
WAFFLE
STUDIO

와플스튜디오 Backend Seminar

Instructor:

강지혁 @Jhvictor4

2021.09.25.(토) 10:00 - Zoom

세미나 3차시

TA: 유진님, 원식님, 도현님

Contributor 변다빈 @bdv111

서울대학교 앱/웹개발 동아리 와플스튜디오

Assignment 1 Review

attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

채점이 생각보다 너무 빡세다..

Most Fail-Point

```
    def get_user(self, survey):
        if survey.user:
            return UserSerializer(survey.user, context=self.context).data
        return None
```

```
>>> from survey.serializers import UserSerializer
>>> UserSerializer(None).data
{'username': '', 'email': '', 'password': '', 'last_login': None, 'date_joined': None, 'first_name': '', 'last_name': ''}
>>> None
>>>
```

WA LE
STUDIO

Assignment 1 Review

attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

Use Serializers!!

View는 깔끔하게

```python
def create(self, request):
    # copy makes request.data mutable
    data = request.data.copy()
    data.update(os_name=data.get('os'))

    serializer = self.get_serializer(data=data)
    serializer.is_valid(raise_exception=True)
    serializer.save()
    return Response(serializer.data, status=status.HTTP_201_CREATED)
```

Serializer를 통해서 Service Layer 구성해본다는 생각

(c.f. Reference)

WA LE
STUDIO

Assignment 2 Review

attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

화이팅

과제 0, 1까지 채점한 결과

-> 현재까지는 모두들 잘 해주고 계십니다. Keep Going!!

과제 2는 다소 어려우셨을 것으로 생각합니다.

앞으로도..

난이도는 비슷하거나 점점 더 어려워질 것 같습니다 😂

(코드는 과제 due가 지나면 세미나 자료에 항상 최신화 되어있으니 확인하시면 좋아요!)

WA#LE
STUDIO

Assignment 2 Review

attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

What you did so far

복습

- JWT 로그인
- DB Relation 설계
- Serializer + View 통해 서비스 레이어 구현하기

++ 디버깅

(코드는 과제 due가 지나면 세미나 자료에 항상 최신화 되어있으니 확인하시면 좋아요!)

