["HTTP_JWT"] # 会被加前缀HTTP_且全大写
            # payload = jwt.decode(auth, settings.SECRET_KEY, algorithms=[settings.ALG])# 篡改，
过期
            # print(payload, '~~~~~~~~~~~~~~~~~~')
            # print(datetime.datetime.now().timestamp(), '~~~~~~~~~~~~~~~')

            print('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
            print(type(request.session), request.session)
            payload: SessionStore = request.session

            print(payload.items())
            print(payload['user_id']) # 登录成功才会有
```

```
            user = User.objects.get(pk=payload['user_id'])  # 以后要注意查询条件

            request.user = user
            print(user)
            print('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')

        except Exception as e:
            print(e)
            return HttpResponse(status=401)

        ret = viewfunc(request)# 调用视图函数
        # 特别注意view调用的时候，里面也有返回异常
        return ret
    return wrapper
```

登出代码

```
@authenticate # 没有登录成功过，就不用登出了，所以要认证
def logout(request):
    s = '{} logout ok.'.format(request.session['user_id'])
    #del request.session['user_id'] # 不会清除数据库中记录
    request.session.flush()  # 清空当前session， 删除对应表记录
    return HttpResponse(s)
```

登出时，需要调用flush方法，清除session，清除数据库记录。

登录成功，为当前session在django_session表中增加一条记录，如果没有登出操作，那么该记录不会消失。
Django也没有自动清除失效记录的功能。

但Django提供了一个命令clearsessions，建议放在cron中定期执行。

```
django-admin.py clearsessions
manage.py clearsessions
```

不需要认证的view函数中使用Session

在当前会话中，每次request请求中都会得到浏览器端发出的SessionID，因此都可以在服务器端找到该ID对应的Session字典，可以使用request.session访问。
所有请求都可以使用request.session对象，ID找不到可以认为返回个空字典。

```
def test1(request): # 视图函数，需要配置url映射
    if request.session.get('user_id'): # 访问Session字典中的值
        print(request.session.items())
        return HttpResponse('test1 ok')
    else:
        return HttpResponseBadRequest() # 没有此信息，说明没有登录成功过
```