day83 drf

回顾和补充

1. restful规范

```
1. 建议用https代替http
2. 在URL中体现api，添加api标识
    https://www.cnblogs.com/xwgblog/p/11812244.html   # 错误
    https://www.cnblogs.com/api/xwgblog/p/11812244.html  # 正确
    https://api.cnblogs.com/xwgblog/p/11812244.html # 正确

    建议：https://www.cnblogs.com/api/...
3. 在URL中要体现版本
    https://www.cnblogs.com/api/v1/userinfo/
    https://www.cnblogs.com/api/v2/userinfo/
4. 一般情况下对于api接口，用名词不用动词。
    https://www.cnblogs.com/api/v1/userinfo/
5. 如果有条件的话，在URL后面进行传递。
    https://www.cnblogs.com/api/v1/userinfo/?page=1&category=2
6. 根据method不同做不同操作
    get/post/put/patch/delete
```

2. 简述drf组件。
3. 类继承关系

```
class View(object):
    def dipatch(self):
        print(123)

class APIView(View):
    def dipatch(self):
        method = getattr(self,"get")
        return method()

class GenericAPIView(APIView):
    queryset = None
    serilizer_class = None
    def get_queryset(self):
        return self.queryset

    def get_serilizer(self,*arg,**kwargs):
        cls = self.get_serilizer_class()
        return cls(*arg,**kwargs)

    def get_serilizer_class(self):
        return self.serilizer_class
class ListModelMixin(object):
    def list(self):
        queryset = self.get_queryset()
        ser = self.get_serilizer(queryset,many=True)
```

```
        return Reponse(ser.data)

class ListAPIView(ListModelMixin,GenericAPIView):
    def get(self):
        return self.list(...)


class TagView(ListAPIView):
    queryset = models.User.object.all()
    serilizer_class = TagSerilizer


obj = TagView()
x = obj.dispatch()
给用户返回x
```

4. 继承关系

```
class View(object):
    def dipatch(self):
        print(123)

class APIView(View):
    version_class = settings.xxx
    parser_class = settings.sxx
    permision_classes = []

    def dipatch(self):

        self.initial()

        method = getattr(self,"get")
        return method()

    def initial(self):
        self.version_class()
        self.parser_class()
        for item in self.permision_classes:
            item()

class GenericAPIView(APIView):
    queryset = None
    serilizer_class = None
    def get_queryset(self):
        return self.queryset

    def get_serilizer(self,*arg,**kwargs):
        cls = self.get_serilizer_class()
        return cls(*arg,**kwargs)

    def get_serilizer_class(self):
        return self.serilizer_class
class ListModelMixin(object):
    def list(self):
        queryset = self.get_queryset()
        ser = self.get_serilizer(queryset,many=True)
        return Reponse(ser.data)
```

```
class ListAPIView(ListModelMixin,GenericAPIView):
    def get(self):
        return self.list(...)


class TagView(ListAPIView):
    queryset = models.User.object.all()
    serilizer_class = TagSerilizer
    version_class = URLPathClass
    parser_class = JSONParser
    permission_classes = [Foo,Bar ]

obj = TagView()
x = obj.dispatch()
给用户返回x
```

今日概要

- request请求对象封装
- 版本
- 认证
- 权限
- 频率限制

今日详细

1.请求的封装

```
class HttpRequest(object):
    def __init__(self):
        pass

    @propery
    def GET(self):
        pass

    @propery
    def POST(self):
        pass

    @propery
    def body(self):
        pass

class Request(object):
    def __init__(self,request):
        self._request = request

    def data(self):
```

```
        if content-type == "application/json"
            reutrn json.loads(self._request.body.decode('urf-8'))
        elif content-type == "x-www-...":
            return self._request.POST

    def query_params(self):
        return self._reqeust.GET

req = HttpRequest()
request = Request(req)

request.data
request.query_prams
request._request.GET
request._request.POST
request._request.body
```

drf入口请求流程：

- 路由

```
urlpatterns = [
    url(r'^order/$', views.OrderView.as_view()),
]
```

- 视图关系

```
class View(object):
    @classonlymethod
    def as_view(cls, **initkwargs):
        def view(request, *args, **kwargs):
            return self.dispatch(request, *args, **kwargs)
        return view

class APIView(View):

    @classmethod
    def as_view(cls, **initkwargs):
        view = super().as_view(**initkwargs)
        return csrf_exempt(view)
    def dispatch(self, request, *args, **kwargs):
        # 新request内部包含老request(_reuqest=老request)
        request = self.initialize_request(request, *args, **kwargs)
        self.request = request

        self.initial(request, *args, **kwargs)

        # 通过反射执行“get”方法，并传入新的request
        handler = getattr(self, request.method.lower())
        response = handler(request, *args, **kwargs) # get(requst)
        return self.response

class OrderView(APIView):

    def get(self,request,*args,**kwargs):
        return Response('海狗')
```