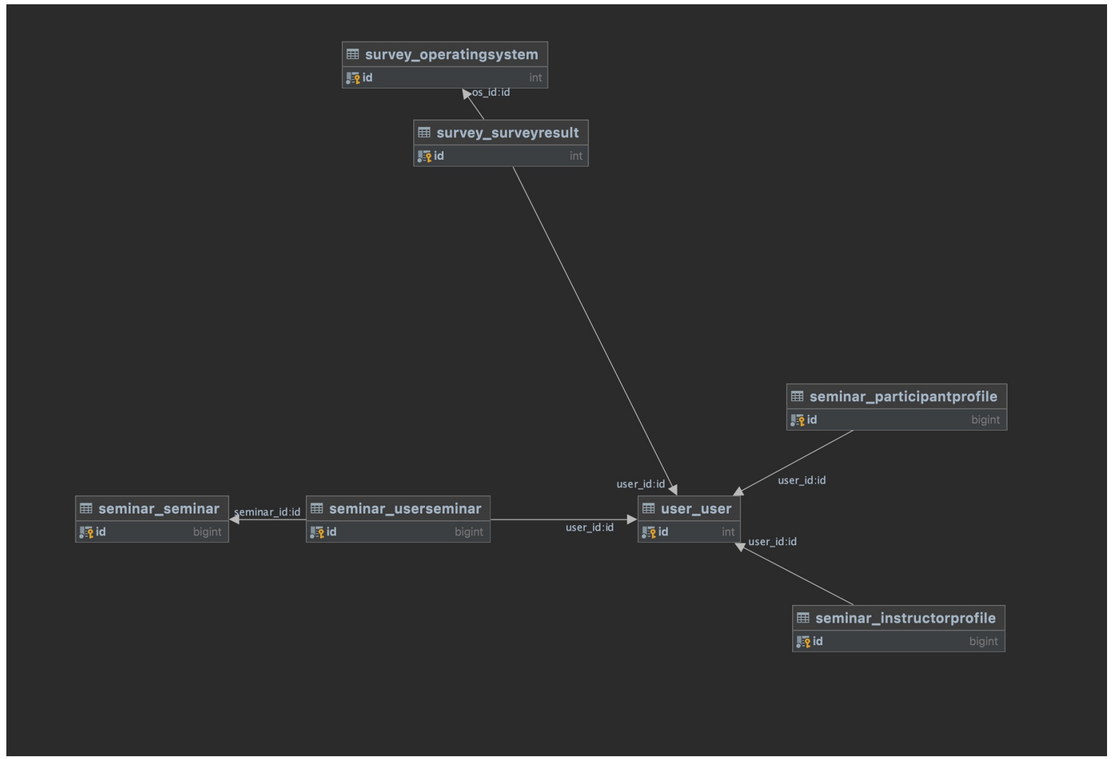
WA LE
STUDIO

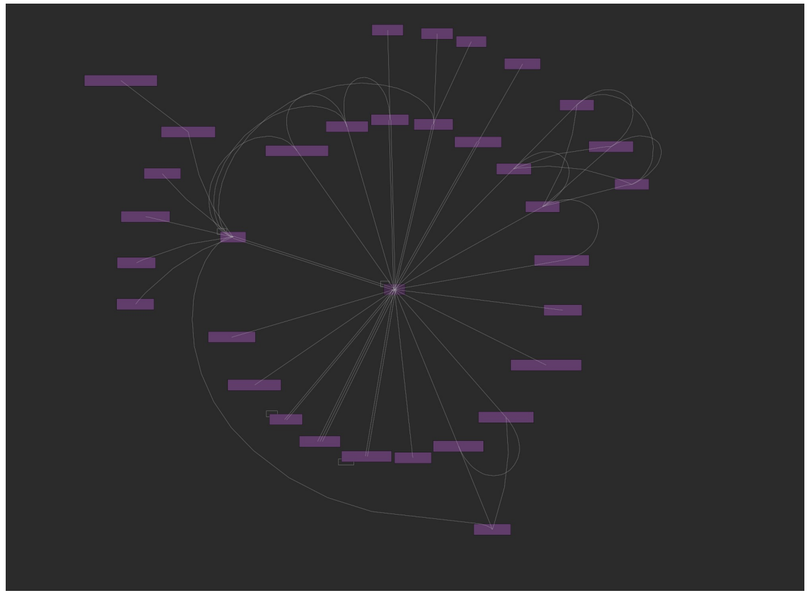
DB Relation

attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

설계를 신중히, 또 잘 생각해보기

- 모델 설계, 필드 설정에 정답은 없습니다. (cascade 설정, 필드 관리 등등 …)
- 이걸 잘 고민하는 건 서버 개발의 핵심 !!





WA LE
STUDIO

Serializer + Views

attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

Serializer + View 통해 서비스 레이어 구현하기

```python
def post(self, request, seminar_id=None):

    role, user = request.data.get('role'), request.user
    service = RegisterSeminarService(data={'role': role},
                                     context={'user': user, 'seminar': seminar_id})
    status, data = service.execute()

    return Response(status=status, data=data)
```

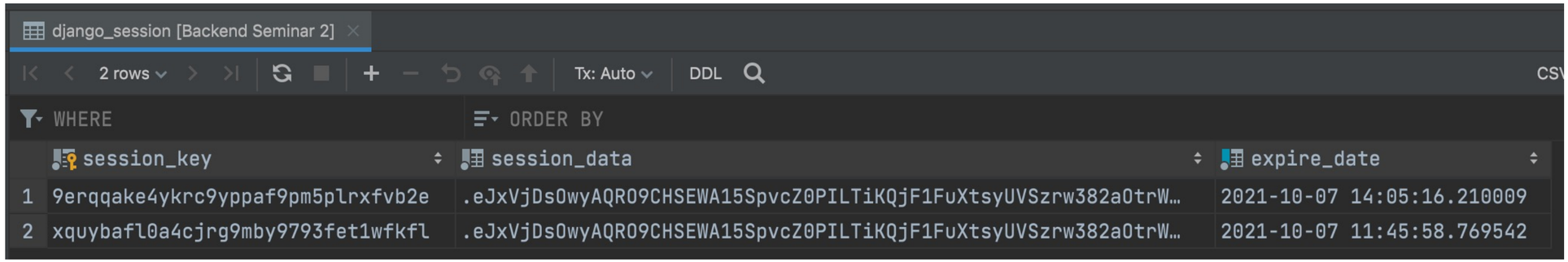
[Serializer 최대한 최대한 써먹기]

-> (practice)

WA LE
STUDIO

원래 하던 로그인

attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a



세션 열어서 서버에서 보관하고 있기

어디에 보관하든 서버에는 부하가 걸림

+ 서버가 여러 곳에 위치하고 있다면?

+ CORS 대응해주어야 함

(c.f. Reference)

