Ask Company

리액트와 함께 장고 시작하기 / 장고 DRF

APIView, JSON 응답 뷰 만들기

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

Serializer를 통한 뷰 처리

Form 처리와 유사한 방식으로 동작

```
class PostSerializer(serializers.ModelSerializer):
    class Meta:
        model = Post
        fields = '__all__'

# Views
serializer = PostSerializer(data=request.POST)
if serializer.is_valid():
    return JsonResonse(serializer.data, status=201)
return JsonResponse(serializer.errors, status=400)
```

주의) Form 생성자의 첫번째 인자는 data이지만,
Serializer 생성자의 첫번째 인자는 instance입니다.

DRF의 기본 CBV인 APIView

APIView 클래스 혹은 @api_view 장식자

View에 여러 기본 속성을 부여합니다.

1. renderer_classes : 직렬화 class 다수
2. parser_classes : 비직렬화 class 다수
3. authentication_classes : 인증 class 다수
4. throttle_classes : 사용량 제한 class 다수
5. permission_classes : 권한 class 다수
6. content_negotiation_class : 요청에 따라 적절한 직렬화/비직렬화 class를 선택하는 class
7. metadata_class : 메타 정보를 처리하는 class
8. versioning_class : 요청에서 API버전 정보를 탐지하는 class

https://github.com/encode/django-rest-framework/blob/3.11.0/rest_framework/views.py - L104

각 옵션의 디폴트 값 (1/2)

1) renderer_classes

- rest_framework.renderers.JSONRenderer : JSON 직렬화
- rest_framework.renderers.TemplateHTMLRendere` : HTML 페이지 직렬화

2) parser_classes

- rest_framework.parsers.JSONParser : JSON 포맷 처리
- rest_framework.parsers.FormParser
- rest_framework.parsers.MultiPartParser

3) authentication_classes

- rest_framework.authentication.SessionAuthentication : 세션에 기반한 인증
- rest_framework.authentication.BasicAuthentication : HTTP Basic 인증

4) throttle_classes

- 빈 튜플

각 옵션의 디폴트 값 (2/2)

permission_classes

- rest_framework.permissions.AllowAny : 누구라도 접근 허용

content_negotiation_class

- rest_framework.negotiation.DefaultContentNegotiation
- 같은 URL로의 요청이지만, JSON응답을 요구하는 것이냐 / HTML응답을 요구하는 것인지 판단

metadata_class

- rest_framework.metadata.SimpleMetadata

versioning_class

- None : API 버전 정보를 탐지하지 않겠다.
- 요청 URL에서, GET인자에서, HEADER에서 버전정보를 탐지하여, 해당 버전의 API뷰가 호출되도록 합니다.

APIView와 @api_view

공식 튜토리얼 : https://www.django-rest-framework.org/tutorial/3-class-based-views/

DRF의 2가지 기본 뷰

1. APIView : 클래스 기반 뷰
2. @api_view : 함수 기반 뷰를 위한 장식자

APIView

하나의 CBV이므로 → 하나의 URL만 처리 가능

각 method(get, post, put, delete)에 맞게 멤버함수를 구현하면, 해당 method 요청이 들어올 때 호출

1. 직렬화/비직렬화 처리 (JSON 등)
2. 인증 체크
3. 사용량 제한 체크 : 호출 허용량 범위인지 체크
4. 권한 클래스 지정 : 비인증/인증 유저에 대해 해당 API 호출을 허용할 것인지를 결정
5. 요청된 API 버전 문자열을 탐지하여, request.version에 저장

