# Django advanced project #week_2

Writer : Taeyeong Kim, DGIST LikeLion

Date: 2019.10.09

*아래 자료를 공부하기 전에, https://github.com/lizard-kim/LikeLion_study_Fall_Semester 에 있는 코드를 clone하여 참고하기 바란다. 코드에 대한 자세한 정보는 주석을 통해 알 수 있으니, 이곳에서는 대략적인 설명만 할 것이다.*

# Restful API in Django

## REST & REST API

### REST 란?

"Representational State Transfer" 의 약자로, HTTP를 이용해 통신하는 네트워크상에서 정한 약속. 분산 하이퍼미디어 시스템을 위한 소프트웨어 설계형식.

쉽게 말해서 자원을 이름으로 구분하여 상태를 전송하는 방법이다.

> 여기서 말하는 자원과 이름이란,
>
> 자원(resource)의 표현(representation)
>
> 자원: 해당 소프트웨어가 관리하는 모든 것
> -> Ex) 문서, 그림, 데이터, 해당 소프트웨어 자체 등
> 자원의 이름: 그 자원을 표현하기 위한 이름
> -> Ex) DB의 학생 정보가 자원일 때, 'students'를 자원의 이름으로 정한다.

### REST의 장점

Web의 독립적인 운용과 발전의 측면에서 기존의 약속, 하위호환을 깨트리지 않고 네트워크의 독립적인 운용/발전이 가능하다는 장점이 있다. 즉, 패치내용을 일일이 다 고지하고 적용하지 않아도 된다는 것이다.

> 그밖의 장점은 다음과 같다
>
> - HTTP 프로토콜의 인프라를 그대로 사용하므로 REST API 사용을 위한 별도의 인프라를 구출할 필요가 없다.
> - HTTP 프로토콜의 표준을 최대한 활용하여 여러 추가적인 장점을 함께 가져갈 수 있게 해준다.
> - HTTP 표준 프로토콜에 따르는 모든 플랫폼에서 사용이 가능하다.
> - Hypermedia API의 기본을 충실히 지키면서 범용성을 보장한다.
> - REST API 메시지가 의도하는 바를 명확하게 나타내므로 의도하는 바를 쉽게 파악할 수 있다.
> - 여러가지 서비스 디자인에서 생길 수 있는 문제를 최소화한다.
> - 서버와 클라이언트의 역할을 명확하게 분리한다.

### REST가 되기 위한 필요충분조건

1. Server-Client(서버-클라이언트 구조)
    - 자원이 있는 쪽이 Server, 자원을 요청하는 쪽이 Client가 된다.

        - REST Server: API를 제공하고 비즈니스 로직 처리 및 저장을 책임진다.
        - Client: 사용자 인증이나 context(세션, 로그인 정보) 등을 직접 관리하고 책임진다.
    - 서로 간 의존성이 줄어든다.

2. Stateless(무상태)
    - HTTP 프로토콜은 Stateless Protocol이므로 REST 역시 무상태성을 갖는다.
    - Client의 context를 Server에 저장하지 않는다.
        - 즉, 세션과 쿠키와 같은 context 정보를 신경쓰지 않아도 되므로 구현이 단순해진다.
    - Server는 각각의 요청을 완전히 별개의 것으로 인식하고 처리한다.
        - 각 API 서버는 Client의 요청만을 단순 처리한다.
        - 즉, 이전 요청이 다음 요청의 처리에 연관되어서는 안된다.
        - 물론 이전 요청이 DB를 수정하여 DB에 의해 바뀌는 것은 허용한다.
        - Server의 처리 방식에 일관성을 부여하고 부담이 줄어들며, 서비스의 자유도가 높아진다.
3. Cacheable(캐시 처리 가능)
    - 웹 표준 HTTP 프로토콜을 그대로 사용하므로 웹에서 사용하는 기존의 인프라를 그대로 활용할 수 있다.
        - 즉, HTTP가 가진 가장 강력한 특징 중 하나인 캐싱 기능을 적용할 수 있다.
        - HTTP 프로토콜 표준에서 사용하는 Last-Modified 태그나 E-Tag를 이용하면 캐싱 구현이 가능하다.
    - 대량의 요청을 효율적으로 처리하기 위해 캐시가 요구된다.
    - 캐시 사용을 통해 응답시간이 빨라지고 REST Server 트랜잭션이 발생하지 않기 때문에 전체 응답시간, 성능, 서버의 자원 이용률을 향상시킬 수 있다.
4. Layered System(계층화)
    - Client는 REST API Server만 호출한다.
    - REST Server는 다중 계층으로 구성될 수 있다.
        - API Server는 순수 비즈니스 로직을 수행하고 그 앞단에 보안, 로드밸런싱, 암호화, 사용자 인증 등을 추가하여 구조상의 유연성을 줄 수 있다.
        - 또한 로드밸런싱, 공유 캐시 등을 통해 확장성과 보안성을 향상시킬 수 있다.
    - PROXY, 게이트웨이 같은 네트워크 기반의 중간 매체를 사용할 수 있다.
5. Code-On-Demand(optional)
    - Server로부터 스크립트를 받아서 Client에서 실행한다.
    - 반드시 충족할 필요는 없다.
6. Uniform Interface(인터페이스 일관성)
    - URI로 지정한 Resource에 대한 조작을 통일되고 한정적인 인터페이스로 수행한다.
    - HTTP 표준 프로토콜에 따르는 모든 플랫폼에서 사용이 가능하다.
        - 특정 언어나 기술에 종속되지 않는다.