WA LE STUDIO

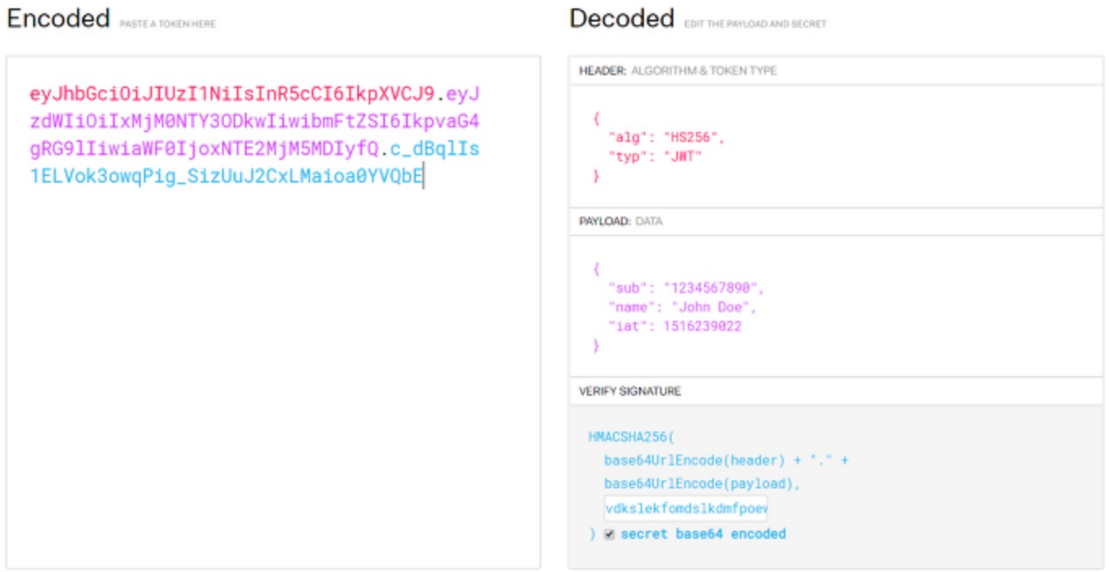
JWT 로그인

attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

“Token” : “HEADER.PAYLOAD.SIGNATURE”

[JWT 구조]

JWT는 Header, Payload, Signature의 3 부분으로 이루어지며, Json 형태인 각 부분은 Base64로 인코딩 되어 표현된다. 또한 각각의 부분을 이어 주기 위해 . 구분자를 사용하여 구분한다. 추가로 Base64는 암호화된 문자열이 아니고, 같은 문자열에 대해 항상 같은 인코딩 문자열을 반환한다.



