

# Ask Django

여러분의 파이썬/장고 페이스메이커가 되겠습니다.  
**EP 12. PostAPIView** 차근차근 응답시간 줄여보기

# 관련 문서

원문 : [Web API performance: profiling Django REST framework](#)

장고 공식문서

- [Performance and optimization](#)
- [Database access optimization](#)

# 왜 최적화를 해야하나요?

- 보다 빠르게 동작하는 프로그램을 위해 !!!
  - 보다 낮은 CPU 타임
  - 보다 낮은 메모리 소모
- 개발비용이 가장 큰 리소스입니다. => 개발시간 + 인건비
- 최적화를 통해 성능은 높아지지만, 유지관리성이 낮아질 수도 있습니다. => 가성비를 체크해보세요.
- 한 영역의 개선이 다른 부분을 희생시킬 수 있습니다.
  - ex) CPU 연산을 아꼈는데, 메모리 소모가 늘었다.

# 시작하기에 앞서

Throttle와 Pagination은 꺼주시고,  
현재, DB Record갯수가 적기에 ... **뱅튀기**

```
from ep08.models import Post
```

```
post = Post.objects.first()
```

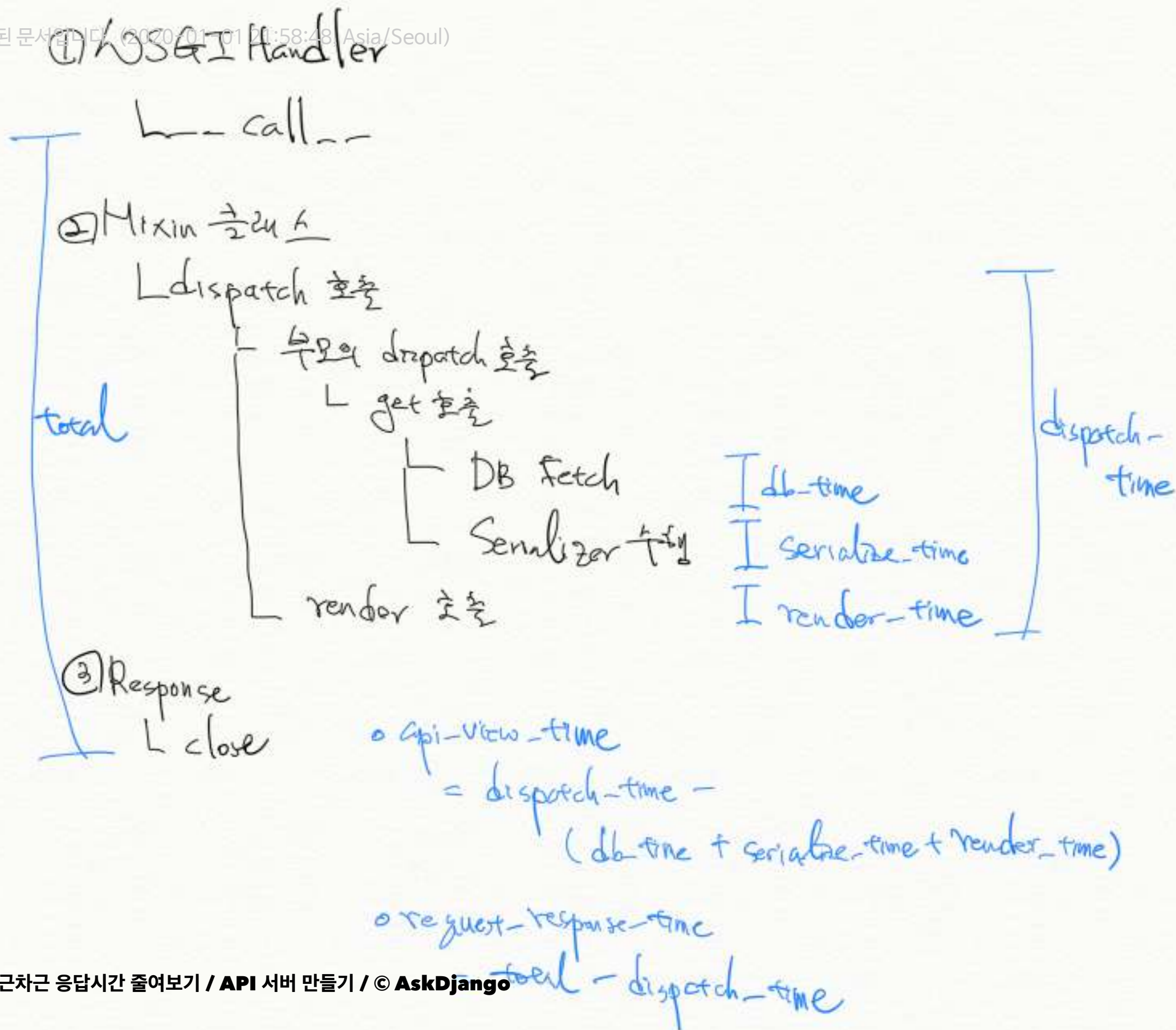
```
for i in range(100):  
    post.id = None      # Django Model은 id=None일 경우, CREATE를 수행합니다.  
    post.save()
```