- 路由

```
urlpatterns = [
    url(r'^order/$', views.OrderView.as_view()),
]
```

- 视图关系

```
class View(object):
    @classonlymethod
    def as_view(cls, **initkwargs):
        def view(request, *args, **kwargs):
            return self.dispatch(request, *args, **kwargs)
        return view

class APIView(View):

    @classmethod
    def as_view(cls, **initkwargs):
        view = super().as_view(**initkwargs)
        return csrf_exempt(view)
    def dispatch(self, request, *args, **kwargs):
        # 新request
        request = self.initialize_request(request, *args, **kwargs)
        self.request = request

        self.initial(request, *args, **kwargs)

        # 通过反射执行"get"方法，并传入新的request
        handler = getattr(self, request.method.lower())
        response = handler(request, *args, **kwargs) # get(requst)
        return self.response

class OrderView(APIView):

    def get(self,request,*args,**kwargs):
        return Response('海狗')
```

python

2.版本

```
from django.conf.urls import url,include
from django.contrib import admin
from . import views
urlpatterns = [
    url(r'^order/$', views.OrderView.as_view()),
]
```

```
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework.request import Request

class OrderView(APIView):
    def get(self,request,*args,**kwargs):
        print(request.version)
        print(request.versioning_scheme)
        return Response('...')

    def post(self,request,*args,**kwargs):
        return Response('post')
```

源码流程：

```
class APIView(View):
    versioning_class = api_settings.DEFAULT_VERSIONING_CLASS

    def dispatch(self, request, *args, **kwargs):

        # ####################### 第一步 ############################
        """
        request,是django的request，它的内部有：
request.GET/request.POST/request.method
        args,kwargs是在路由中匹配到的参数，如：
            url(r'^order/(\d+)/(?P<version>\w+)/$', views.OrderView.as_view()),
            http://www.xxx.com/order/1/v2/
        """
        self.args = args
        self.kwargs = kwargs

        """
        request = 生成了一个新的request对象，此对象的内部封装了一些值。
        request = Request(request)
            - 内部封装了 _request = 老的request
        """
        request = self.initialize_request(request, *args, **kwargs)
        self.request = request

        self.headers = self.default_response_headers  # deprecate?

        try:
            # ####################### 第二步 ############################
            self.initial(request, *args, **kwargs)

            执行视图函数。。

    def initial(self, request, *args, **kwargs):

        # ############### 2.1 处理drf的版本 ##############
        version, scheme = self.determine_version(request, *args, **kwargs)
        request.version, request.versioning_scheme = version, scheme
        ...

    def determine_version(self, request, *args, **kwargs):
        if self.versioning_class is None:
            return (None, None)
        scheme = self.versioning_class() # obj = XXXXXXXXXXXX()
        return (scheme.determine_version(request, *args, **kwargs), scheme)

class OrderView(APIView):
    versioning_class = URLPathVersioning
    def get(self,request,*args,**kwargs):
        print(request.version)
        print(request.versioning_scheme)
        return Response('...')

    def post(self,request,*args,**kwargs):
        return Response('post')
```

```
class URLPathVersioning(BaseVersioning):
```

```
    """
    urlpatterns = [
        url(r'^(?P<version>[v1|v2]+)/users/$', users_list, name='users-list'),

    ]
    """
    invalid_version_message = _('Invalid version in URL path.')

    def determine_version(self, request, *args, **kwargs):
        version = kwargs.get(self.version_param, self.default_version)
        if version is None:
            version = self.default_version

        if not self.is_allowed_version(version):
            raise exceptions.NotFound(self.invalid_version_message)
        return version
```

使用（局部）

- url中写version

```
url(r'^(?P<version>[v1|v2]+)/users/$', users_list, name='users-list'),
```

- 在视图中应用

```
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework.request import Request
from rest_framework.versioning import URLPathVersioning


class OrderView(APIView):

    versioning_class = URLPathVersioning
    def get(self,request,*args,**kwargs):
        print(request.version)
        print(request.versioning_scheme)
        return Response('...')

    def post(self,request,*args,**kwargs):
        return Response('post')
```

- 在settings中配置

```
REST_FRAMEWORK = {
    "PAGE_SIZE":2,

 "DEFAULT_PAGINATION_CLASS":"rest_framework.pagination.PageNumberPagination"
,
    "ALLOWED_VERSIONS":['v1','v2'],
    'VERSION_PARAM':'version'
}
```