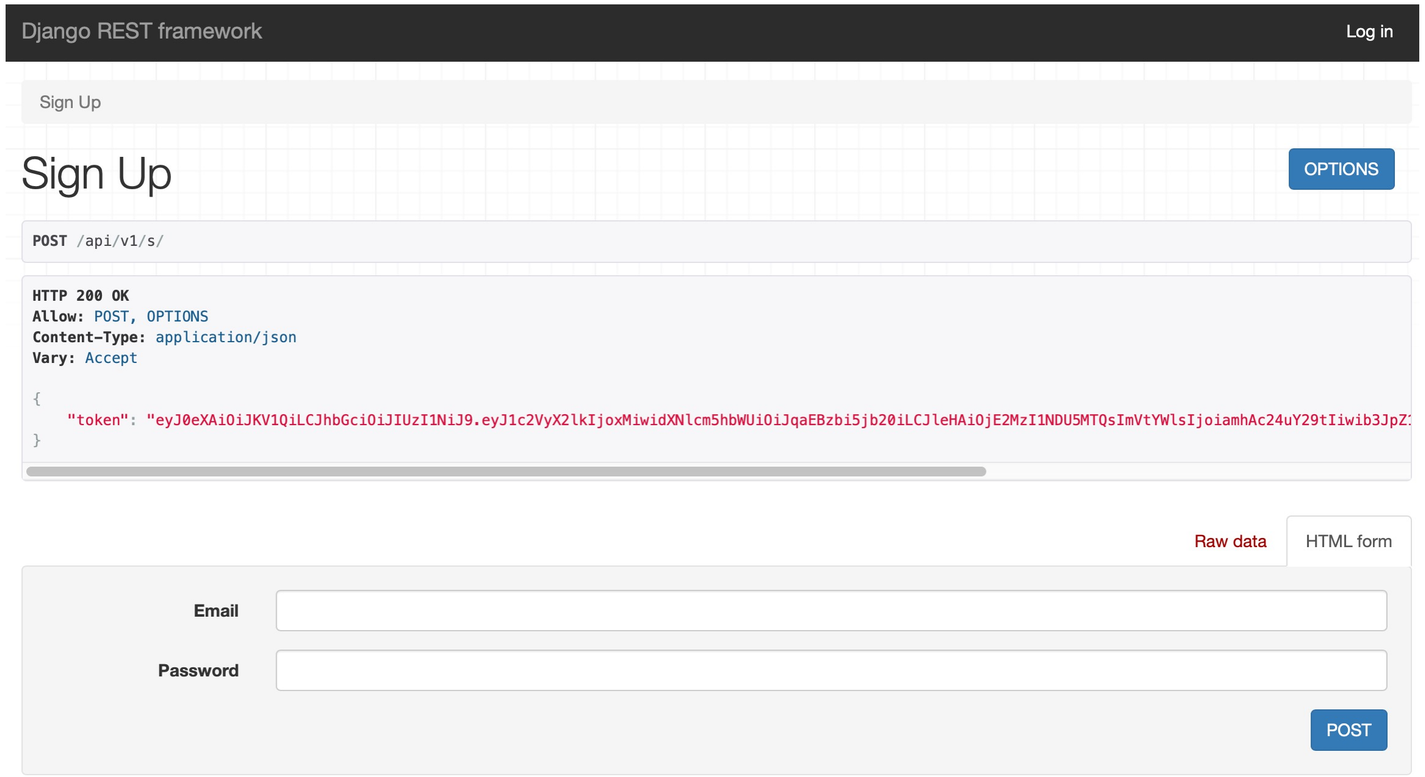
WA LE STUDIO

JWT 로그인

attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

djangorestframework.jwt

- Obtain, Verify, Refresh Token 기능
- Check yourself



c.f.

이 패키지는 deprecated 되었지만,

충분한 기능을 제공하고 있고 + 사용이 편리해서

여전히 여러 곳에서 쓰이고 있습니다.

꾸준히 관리되는 패키지를 사용하고 싶으시다면

Djangorestframework-simplejwt 참고 !

Python + Django 조합의 개발에서는

패키지 이해도 참 중요한 것 같습니다.

RTFM!!

WA#LE STUDIO

Debugging

attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

개발 시간의 8할은 에러 고치고 있는 우리들..

일단 로그를 진짜로 진짜로 꼼꼼히 집중해서 읽는다.

그래도 모르겠으면..

-> 내가 Request다.. 생각하면서 call stack을 천천히 따라가봅니다.

에러 화면을 보면 뇌정지가 오기 십상입니다..만!

빠르게 문제를 파악하는 능력을 길러봅시다.