주의: 실제 서비스에서는 id=None과 같은 코드는 절대 쓰지 마세요.

# 측정 **Metric**

다음 코드를 통해, 요청 처리시의 각 부분에 대한 시간을 측정해보겠습니다.

- Database Lookup (db\_time) : 데이터베이스 Fetch 수행시간
- Serialization (serializer\_time) : Serializer 직렬화 수행시간
- API View (api\_view\_time) : APIView 수행시간
- Response rendering (render\_time) : Response 렌더링 수행시간
- Django request/response : request/response 수행시간



# 수행시간 비교 (단위 : 초)

구분	DB Lookup/ 캐싱	Serializer	APIView	Response	Req/Res	Total	줄여진 시간 비율
기본	0.119091	0.680896	0.000511	0.000243	0.003622	0.804363	
Serializer 제거, 추가	0.000190	0.000000	0.000508	0.136948	0.003667	0.141313	-82.4%
캐싱, 추가	0.006942	0.000000	0.000375	0.000227	0.001927	0.009471	-98.8%
APIView 설정 최소화, 추가	0.008560	0.000000	0.000609	0.000229	0.001850	0.011248	-98.6%
미들웨어 제거, 추가	0.000335	0.000000	0.000172	0.000197	0.000499	0.001203	-99.85%
HttpResponse, 추가	0.000192	0.000000	0.000286	0.000000	0.000512	0.000990	-99.87%

# 측정을 위한 코드 #gist

## 코드 1/3

```
import time
from rest_framework.response import Response

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

    def dispatch(self, request, *args, **kwargs):
        global cbv
        cbv = self

        dispatch_start = time.time()
        response = super().dispatch(request, *args, **kwargs)

        render_start = time.time()
        response.render()
        self.render_time = time.time() - render_start

        self.dispatch_time = time.time() - dispatch_start
        self.api_view_time = self.dispatch_time - (self.render_time + self.serializer_time + self.db_time)

        return response
```



## 코드 2/3

```
def list(self, request, *args, **kwargs):  
    db_start = time.time()  
    post_list = list(self.queryset)  
    self.db_time = time.time() - db_start  
  
    serializer_start = time.time()  
    serializer = self.get_serializer(self.queryset, many=True)  
    data = serializer.data  
    self.serializer_time = time.time() - serializer_start  
  
    return Response(data)
```

## 코드 3/3

주의 : 아래코드는 Single Request에서만 동작합니다. 동시 요청에서는 제대로 동작하지 않습니다.

```
from django.core.signals import request_started, request_finished

def started_fn(sender, **kwargs):
    global started
    started = time.time()

def finished_fn(sender, **kwargs):
    request_response_time = (time.time() - started) - cbv.dispatch_time

    total = cbv.db_time + cbv.serializer_time + cbv.api_view_time + cbv.render_time + request_response_time

    print('Database Lookup      - db_time          : {:.6f}s, {:>4.1f}%'.format(cbv.db_time, 100*(cbv.db_time/total)))
    print('Serialization      - serializer_time : {:.6f}s, {:>4.1f}%'.format(cbv.serializer_time, 100*(cbv.serializer_time/total)))
    print('API View              - api_view_time   : {:.6f}s, {:>4.1f}%'.format(cbv.api_view_time, 100*(cbv.api_view_time/total)))
    print('Response rendering - render_time      : {:.6f}s, {:>4.1f}%'.format(cbv.render_time, 100*(cbv.render_time/total)))
    print('Django request/response : {:.6f}s, {:>4.1f}%'.format(request_response_time, 100*(request_response_time/total)))

request_started.connect(started_fn)    # 요청 처리 시작
request_finished.connect(finished_fn) # 요청 처리 끝
```

## 기본 코드, 수행

```
"GET /ep08/post/?format=json HTTP/1.1" 200 6497
```

Total	:	0.804363s	
Database Lookup	- db_time	: 0.119091s,	14.8%
Serialization	- serializer_time	: 0.680896s,	84.7%
API View	- api_view_time	: 0.000511s,	0.1%
Response rendering	- render_time	: 0.000243s,	0.0%
Django request/response		: 0.003622s,	0.5%

=> DB Lookup 및 DRF Serializer의 시간비율이 가장 높네요.