> 하지만, 이 모든 조건을 다 지키지는 않는다.

**API란?**

Client와 Server간의 정보를 교환가능하도록 하는 것.

> 김밥천국의 주문표를 생각하면 편하다.

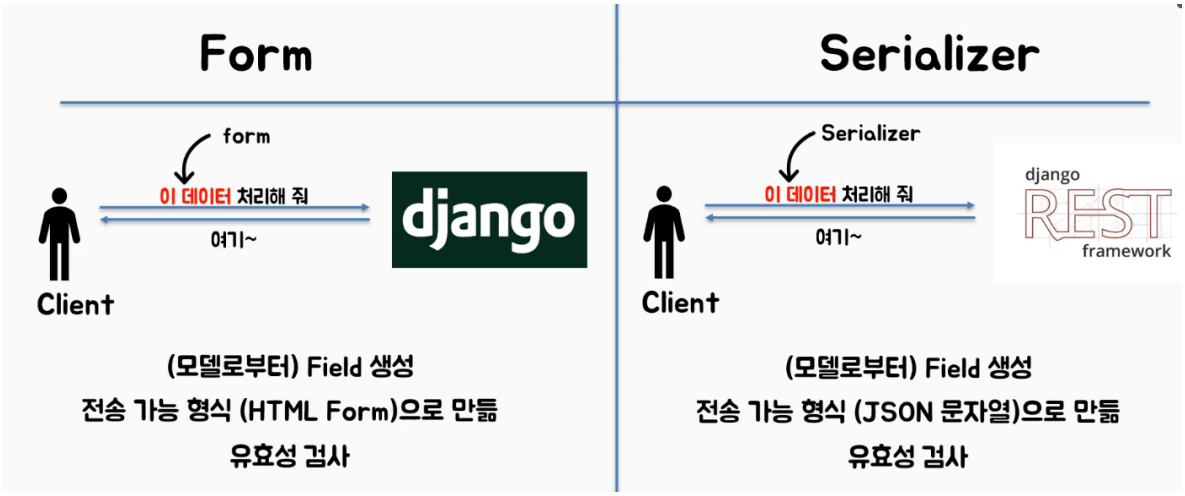
**그렇다면, REST API란?**

쉽게 말해, REST기반으로 서비스 API를 구축한 것으로, HTTP(GET, POST 등)로 CRUD를 구현할 수 있는 API를 말한다.

---

## REST API를 만들어보자

1주차 수업에서 JSON을 **serialization**하여 client가 server에게 보낸다고 했다. 이러한 과정은 form과 아주 유사하다.




> 그림을 보면 한눈에 이해할 수 있다.

프로젝트를 생성하기 이전에, django_restframework를 설치하자

```
$ pip install djangorestframeowork
```

이때, 프로젝트의 settings.py에 'rest_framework' 라는 앱을 우리가 생성한 앱과 같이 등록해주자!

프로젝트를 생성하자.

```
$ django-admin startproject firstrest
$ python manage.py startapp post # 이후 settings.py에 app을 추가해주자 /
rest_framework도 잊지않고 등록하기!
```

### models.py

다음과 같이 model.py에 model을 정의해주자.

```
from django.db import models

class Post(models.Model):
    title = models.CharField(max_length = 100)
    body = models.TextField()
```

> migrate 하는 것 잊지않기!

**firstrest/url.py**

다음과 같이 url을 정의해주자. include로 app의 url과 분리하여 정의하자.

```
from django.contrib import admin
from django.urls import path, include
import post.urls

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include(post.urls)),
]
```

**serializer.py**

app에서 url을 정의하기 전에, app 폴더 안에 `serializer.py` 파일을 생성하자.

```
from .models import Post
from rest_framework import serializers # use rest_framework

class PostSerializer(serializers.ModelSerializer):
    class Meta:
        model = Post
        fields = '__all__' # make all field serializer
        # fields = ['title', 'body'] make title and body fields serializer
        read_only_fields = ('title', ) # tuple recommended / title을 read_only로
만들수 있다.
```

**views.py**

```
from rest_framework import viewsets # what is viewsets?
from .models import Post
from .serializer import PostSerializer

#CBV

class PostViewSet(viewsets.ModelViewSet):
    queryset = Post.objects.all() # Post model의 모든 것을 가져다 queryset으로 쓰겠다.
    serializer_class = PostSerializer # PostSerializer를 사용하겠다.
```

viewsets은 View를 설계하는 쉽고 간단한 방법이다. 자세한 내용은 뒤에서 알아보자.