(+ 그렇다고 빠르게 고치는 건 좋지 않아요) ->

WA LE
STUDIO

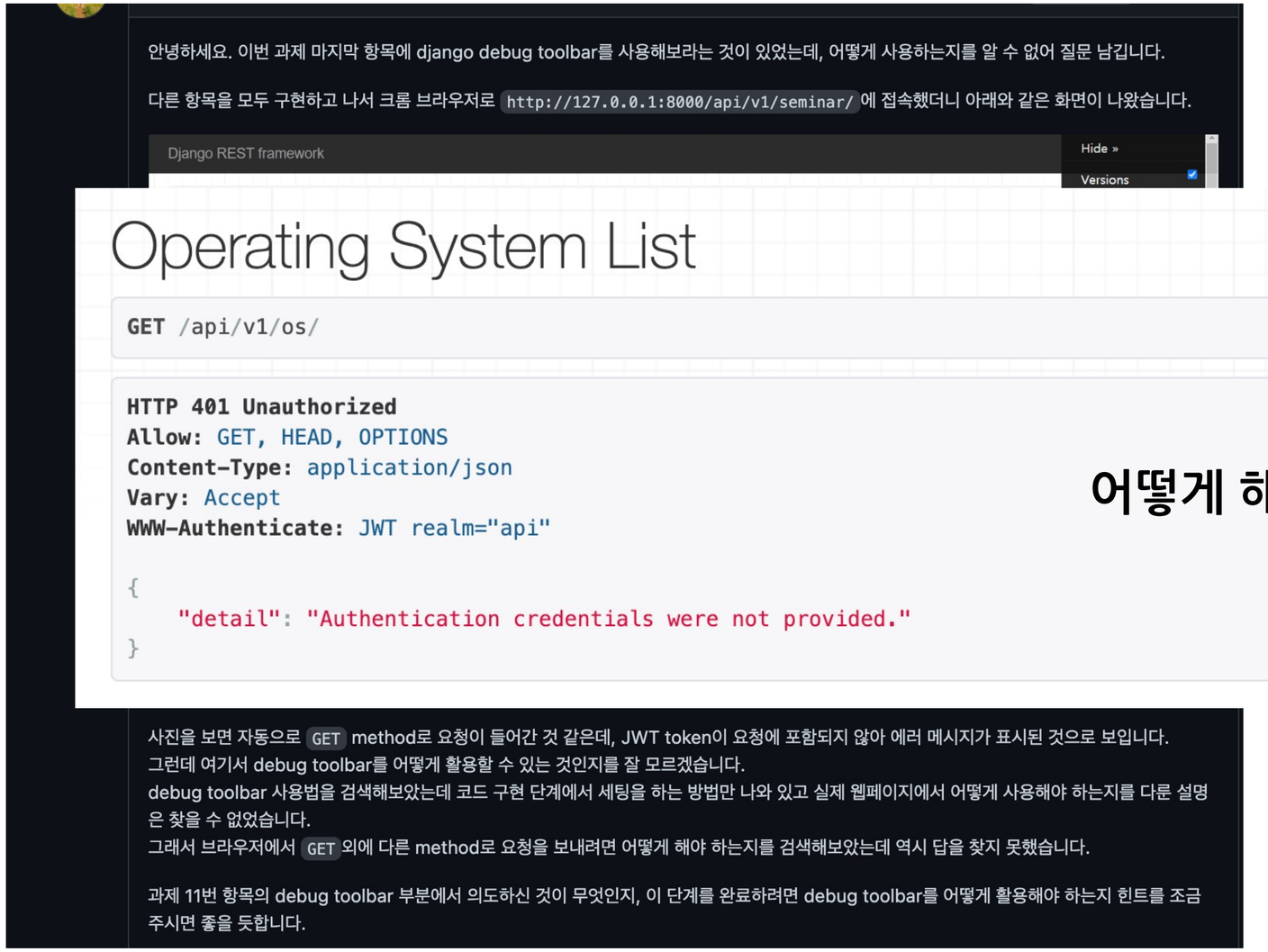
Django Debug Toolbar

attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

과제 11번에 들어가있는데.. 왜 안됐을까?

안녕하세요. 이번 과제 마지막 항목에 django debug toolbar를 사용해보라는 것이 있었는데, 어떻게 사용하는지를 알 수 없어 질문 남깁니다.

다른 항목을 모두 구현하고 나서 크롬 브라우저로 `http://127.0.0.1:8000/api/v1/seminar/` 에 접속했더니 아래와 같은 화면이 나왔습니다.



어떻게 해결하면 될까

사진을 보면 자동으로 `GET` method로 요청이 들어간 것 같은데, JWT token이 요청에 포함되지 않아 에러 메시지가 표시된 것으로 보입니다.
그런데 여기서 debug toolbar를 어떻게 활용할 수 있는 것인지를 잘 모르겠습니다.
debug toolbar 사용법을 검색해보았는데 코드 구현 단계에서 세팅을 하는 방법만 나와 있고 실제 웹페이지에서 어떻게 사용해야 하는지를 다룬 설명은 찾을 수 없었습니다.
그래서 브라우저에서 `GET` 외에 다른 method로 요청을 보내려면 어떻게 해야 하는지를 검색해보았는데 역시 답을 찾지 못했습니다.

과제 11번 항목의 debug toolbar 부분에서 의도하신 것이 무엇인지, 이 단계를 완료하려면 debug toolbar를 어떻게 활용해야 하는지 힌트를 조금 주시면 좋을 듯합니다.

WA LE STUDIO

Django Debug Toolbar

attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

브라우저가 보내는 요청에는 기본적으로 헤더를 추가할 수가 없다.

해결책

1. 인증 잠시 막아 버리기

2. 인증 방식 여러 개 허용 해주기

3. 크롬 해킹해서 헤더 수정해서 넣어주기 (ref)

서버 개발자 할거면.. 당연히 2번을 생각해주자

WA LE
STUDIO

Django Debug Toolbar

attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

Query Benchmarking

쿼리 분석 제공

SQL queries from 1 connection

default
5.15 ms (16 queries including 13 similar and 13 duplicates)

~~-> 죄책감 느껴야 되는 쿼리~~

쿼리 최적화는 언제나 중요하다.

(웨이터 젓가락 천만 개 예시를 기억하자)

WA LE
STUDIO

attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

Table of Contents

WA LE
STUDIO

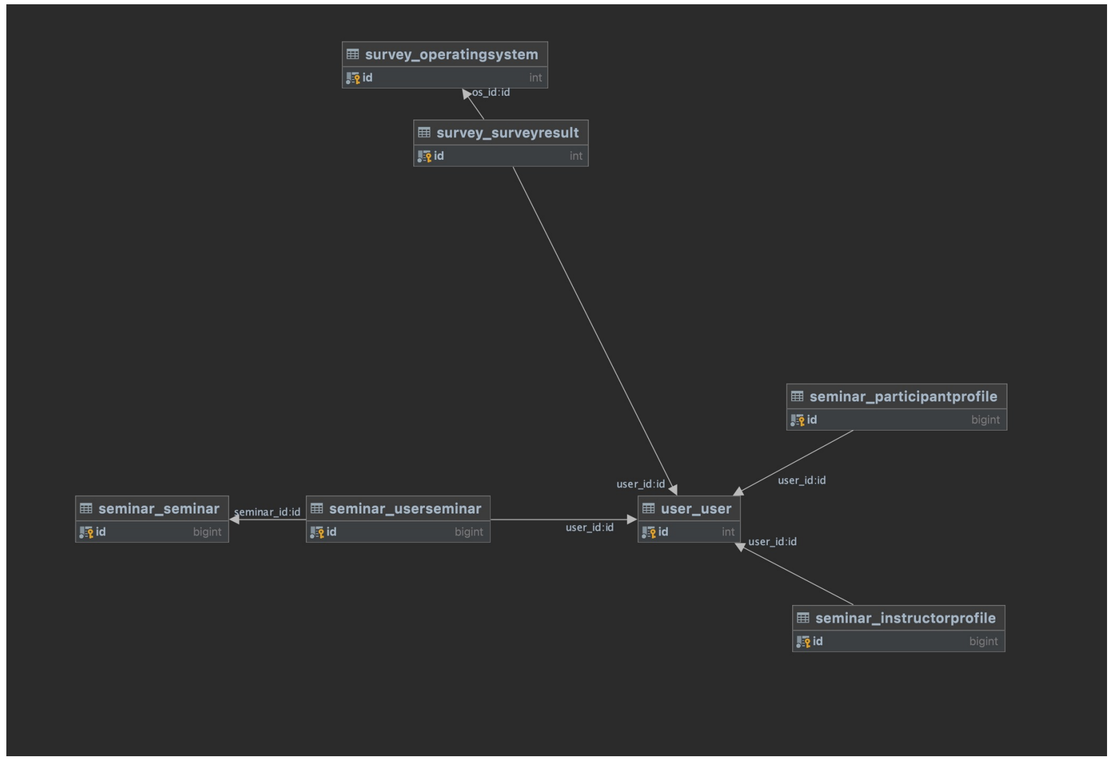
attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