# Serializer를 제거해봅시다.

**QuerySet.values(필드명)**를 통해, 원하는 필드만 가져오기

```
def list(self, request, *args, **kwargs):
    db_start = time.time()
    # post_list = list(self.queryset)
    data = self.queryset.values('author__username', 'message')
    self.db_time = time.time() - db_start

    # serializer_start = time.time()
    # serializer = self.get_serializer(self.queryset, many=True)
    # data = serializer.data
    # self.serializer_time = time.time() - serializer_start
    self.serializer_time = 0

    return Response(data)
```

## 수행결과

"GET /ep08/post/?format=json HTTP/1.1" 200 6598

Total	:	0.141313s	
Database Lookup	- db_time	: 0.000190s,	0.1%
Serialization	- serializer_time	: 0.000000s,	0.0%
API View	- api_view_time	: 0.000508s,	0.4%
Response rendering	- render_time	: 0.136948s,	96.9%
Django request/response	:	0.003667s,	2.6%

=> 해당 부분이 0.8초에서 0.00019초로 줄었네요.<sup>1</sup>

---

<sup>1</sup> 하지만, Serializer으로 얻는 막대한 개발생산성이 있으며, Serializer를 쓰더라도 적절한 캐싱을 통해 극복할 수 있습니다.

# DB/Serializer 대신에 캐싱

데이터가 변경되지 않는다면, 캐싱을 통해 성능을 높일 수 있습니다.

```
from django.core.cache import cache

# 중략

def list(self, request, *args, **kwargs):
    db_start = time.time()

    data = cache.get('post_list_data')
    if data is None:
        data = self.queryset.values('author__username', 'message')
        cache.set('post_list_data', data, 60)

    self.db_time = time.time() - db_start

    self.serializer_time = 0

    return Response(data)
```

# 수행결과

"GET /ep08/post/?format=json HTTP/1.1"	200	6598	# 캐싱 전
Total	:	0.152553s	
Database Lookup - db_time	:	0.146078s, 95.8%	
Serialization - serializer_time	:	0.000000s, 0.0%	
API View - api_view_time	:	0.000642s, 0.4%	
Response rendering - render_time	:	0.000283s, 0.2%	
Django request/response	:	0.005550s, 3.6%	
"GET /ep08/post/?format=json HTTP/1.1"	200	6598	# 캐싱된 값 활용
Total	:	0.009471s	
Database Lookup - db_time	:	0.006942s, 73.3%	
Serialization - serializer_time	:	0.000000s, 0.0%	
API View - api_view_time	:	0.000375s, 4.0%	
Response rendering - render_time	:	0.000227s, 2.4%	
Django request/response	:	0.001927s, 20.3%	

Tip: 코드 최적화가 먼저입니다. 캐시는 적당히. 무분별한 캐시는 마약과도 같습니다.

# APIView에 필요한 설정만 넣기

```
from rest_framework.negotiation import BaseContentNegotiation
from rest_framework.renderers import JSONRenderer

class IgnoreClientContentNegotiation(BaseContentNegotiation):
    def select_parser(self, request, parsers):
        "Select the first parser in the `parser_classes` list."
        return parsers[0]

    def select_renderer(self, request, renderers, format_suffix):
        "Select the first renderer in the `renderer_classes` list."
        return (renderers[0], renderers[0].media_type)

class PostViewSet(ModelViewSet):
    queryset = Post.objects.all()
    serializer_class = PostSerializer
    permission_classes = []
    authentication_classes = []
    renderer_classes = [JSONRenderer]
    content_negotiation_class = IgnoreClientContentNegotiation
```



# 수행결과

"GET /ep08/post/?format=json HTTP/1.1"	200 6598	# 캐싱 전
Total	: 0.143334s	
Database Lookup - db_time	: 0.138824s, 96.9%	
Serialization - serializer_time	: 0.000000s, 0.0%	
API View - api_view_time	: 0.000358s, 0.2%	
Response rendering - render_time	: 0.000309s, 0.2%	
Django request/response	: 0.003843s, 2.7%	
"GET /ep08/post/?format=json HTTP/1.1"	200 6598	# 캐싱된 값 활용
Total	: 0.011248s	
Database Lookup - db_time	: 0.008560s, 76.1%	
Serialization - serializer_time	: 0.000000s, 0.0%	
API View - api_view_time	: 0.000609s, 5.4%	
Response rendering - render_time	: 0.000229s, 2.0%	
Django request/response	: 0.001850s, 16.4%	

