



리액트와 함께 장고 시작하기 / 장고 DRF

Form와 Serializer 관점에서 DRF 비교

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

Serializer / ModelSerializer

데이터 변환/직렬화 지원

QuerySet/Model객체 ⇔ Native Python 데이터타입, JSON/XML 등

Django의 Form/ModelForm과 유사

Serializer는 뷰 응답을 생성하는 데에 범용적이고 강력한 방법을 제공

ModelSerializer는 Serializer 생성을 위한 Shortcut

<https://www.django-rest-framework.org/api-guide/serializers/>

비교) Features

Form / ModelForm

HTML 입력폼을 통한 입력에 대한 유효성 검사

주로 Create/Update에 대한 처리에서 활용 → 장고 **admin** 에서 활용

CreateView/UpdateView CBV를 통한 뷰 처리 → 단일 뷰

Serializer / ModelSerializer

데이터 변환 및 직렬화 지원 (JSON 포맷 등)

주로 JSON 포맷 (주된 Web API 포맷) 입력에 대한 **유효성 검사**

List/**Create** 및 특정 Record에 대한 Retrieve/**Edit**/Delete 등 에서 활용

APIView를 통한 뷰 처리 → 단일 뷰

ViewSet을 통한 뷰 처리 → 2개 뷰 → 2개 URL 처리

비교) 주된 호출 주체

Form

일반적으로 웹브라우저 상에서

HTML Form Submit

JavaScript에 의한 비동기 호출

물론, Android/iOS 앱에 의한 요청/응답도 가능

모두 http(s) 프로토콜 요청/응답이기에.

Serializer

다양한 Client에 대한 Data 위주의 http(s) 요청

by : Web/Android/iOS 등

비교) 클래스 정의 - Form/ModelForm vs Serializer/ModelSerializer

```
from django import forms
```

```
class PostForm(forms.Form):  
    email = forms.EmailField()  
    content = forms.CharField(widget=forms.Textarea)  
    created_at = forms.DateTimeField()
```

```
class PostModelForm(forms.ModelForm):  
    class Meta:  
        model = Post  
        fields = '__all__'
```

```
from rest_framework import serializers
```

```
class PostSerializer(serializers.Serializer):  
    email = serializers.EmailField()  
    content = serializers.CharField(max_length=200)  
    created_at = serializers.DateTimeField()
```

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

비교) FBV를 통한 요청/응답 - Form vs Serializer

```
def post_list(request):
    if request.method == 'POST':
        form = PostForm(request.POST, request.FILES)
        if form.is_valid():
            post = form.save()
            return redirect(post)
    else:
        form = PostForm()

    return render(request, 'myapp/post_form.html', {
        'form': form,
    })

def post_list_or_create(request):
    # 새 글 저장을 구현
    if request.method == 'POST':
        form = PostForm(request.POST, request.FILES)
        if form.is_valid():
            post = form.save()
            # 커스텀 직렬화 루틴이 필요
            return JsonResponse(post)
        return JsonResponse(form.errors)

    # 목록 응답을 구현
    else:
        qs = Post.objects.all()
        return JsonResponse(qs) # 커스텀 직렬화 루틴이 필요
```

```
def post_list_or_create(request):
    if request.method == 'POST':
        serializer = PostSerializer(data=request.POST)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=201)
        return Response(serializer.errors, status=400)
    else:
        qs = Post.objects.all()
        serializer = PostSerializer(qs, many=True)
        return Response(serializer.data)

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

class PostListCreateAPIView(APIView):
    def get(self, request):
        serializer = PostSerializer(Post.objects.all(), 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)
```

비교) CBV를 통한 요청 및 응답 (1) - 장고 기본

```
from django.views.generic import ListView, CreateView
```

```
post_list = ListView.as_view(model=Post)
```

```
post_new = CreateView.as_view(model=Post, form_class=PostModelForm)
```

```
# 앱/urls.py
```

```
urlpatterns = [  
    path('', post_list),  
    path('new/', post_new),  
]
```

```
<form action="" method="post" enctype="multipart/form-data">  
    {% csrf_token %}  
    <table>  
        {{ form.as_table }}  
    </table>  
    <input type="submit" />  
</form>
```

localhost:8000/new/

Post Form

이 필드는 필수 항목입니다.
이 필드는 필수 항목입니다.

Title

Title

이 필드는 필수 항목입니다.
글 제목입니다. 100자 이내로 입력해주세요.