DB & QuerySet

- Complicated Queries

이제 슬슬 복잡해지기 시작하는 DB diagram

DB가 복잡해질수록 요구 사항이 까다로워지기 시작한다..!



Ex. 유저의 설문 결과에 담긴 운영 체제가 리눅스가 아니면서

강사 프로필이 없으면서 세미나 3번을 수강하는 동시에

그 어떤 세미나도 드랍한 적이 없는 유저를 찾고 싶다.

WA LE STUDIO

attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

DB & QuerySet

- Complicated Queries

QuerySet 복습

쿼리셋은 게으르다

- 필요해 지기 전까지 쿼리 안 찍음

쿼리셋은 캐싱을 한다

- 같은 객체가 여러 번 불리게 되면, 디비에 굳이 여러 번 들리지 않음

._result_cache

venv/lib/python3.8/site-packages/django/db/models/query.py:349 보고 또 보고 또 보고 또 보자

WA#LE STUDIO

attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

DB & QuerySet

- Complicated Queries

Select_related

-> Use In OneToOne, ForeignKey(정참조) Relationship

Prefetch_related

-> Use In Every Relationship

직접 해보기!!

venv/lib/python3.8/site-packages/django/db/models/query.py:349 보고 또 보고 또 보고 또 보자

(Ref.)

WA LE STUDIO

attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

DB & QuerySet

- Complicated Queries

Query Expressions (Ref. 공식 문서)

-> F 객체가 뭔지는 알고 있도록 하자

Complex Query Lookups (Ref. 공식 문서)

-> 얘도 Q 객체가 뭔지 파악하기

(Ref. Q)

(Ref. F)

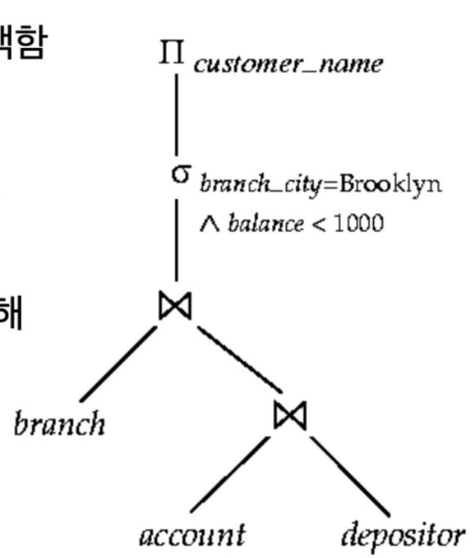
WA LE
STUDIO

attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

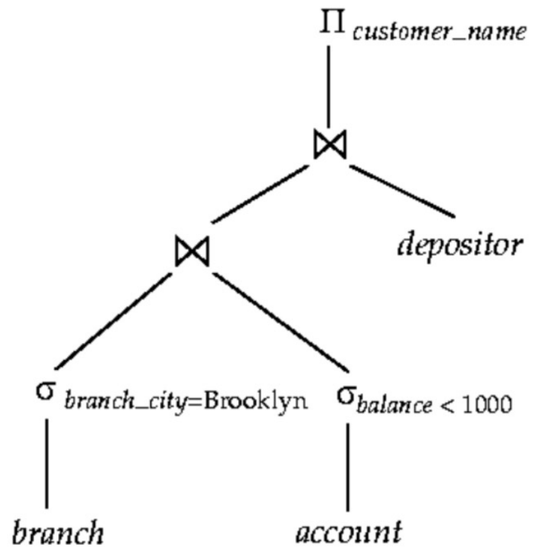
DB & QuerySet
- BenchMarking

DB Execution plan

- DBMS는 논리적으로 같은 여러 plan 생성
- 각 추정 cost를 고려하여 최적으로 예상되는 plan 택함
- cost는 table 특성 및 통계뿐 아니라
  disk access, CPU, network 등 많은 요소가 영향
- 한정된 main memory 내에서 많은 record들에 대해
  어떻게 join, merge 할지 등의 문제도 있음



