

Ask Company



리액트와 함께 장고 시작하기 / 장고 DRF

JSON 응답뷰 만들기

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

장고 기본 구현과 DRF 구현 비교

한 Post 모델에 대한 API 서비스를 제공할 때 ...

이에 대한 URL을 설계한다면?

1. 새 포스팅 내용을 받아 등록하고, 확인 응답 → /post/new/ 주소로 POST 요청
2. 포스팅 목록 및 검색 응답 → /post/ 주소로 GET 요청
3. 10번 포스팅 내용 응답 → /post/10/ 주소로 GET 요청
4. 10번 포스팅 내용 갱신하고, 확인 응답 → /post/10/update/ 주소로 POST 요청
5. 10번 포스팅 내용 삭제하고, 확인 응답 → /post/10/delete/ 주소로 POST 요청

REST API 스타일로 재설계를 해본다면?

/post/ 주소

- GET 방식 요청 : 목록 응답
- POST 방식 요청 : 새 글 생성하고, 확인 응답
- ~~• (사용 X) PUT/PATCH 방식 요청~~
- ~~• (사용 X) DELETE 방식 요청~~

총 5개의 뷰 처리가 필요.

/post/10/ 주소

- GET 방식 요청 : 10번 글 내용 응답
- ~~• (사용 X) POST 방식 요청~~
- PUT 방식 요청 : 10번 글 수정/저장하고, 확인 응답
- DELETE 방식 요청 : 10번 글 삭제하고, 확인 응답

컨셉 코드 (1/3)

```
#  
# myapp/models.py  
#  
from django.db import models  
  
class Post(models.Model):  
    message = models.TextField()  
  
#  
# myapp/forms.py  
#  
from django import forms  
  
class PostForm(forms.ModelForm):  
    class Meta:  
        model = Post  
        fields = '__all__'
```

컨셉 코드 (2/3)

```
# myapp/views.py
```

```
def post_list(request):  
    if request.method == 'POST':  
        # 새 글 저장을 구현  
        form = PostForm(request.POST, request.FILES)  
        if form.is_valid():  
            post = form.save()  
            return JsonResponse(post)  
        return JsonResponse(form.errors)  
    else:  
        # 목록 응답을 구현  
        return JsonResponse(Post.objects.all())
```

컨셉 코드 (3/3)

```
def post_detail(request, pk):
    post = get_object_or_404(Post, pk=pk)

    if request.method == 'PUT':
        # 특정 글 갱신을 구현
        put_data = QueryDict(request.body)
        form = PostForm(put_data, instance=post)
        if form.is_valid():
            post = form.save()
            return JsonResponse(post)
        return JsonResponse(form.errors)

    elif request.method == 'DELETE':
        # 특정 글 삭제를 구현
        post.delete()
        return HttpResponse()

    else:
        # 특정 글 내용 응답을 구현
        return JsonResponse(post)
```

컨셉 코드 (3/3)

```
def post_detail(request, pk):
    post = get_object_or_404(Post, pk=pk)

    if request.method == 'PUT':
        # 특정 글 갱신을 구현
        put_data = QueryDict(request.body)
        form = PostForm(put_data, instance=post)
        if form.is_valid():
            post = form.save()
            return JsonResponse(post)
        return JsonResponse(form.errors)

    elif request.method == 'DELETE':
        # 특정 글 삭제를 구현
        post.delete()
        return HttpResponse()

    else:
        # 특정 글 내용 응답을 구현
        return JsonResponse(post)
```

다른 모델에 대해서 동일한 기능을 구현한다면,
Model/Form을 제외하고는 거의 정형화된 패턴.

➔ DRF는 이런 정형화된 중복을 줄일 수 있도록 도와주는
Class Based View를 비롯하여 다양한 기능을 지원

DRF 샘플 구현

Form/ModelForm → Serializer/ModelSerializer

FBV/CBV → FBV/CBV 그리고 ViewSet/ModelViewSet 등

```
#  
# myapp/models.py  
#  
from django.db import models  
  
class Post(models.Model):  
    message = models.TextField()
```

```
#  
# myapp/serializers.py  
#  
from rest_framework import serializers  
from .forms import Post
```

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

```
#  
# myapp/views.py  
#  
from rest_framework import viewsets  
  
class PostViewSet(viewsets.ModelViewSet):  
    queryset = Post.objects.all()  
    serializer_class = PostSerializer
```

```
#  
# myapp/urls.py  
#  
from rest_framework.routers import DefaultRouter  
from . import views
```

```
router = DefaultRouter()  
router.register('posts', views.PostViewSet)
```

```
urlpatterns = [  
    path('', include(router.urls)),  
]
```

다양한 HTTP 클라이언트

다양한 HTTP 클라이언트 프로그램

유저가 웹 브라우저를 통해 웹페이지 간 이동을 할 때

웹 프론트엔드에서 JavaScript를 통한 호출

Android/iOS 앱 코드를 통한 호출

웹 요청 개발 프로그램을 통한 호출

- GUI 프로그램 : Postman → Powerful API Client
- CLI 프로그램 : cURL, HTTPie
- 라이브러리 : requests

우리는 이 중에 HTTPie를 통해, 실습을 진행토록 하겠습니다.