使用（全局）推荐

- url中写version

```
url(r'^(?P<version>[v1|v2]+)/users/$', users_list, name='users-list'),

url(r'^(?P<version>\w+)/users/$', users_list, name='users-list'),
```

- 在视图中应用

```
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework.request import Request
from rest_framework.versioning import URLPathVersioning


class OrderView(APIView):
    def get(self,request,*args,**kwargs):
        print(request.version)
        print(request.versioning_scheme)
        return Response('...')

    def post(self,request,*args,**kwargs):
        return Response('post')
```

- 在settings中配置

```
REST_FRAMEWORK = {
    "PAGE_SIZE":2,

 "DEFAULT_PAGINATION_CLASS":"rest_framework.pagination.PageNumberPagination"
,

 "DEFAULT_VERSIONING_CLASS":"rest_framework.versioning.URLPathVersioning",
    "ALLOWED_VERSIONS":['v1','v2'],
    'VERSION_PARAM':'version'
}
```

3.认证（面试）

```
from django.conf.urls import url,include
from django.contrib import admin
from . import views
urlpatterns = [
    url(r'^login/$', views.LoginView.as_view()),
    url(r'^order/$', views.OrderView.as_view()),
    url(r'^user/$', views.UserView.as_view()),
]
```

```
import uuid
from django.shortcuts import render
from django.views import View
from django.views.decorators.csrf import csrf_exempt
from django.utils.decorators import method_decorator
from rest_framework.versioning import URLPathVersioning
```

```
from rest_framework.views import APIView
from rest_framework.response import Response

from . import models

class LoginView(APIView):

    def post(self,request,*args,**kwargs):
        user_object = models.UserInfo.objects.filter(**request.data).first()
        if not user_object:
            return Response('登录失败')
        random_string = str(uuid.uuid4())
        user_object.token = random_string
        user_object.save()
        return Response(random_string)

class MyAuthentication:
    def authenticate(self, request):
        """
        Authenticate the request and return a two-tuple of (user, token).
        """
        token = request.query_params.get('token')
        user_object = models.UserInfo.objects.filter(token=token).first()
        if user_object:
            return (user_object,token)
        return (None,None)

class OrderView(APIView):
    authentication_classes = [MyAuthentication, ]
    def get(self,request,*args,**kwargs):
        print(request.user)
        print(request.auth)
        return Response('order')

class UserView(APIView):
    authentication_classes = [MyAuthentication,]
    def get(self,request,*args,**kwargs):
        print(request.user)
        print(request.auth)
        return Response('user')
```

源码分析

```
class Request:

    def __init__(self, request,authenticators=None):
        self._request = request
        self.authenticators = authenticators or ()

    @property
    def user(self):
        """
        Returns the user associated with the current request, as authenticated
        by the authentication classes provided to the request.
        """
        if not hasattr(self, '_user'):
            with wrap_attributeerrors():
```

```
                self._authenticate()
        return self._user

    def _authenticate(self):
        """
        Attempt to authenticate the request using each authentication instance
        in turn.
        """
        for authenticator in self.authenticators:
            try:
                user_auth_tuple = authenticator.authenticate(self)
            except exceptions.APIException:
                self._not_authenticated()
                raise

            if user_auth_tuple is not None:
                self._authenticator = authenticator
                self.user, self.auth = user_auth_tuple
                return

        self._not_authenticated()

    @user.setter
    def user(self, value):
        """
        Sets the user on the current request. This is necessary to maintain
        compatibility with django.contrib.auth where the user property is
        set in the login and logout functions.

        Note that we also set the user on Django's underlying `HttpRequest`
        instance, ensuring that it is available to any middleware in the stack.
        """
        self._user = value
        self._request.user = value
```

```
class APIView(View):
    authentication_classes = api_settings.DEFAULT_AUTHENTICATION_CLASSES

    def dispatch(self, request, *args, **kwargs):
        """
        `.dispatch()` is pretty much the same as Django's regular dispatch,
        but with extra hooks for startup, finalize, and exception handling.
        """
        # ####################### 第一步 ############################
        """
        request,是django的request，它的内部有:
request.GET/request.POST/request.method
        args,kwargs是在路由中匹配到的参数，如:
            url(r'^order/(\d+)/(?P<version>\w+)/$', views.OrderView.as_view()),
            http://www.xxx.com/order/1/v2/
        """
        self.args = args
        self.kwargs = kwargs
```

```
        """
        request = 生成了一个新的request对象，此对象的内部封装了一些值。
        request = Request(request)
            - 内部封装了 _request = 老的request
            - 内部封装了 authenticators = [MyAuthentication(), ]
        """
        request = self.initialize_request(request, *args, **kwargs)
        self.request = request

    def initialize_request(self, request, *args, **kwargs):
        """
        Returns the initial request object.
        """
        parser_context = self.get_parser_context(request)

        return Request(
            request,
            parsers=self.get_parsers(),
            authenticators=self.get_authenticators(), # [MyAuthentication(),]
            negotiator=self.get_content_negotiator(),
            parser_context=parser_context
        )

    def get_authenticators(self):
        """
        Instantiates and returns the list of authenticators that this view can
use.
        """
        return [ auth() for auth in self.authentication_classes ]

class LoginView(APIView):
    authentication_classes = []
    def post(self,request,*args,**kwargs):
        user_object = models.UserInfo.objects.filter(**request.data).first()
        if not user_object:
            return Response('登录失败')
        random_string = str(uuid.uuid4())
        user_object.token = random_string
        user_object.save()
        return Response(random_string)

class OrderView(APIView):
    # authentication_classes = [TokenAuthentication, ]
    def get(self,request,*args,**kwargs):
        print(request.user)
        print(request.auth)
        if request.user:
            return Response('order')
        return Response('滚')

class UserView(APIView):
    同上
```