(a) Initial expression tree



(b) Tree after multiple transformations

- **number of block transfers** *from disk*
  - t_T – time to transfer one block: 10~40 ms
- **number of seeks**
  - t_S – time for one seek: 8~20 ms (disk seek + rotational delay)

WA LE STUDIO

DB & QuerySet

- BenchMarking

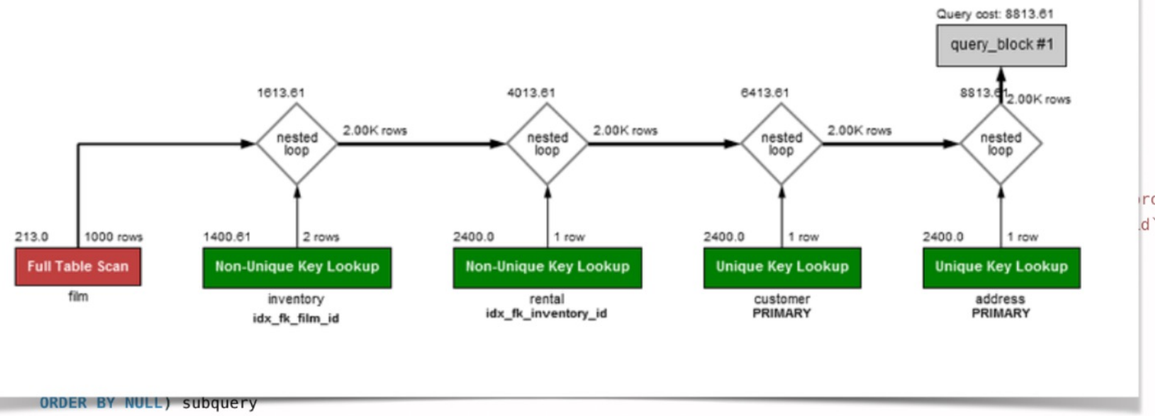
attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

https://snowple.tistory.com/377
https://cheese10yun.github.io/mysql-explian/

DB Execution plan

- **MySQL EXPLAIN**
- Extended EXPLAIN
  - JSON format
  - Visual EXPLAIN
  - From version 5.6.5

```
1  EXPLAIN
2  SELECT
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20                                                    ORDER BY NULL) subquery
```



100% 1:1

Result Grid Filter Rows: Search Export:

id	select_type	table	type	possible_keys	key
1	PRIMARY	<derived2>	ALL	실행 계획 선정 전 후보가 된 index 목록	실행 계획으로 선정된 index
2	DERIVED	universities_university	ref		
2	DERIVED	accounts_teacheruniversity	ref		
2	DERIVED	accounts_teacherprofile	eq_ref		
2	DERIVED	accounts_user	eq_ref		
2	DERIVED	answers_answer	ref		

WA LE STUDIO

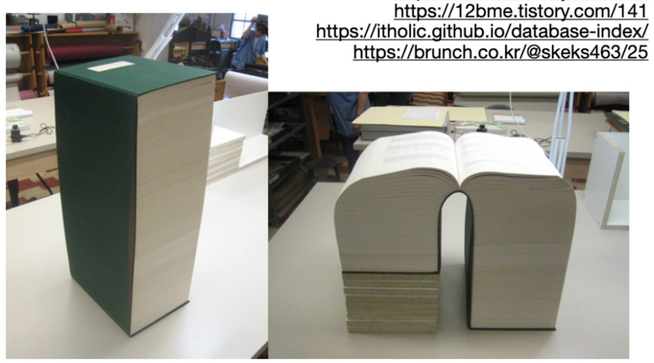
attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

DB & QuerySet

- Indexing

쿼리 가져오는 지름길; SHOW INDEX from …

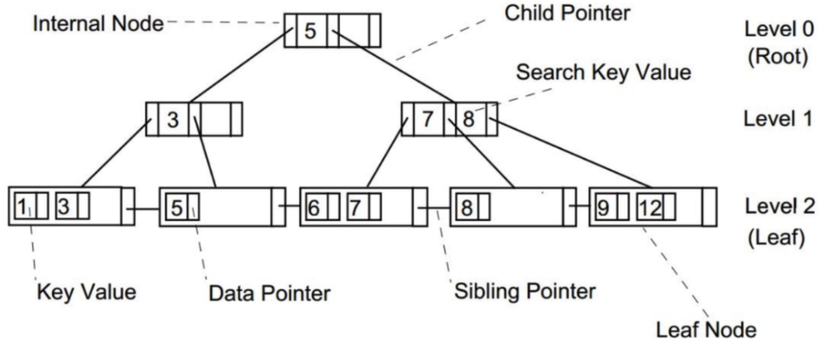
https://12bme.tistory.com/138
https://12bme.tistory.com/141
https://itholic.github.io/database-index/
https://brunch.co.kr/@skeks463/25



Index

모르면 반드시 터진다

- 어떻게 빨리 찾을 것인가?
  - B-Tree
  - Hash
- 시간과 공간의 trade-off
  - 다 걸어두면 좋은가?