HTTPIe를 통한 HTTP 요청 (1/2)

설치 : pip3 install httpie

HTTPIe 명령 예시

```
셸> http GET 요청할주소 GET인자명==값 GET인자명==값
셸> http --json POST 요청할주소 GET인자명==값 GET인자명==값 POST인자명=값 POST인자명=값
셸> http --form POST 요청할주소 GET인자명==값 GET인자명==값 POST인자명=값 POST인자명=값
셸> http PUT 요청할주소 GET인자명==값 GET인자명==값 PUT인자명=값 PUT인자명=값
셸> http DELETE 요청할주소 GET인자명==값 GET인자명==값
```

HTTPIe를 통한 HTTP 요청 (2/2)

2종류의 POST 요청 → 인코딩 방법의 차이

- --form 옵션 지정 : multipart/form-data
- --json 옵션을 지정하거나 생략 : application/json → 요청 데이터를 JSON 직렬화

httpbin.org 서비스를 통해, http 요청 연습을 해봅시다.

- 셸> http GET httpbin.org/get x==1 y==2
- 셸> http --form POST httpbin.org/post a=1 b=2 c=3
- 셸> http PUT httpbin.org/put hello=world
- 셸> http DELETE httpbin.org/delete

Django/DRF를 통한 API 구현 샘플

DRF 설치


공식문서를 참고해서 설치하는 습관이 중요 !!!

설치 : pip install **djangorestframework**~=3.11.0 (2020년 2월 현재 최신버전)

settings.INSTALLED_APPS에 "rest_framework" 추가

urlpatterns 다음 패턴 추가

```
urlpatterns = [  
    ...  
    path('api-auth/', include('rest_framework.urls'))  
]
```



```
from django.conf.urls import url  
from django.contrib.auth import views  
  
app_name = 'rest_framework'  
  
urlpatterns = [  
    url(r'^login/$', views.LoginView.as_view(template_name='rest_framework/login.html'), name='login'),  
    url(r'^logout/$', views.LogoutView.as_view(), name='logout'),  
]
```

단순히 django.contrib.auth의
login/logout 활용

<https://www.django-rest-framework.org/#installation>

DRF 주요 코드 샘플

```
# myapp/models.py
from django.db import models

class Post(models.Model):
    message = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```

```
# myapp/serializers.py
from rest_framework.serializers import ModelSerializer
from .models import Post

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

```
# myapp/views.py
from rest_framework.viewsets import ModelViewSet
from .serializers import PostSerializer
from .models import Post

class PostViewSet(ModelViewSet):
    queryset = Post.objects.all()
    serializer_class = PostSerializer

    def dispatch(self, request, *args, **kwargs):
        print("request.body:", request.body)
        print("request.POST:", request.POST)
        return super().dispatch(request, *args, **kwargs)
```

```
# myapp/urls.py
from django.urls import include, path
from rest_framework.routers import DefaultRouter
from . import views

router = DefaultRouter()
router.register('', views.PostViewSet)

urlpatterns = [
    path('', include(router.urls)),
]
```


요청 테스트

```
셸> http :8000
셸> http --form POST :8000 message="hello world"
셸> http :8000/1/
셸> http --form PUT :8000/1/ message="hello django"
셸> http :8000/1/
셸> http DELETE :8000/1/
셸> http :8000
```

```
# 목록 조회
# 새 포스팅 등록
# 1번 포스팅 조회
# 1번 포스팅 수정
# 1번 포스팅 수정
# 1번 포스팅 삭제
# 목록 조회
```

DRF를 통해 중복을 줄여보세요.

코드가 보다 간결하게 바뀌었음을 확인하실 수 있습니다.

DRF가 만능은 아닙니다.

DRF를 보다 잘 이해하고,
적절히 DRF의 기능을 **선택적으로 활용하는 능력**이 필요합니다.

인생은 짧습니다.
파이썬/장고를 쓰세요.

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

Ask Company