# 미들웨어 제거하기

본 프로젝트가 **API** 기능만 할 경우, 장고 웹페이지를 위한 기능을 꺼볼 수 있습니다.

```
# 프로젝트/settings.py
```

```
INSTALLED_APPS = [  
    # 'debug_toolbar',  
]
```

```
MIDDLEWARE = [  
    # 'debug_toolbar.middleware.DebugToolbarMiddleware',  
    # 'django.middleware.security.SecurityMiddleware',  
    # 'django.contrib.sessions.middleware.SessionMiddleware',  
    # 'django.middleware.common.CommonMiddleware',  
    # 'django.middleware.csrf.CsrfViewMiddleware',  
    # 'django.contrib.auth.middleware.AuthenticationMiddleware',  
    # 'django.contrib.messages.middleware.MessageMiddleware',  
    # 'django.middleware.clickjacking.XFrameOptionsMiddleware',  
]
```

# 수행결과

"GET /ep08/post/?format=json HTTP/1.1"	200	6598	# 캐싱 전
Total	:	0.004360s	
Database Lookup - db_time	:	0.003030s, 69.5%	
Serialization - serializer_time	:	0.000000s, 0.0%	
API View - api_view_time	:	0.000300s, 6.9%	
Response rendering - render_time	:	0.000224s, 5.1%	
Django request/response	:	0.000806s, 18.5%	
"GET /ep08/post/?format=json HTTP/1.1"	200	6598	# 캐싱된 값 활용
Total	:	0.001203s	
Database Lookup - db_time	:	0.000335s, 27.8%	
Serialization - serializer_time	:	0.000000s, 0.0%	
API View - api_view_time	:	0.000172s, 14.3%	
Response rendering - render_time	:	0.000197s, 16.4%	
Django request/response	:	0.000499s, 41.5%	

# Django 기본 HttpResponseRedirect 쓰기

```
import json
from django.http import HttpResponseRedirect

# 중략

def dispatch(self, request, *args, **kwargs):
    global cbv
    cbv = self

    dispatch_start = time.time()
    response = super().dispatch(request, *args, **kwargs)

    render_start = time.time()
    # response.render() # 필히, 주석처리 !!!
    self.render_time = time.time() - render_start

    self.dispatch_time = time.time() - dispatch_start
    self.api_view_time = self.dispatch_time - (self.render_time + self.serializer_time + self.db_time)

    return response

def list(self, request, *args, **kwargs):
    db_start = time.time()

    data = cache.get('post_list_data')
    if data is None:
        data = list(self.queryset.values('author__username', 'message')) # QuerySet은 JSON 직렬화 불가
        cache.set('post_list_data', data, 60)

    self.db_time = time.time() - db_start

    self.serializer_time = 0


    return HttpResponseRedirect(json.dumps(data), content_type='application/json; charset=utf-8')
```

# 수행결과

"GET /ep08/post/?format=json HTTP/1.1"	200	8546	# 캐싱 전
Total	:	0.004152s	
Database Lookup - db_time	:	0.002985s, 71.9%	
Serialization - serializer_time	:	0.000000s, 0.0%	
API View - api_view_time	:	0.000406s, 9.8%	
Response rendering - render_time	:	0.000000s, 0.0%	
Django request/response	:	0.000761s, 18.3%	
"GET /ep08/post/?format=json HTTP/1.1"	200	8546	# 캐싱된 값 활용
Total	:	0.000990s	
Database Lookup - db_time	:	0.000192s, 19.4%	
Serialization - serializer_time	:	0.000000s, 0.0%	
API View - api_view_time	:	0.000286s, 28.9%	
Response rendering - render_time	:	0.000000s, 0.0%	
Django request/response	:	0.000512s, 51.7%	

# 정리

- 필요에 따라 Serializer/Response를 쓰지 않고 직접 처리
  - 생산성을 통한 성능의 희생
- 조회 요청의 경우, 적절한 캐싱은 성능을 높여줍니다.
  - 주의) 로직으로 캐싱된 내용을 적절히 만료시켜야만 합니다.
- 어떤 설정이 적용되는 지 정확히 알고, 필요한 설정만 적용하기.
  - APIView 설정 최소화
  - 미들웨어 제거



*Life is short,  
use Python3/Django.*