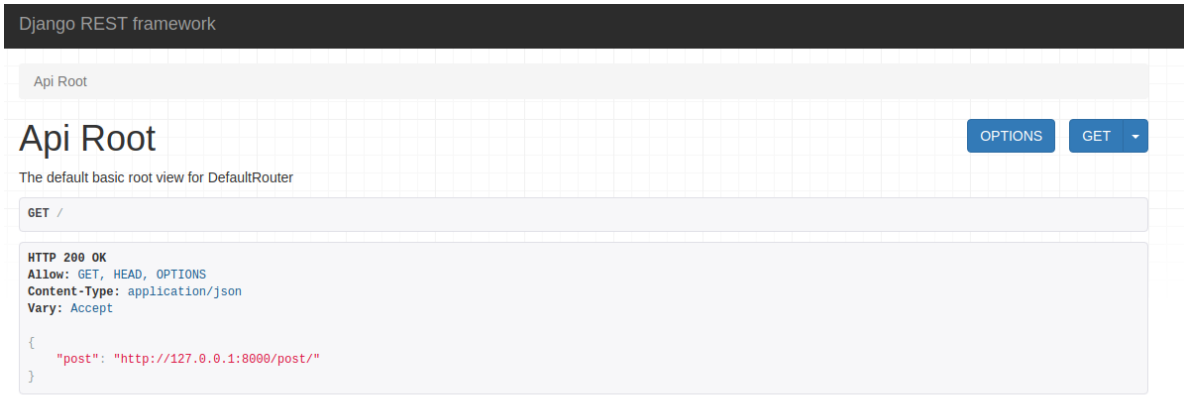
**post/urls.py**

```
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from . import views
#django rest framework -> router -> url

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

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

router라는 새로운 친구가 보인다. 우리는 router를 통해서 rest_framework의 url을 정의할 것이다.

**Run Server!**

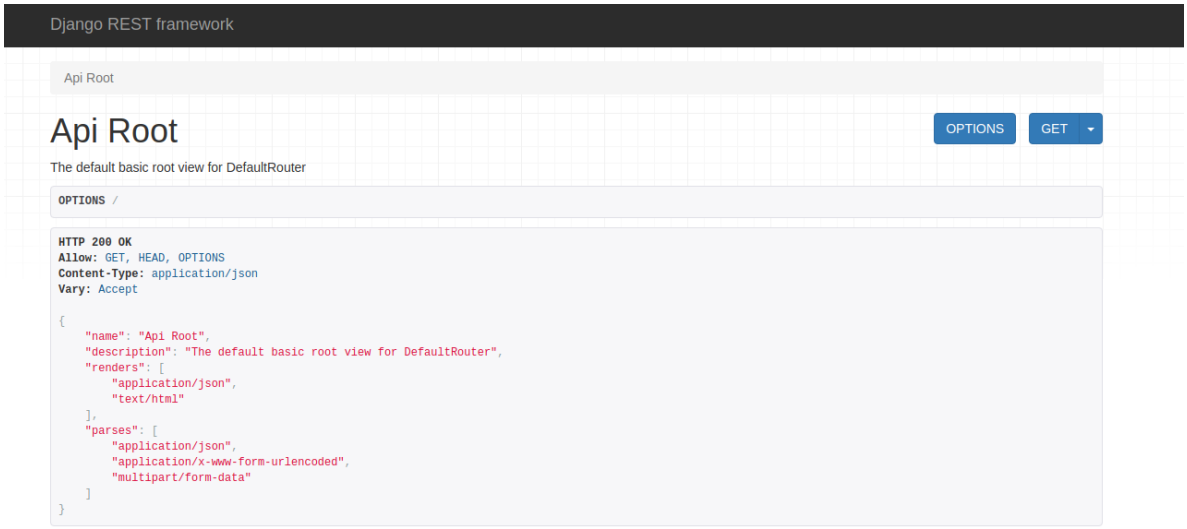
```
$ python manage.py runserver
```




> 잘 돌아가는 것을 확인할 수 있다.

현재 위치에서 저번시간에 배운 httpie를 사용하여 get을 하게되면, 얻게되는 정보들이 표시되어 있다. (직접 한번 해보자.)

이때 오른쪽 상단의 option을 누르게 되면 다음과 같은 창이 생긴다.




이 또한 `http OPTIONS http://127.0.1:8000/` 을 통해 얻을 수 있는 정보가 표시되어 있다.

마찬가지로 /post에 접속하여 글을 직접 post해보고, httpie로 post와 delete 해보자.

```
$ http POST http://127.0.1:8000/post/ title="test title" body="this is test"
$ http DELETE http://127.0.1:8000/post/1/ #번호는 pk값
```

# References

likelion 강의 소스코드 : https://github.com/kangtegong/django-RESTful-API

REST관련 개념 : https://gmlwjd9405.github.io/2018/09/21/rest-and-restful.html