APIView ㄴ dispatch

```
478     # Note: Views are made CSRF exempt from within `as_view` as to prevent
479     # accidental removal of this exemption in cases where `dispatch` needs to
480     # be overridden.
481     def dispatch(self, request, *args, **kwargs):
482         """
483         `.dispatch()` is pretty much the same as Django's regular dispatch,
484         but with extra hooks for startup, finalize, and exception handling.
485         """
486         self.args = args
487         self.kwargs = kwargs
488         request = self.initialize_request(request, *args, **kwargs)
489         self.request = request
490         self.headers = self.default_response_headers  # deprecate?
491
492         try:
493             self.initial(request, *args, **kwargs)
494
495             # Get the appropriate handler method
496             if request.method.lower() in self.http_method_names:
497                 handler = getattr(self, request.method.lower(),
498                                   self.http_method_not_allowed)
499             else:
500                 handler = self.http_method_not_allowed
501
502             response = handler(request, *args, **kwargs)
503
504         except Exception as exc:
505             response = self.handle_exception(exc)
506
507         self.response = self.finalize_response(request, response, *args, **kwargs)
508         return self.response
```

```
395     def initial(self, request, *args, **kwargs):
396         """
397         Runs anything that needs to occur prior to calling the method handler.
398         """
399         self.format_kwarg = self.get_format_suffix(**kwargs)
400
401         # Perform content negotiation and store the accepted info on the request
402         neg = self.perform_content_negotiation(request)
403         request.accepted_renderer, request.accepted_media_type = neg
404
405         # Determine the API version, if versioning is in use.
406         version, scheme = self.determine_version(request, *args, **kwargs)
407         request.version, request.versioning_scheme = version, scheme
408
409         # Ensure that the incoming request is permitted
410         self.perform_authentication(request)
411         self.check_permissions(request)
412         self.check_throttles(request)
```

https://github.com/encode/django-rest-framework/blob/3.11.0/rest_framework/views.py#L493

클래스 형태) APIView 구현 샘플 (list/create)

```python
from rest_framework.response import Response
from rest_framework.views import APIView
from .models import Post
from .serializers import PostSerializer

class PostListAPIView(APIView):

    def get(self, request):
        qs = Post.objects.all()
        serializer = PostSerializer(qs, many=True)
        return Response(serializer.data)

    def post(self, request):
        serializer = PostSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=201)
        return Response(serializer.errors, status=400)
```

```python
# rest_framework/views.py
from django.views.decorators.csrf import csrf_exempt

class APIView(View):
    # …
    @classmethod
    def as_view(cls, **initkwargs):
        # …
        return csrf_exempt(view)
```

← 뷰가 csrf_exempt 장식자로 이미 감싸져있기에 POST 요청에서 csrf token 체크를 하지 않습니다.

rest_framework/views.py#L144

클래스 형태) APIView 구현 샘플 (detail/update/delete)

```python
from django.shortcuts import get_object_or_404
from rest_framework.response import Response
from rest_framework.views import APIView
from .models import Post
from .serializers import PostSerializer

class PostDetailAPIView(APIView):
    def get_object(self, pk):
        return get_object_or_404(Post, pk=pk)

    def get(self, request, pk, format=None):
        post = self.get_object(pk)
        serializer = PostSerializer(post)
        return Response(serializer.data)

    def put(self, request, pk):
        post = self.get_object(pk)
        serializer = PostSerializer(post, data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

    def delete(self, request, pk):
        post = self.get_object(pk)
        post.delete()
        return Response(status=status.HTTP_204_NO_CONTENT)
```

함수 형태) @api_view 장식자 구현 샘플 (list/create)

```python
from django.http import get_object_or_404
from rest_framework import status, Response
from rest_framework.decorators import api_view
from .models import Post
from .serializers import PostSerializer


@api_view(['GET', 'POST'])
def post_list(request):

    if request.method == 'GET':
        serializer = PostSerializer(Post.objects.all(), many=True)
        return Response(serializer.data)

    else:
        serializer = PostSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=201)
        return Response(serializer.errors, status=400)
```

하나의 작업 만을 구현코자 할 때
@api_view를 쓰시면 편리합니다.

함수 형태) @api_view 장식자 구현 샘플 (detail/update/delete)

```
from rest_framework.decorators import api_view


@api_view(['GET', 'PUT', 'DELETE'])
def post_detail(request, pk):
    post = get_object_or_404(Post, pk=pk)

    if request.method == 'GET':
        serializer = PostSerializer(post)
        return Response(serializer.data)

    elif request.method == 'PUT':
        serializer = PostSerializer(post, data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

    else:
        post.delete()

    return Response(status=status.HTTP_204_NO_CONTENT)
```

인생은 짧습니다.
파이썬/장고를 쓰세요.

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

-

Ask Company