总结

当用户发来请求时，找到认证的所有类并实例化成为对象列表，然后将对象列表封装到新的request对象中。

以后在视同中调用request.user

在内部会循环认证的对象列表，并执行每个对象的authenticate方法，该方法用于认证，他会返回两个值分别会赋值给
request.user/request.auth

作业：将认证的功能添加到呼啦圈中。

- 登录表
- 登录视图
- 写认证类
- 应用认证类：全局应用
- Login视图不应用认证

4.权限

```
from rest_framework.permissions import BasePermission
from rest_framework import exceptions

class MyPermission(BasePermission):
    message = {'code': 10001, 'error': '你没权限'}
    def has_permission(self, request, view):
        """
        Return `True` if permission is granted, `False` otherwise.
        """
        if request.user:
            return True

        # raise exceptions.PermissionDenied({'code': 10001, 'error': '你没权限'})
        return False

    def has_object_permission(self, request, view, obj):
        """
        Return `True` if permission is granted, `False` otherwise.
        """
        return False
```

```
class OrderView(APIView):
    permission_classes = [MyPermission,]
    def get(self,request,*args,**kwargs):
        return Response('order')


class UserView(APIView):
    permission_classes = [MyPermission, ]
    def get(self,request,*args,**kwargs):
        return Response('user')
```

```
REST_FRAMEWORK = {
    "PAGE_SIZE":2,
    "DEFAULT_PAGINATION_CLASS":"rest_framework.pagination.PageNumberPagination",
    "DEFAULT_VERSIONING_CLASS":"rest_framework.versioning.URLPathVersioning",
    "ALLOWED_VERSIONS":['v1','v2'],
    'VERSION_PARAM':'version',
    "DEFAULT_AUTHENTICATION_CLASSES":["kka.auth.TokenAuthentication",]
}
```

源码分析

```
class APIView(View):
    permission_classes = api_settings.DEFAULT_PERMISSION_CLASSES

    def dispatch(self, request, *args, **kwargs):
        封装request对象
        self.initial(request, *args, **kwargs)
        通过反射执行视图中的方法

    def initial(self, request, *args, **kwargs):
        版本的处理
        # 认证
        self.perform_authentication(request)

        # 权限判断
        self.check_permissions(request)


        self.check_throttles(request)

    def perform_authentication(self, request):
        request.user

    def check_permissions(self, request):
        # [对象,对象, ]
        for permission in self.get_permissions():
            if not permission.has_permission(request, self):
                self.permission_denied(request, message=getattr(permission,
'message', None))
    def permission_denied(self, request, message=None):
        if request.authenticators and not request.successful_authenticator:
            raise exceptions.NotAuthenticated()
        raise exceptions.PermissionDenied(detail=message)

    def get_permissions(self):
        return [permission() for permission in self.permission_classes]

class UserView(APIView):
    permission_classes = [MyPermission, ]

    def get(self,request,*args,**kwargs):
        return Response('user')
```

作业：实现呼啦圈

- 视图：ListAPIView
- 序列化：多个序列化
- 分页：给列表页面做分页
- 版本：版本设置
- 认证：认证流程 request.user赋值
- 权限：能否访问

```
def xx(func):
    def inner(..):
        return func()
    return inner

@xx
def func():
    pass

print(func)
func =xx(func)
```

```
def xx(func):
    def inner(..):
        return func()
    return inner


def func():
    pass

func =xx(func)

print(func)
```