참고

db_index=True

Or

```python
from django.db import models

class Customer(models.Model):
    first_name = models.CharField(max_length=100)
    last_name = models.CharField(max_length=100)

    class Meta:
        indexes = [
            models.Index(fields=['last_name', 'first_name']),
            models.Index(fields=['first_name'], name='first_name_idx'),
        ]
```

WA LE STUDIO

Testing : UnitTest

attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

Integration test Ref

E2E test Ref

테스트의 종류

: Unit Test -> Integration Test -> E2E Test

우리가 지금까지 사용한 방법은?

잘 되나..? Postman에서 호출 해보기, 크롬에서 확인해보기 (E2E)

좀 더 가볍고, 기능 자체에 대해서만 테스트 할 수 없을까???

-> Try Unit Test

WA LE
STUDIO

Testing : UnitTest

attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

Unit Test
-> 프로그램의 각 부분을 고립 -> 각 파트가 제대로 작동하는지 확인할 필요

(이를 잘 수행하기 위해서는, Service Layer에 대한 적절한 분리가 필요)

Python & Django는 기본적인 Unit Test Tool 지원

(RTFM RTFM)

WA LE
STUDIO

Testing : Unittest

attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

```python
class TestExample(TestCase):

    def add(self, a, b):
        return a + b

    def setUp(self):
        OperatingSystem.objects.create(
            name='os'
        )
        User.objects.create_user(
            email='user@user.com',
            password='password'
        )

    def test_check(self):

        cnt = OperatingSystem.objects.filter(name='os').count()
        self.assertEqual(cnt, 1)

        survey = SurveyResult.objects.filter(
            rdb__gt=F('python')
        )
        self.assertNumQueries(2)

        client = self.client

        client.force_login(User.objects.get(id=1))
        response = client.get('/api/v1/user/me/')
        self.assertEqual(response.status_code, status.HTTP_200_OK)
```

<< TC 예제

```python
from django.test import TestCase
```

self.client : 요청 테스트 가능, response 반환

- client({HEADER_NAME}=…) 로 헤더 설정
- self.force_login으로 인증 스킵 가능
- get, post 등의 요청 가능

self.assert … : 동작에 대한 검증

def setUp : test할 db 환경 설정

WA#LE STUDIO

Testing : Unittest

attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

```python
class TestExample(TestCase):

    def add(self, a, b):
        return a + b

    def setUp(self):
        OperatingSystem.objects.create(
            name='os'
        )
        User.objects.create_user(
            email='user@user.com',
            password='password'
        )

    def test_check(self):

        cnt = OperatingSystem.objects.filter(name='os').count()
        self.assertEqual(cnt, 1)

        survey = SurveyResult.objects.filter(
            rdb__gt=F('python')
        )
        self.assertNumQueries(2)

        client = self.client

        client.force_login(User.objects.get(id=1))
        response = client.get('/api/v1/user/me/')
        self.assertEqual(response.status_code, status.HTTP_200_OK)
```

<< TC 예제

```python
from django.test import TestCase
```

self.client : 요청 테스트 가능, response 반환

- client({HEADER_NAME}=…) 로 헤더 설정
- self.force_login으로 인증 스킵 가능
- get, post 등의 요청 가능

self.assert … : 동작에 대한 검증

def setUp : test할 db 환경 설정

WA#LE STUDIO

Testing : Unittest

attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

./manage.py test 한 줄이면 끝!

Code Together

./manage.py test {test_class_name}

./manage.py test --help

>> check options

WA LE STUDIO

Testing : Unittest +

attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

+ TestCase 이외에도 다양한 SubClasses

+ 손쉽게 DB setUp을 돕는 RequestFactory

See also)

PyTest

- Unittest 와 달리 클래스 선언을 해줄 필요 없음
- 고도화된 Fixture; 테스트 확장성 및 간편함 증가

WA LE
STUDIO

Deployment

attendanceKey : 32ba35e4-558f-4b20-be9b-9ac9bd48580a

시간 관계상 나가지 못한 관계로

다음 세미나 시작 때 다루고자 합니다.

Next Seminar:

- Deployment
- Caching
- Celery & 다양한 성능 개선 방식

WA LE
STUDIO

Q&A

WAFFLE STUDIO

WAFFLE STUDIO