Content

Content

이 필드는 필수 항목입니다.
글 내용

Submit

비교) CBV를 통한 요청 및 응답 (2) - DRF의 APIView

```
from rest_framework.generics import ListCreateAPIView

class PostListCreateAPIView(ListCreateAPIView):
    queryset = Post.objects.all()
    serializer_class = PostModelSerializer

post_list_create = PostListCreateAPIView.as_view()

# 앱/urls.py
urlpatterns = [
    # 단일 URL에서 list/create 요청 처리
    path('api/post/', post_list_create),
]
```

```
→ http://localhost:8000/api/post/
HTTP/1.1 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Length: 4060
Content-Type: application/json
Date: Sat, 05 Jan 2019 05:24:24 GMT
Server: WSGIServer/0.2 CPython/3.7.1
Vary: Accept, Cookie
X-Frame-Options: SAMEORIGIN

[
  {
    "content": "Hello World",
    "created_at": "2018-12-24T03:00:25.595352Z",
    "id": 1,
    "title": "abc",
    "updated_at": "2019-01-02T03:30:12.268944Z",
    "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_1) AppleWebKit/537.36
78.98 Safari/537.36"
  },
  {
    "content": "world",
    "created_at": "2018-12-27T03:09:21.899390Z",
    "id": 2,
    "title": "hello",
  }
]
```


비교) CBV를 통한 요청 및 응답 (3)

```
→ http POST http://localhost:8000/api/post/
HTTP/1.1 400 Bad Request
Allow: GET, POST, HEAD, OPTIONS
Content-Length: 158
Content-Type: application/json
Date: Sat, 05 Jan 2019 16:36:41 GMT
Server: WSGIServer/0.2 CPython/3.7.1
Vary: Accept, Cookie
X-Frame-Options: SAMEORIGIN

{
  "content": [
    "이 필드는 필수 항목입니다."
  ],
  "title": [
    "이 필드는 필수 항목입니다."
  ],
  "user_agent": [
    "이 필드는 필수 항목입니다."
  ]
}
```

```
→ http POST http://localhost:8000/api/post/ title="hello" content="world" user_agent="httpie"
HTTP/1.1 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Length: 151
Content-Type: application/json
Date: Sat, 05 Jan 2019 16:38:10 GMT
Server: WSGIServer/0.2 CPython/3.7.1
Vary: Accept, Cookie
X-Frame-Options: SAMEORIGIN

{
  "content": "world",
  "created_at": "2019-01-05T16:38:10.785392Z",
  "id": 16,
  "title": "hello",
  "updated_at": "2019-01-05T16:38:10.786335Z",
  "user_agent": "httpie"
}
```

참고) user_agent 필드는 요청 Header의 User-Agent 헤더를 활용하는 것이 좋습니다.
→ request.META.get("HTTP_USER_AGENT")

유효성 검사 수행 시점 - Form vs Serializer

```
class ProcessFormView(View):
    def get(self, request, *args, **kwargs):
        return self.render_to_response(self.get_context_data())

    def post(self, request, *args, **kwargs):
        form = self.get_form()          # POST 데이터를 통해 Form 객체 생성
        if form.is_valid():             # 유효성 검사를 수행. 실패하면 False를 반환
            return self.form_valid(form) # DB로의 저장을 수행
        else:
            return self.form_invalid(form) # 오류 HTML 응답

# rest_framework/mixins.py
class CreateModelMixin(object):
    def create(self, request, *args, **kwargs):
        serializer = self.get_serializer(data=request.data) # POST 데이터를 통해 Serializer 인스턴스를 만들고
        serializer.is_valid(raise_exception=True)             # 유효성 검사를 수행. 실패하면 예외발생 !!!
        self.perform_create(serializer)                       # DB로의 저장을 수행
        headers = self.get_success_headers(serializer.data)   # 필요한 헤더를 뽑고
        return Response(serializer.data, status=status.HTTP_201_CREATED, headers=headers) # 응답을 합니다.

    def perform_create(self, serializer): # CREATE 커스텀은 이 함수를 재정의하세요.
        serializer.save()
```

커스텀 유효성 검사 루틴 - clean_* vs validate_*

```
from django import forms
```

```
class PostForm(forms.Form):  
    title = forms.CharField()
```

```
    def clean_title(self):  
        value = self.cleaned_data.get('title', '')  
        if 'django' not in value:  
            raise forms.ValidationError('제목에 필히 django가 포함되어야 합니다.')  
        return value
```

```
from rest_framework.exceptions import ValidationError
```

```
class PostSerializer(serializers.Serializer):  
    title = serializers.CharField(max_length=100)
```

```
    def validate_title(self, value):  
        if 'django' not in value:  
            raise ValidationError('제목에 필히 django가 포함되어야 합니다.')  
        return value
```

Life is short.
You need Python and Django.

I will be your pacemaker.

