**C. SERVER**

**ㄱ. 사용한 프레임워크와 모듈, 지원 프로그램**

**서버구성에 사용**

**서버 프레임워크 : Django 3.0.5**

**API call 구성 : Django Rest Framework 1.2.9**

**배포작업에 이용**

**uWSGI 2.0**

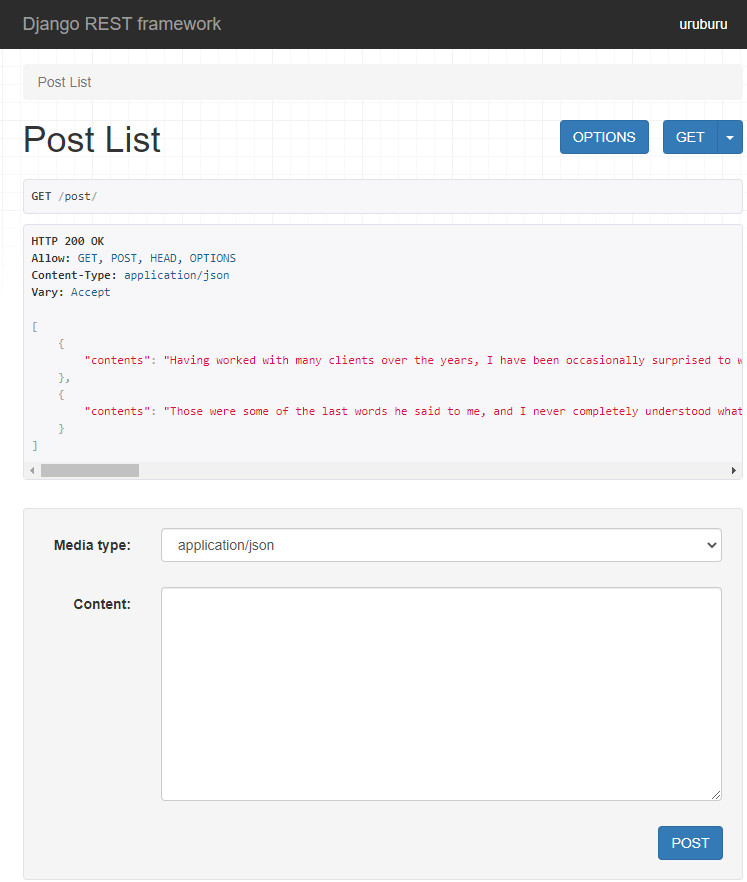
**Nginx 1.14**

**AWS EC2**

**ㄴ. 중간보고서에서의 변경점**

**ㄴ-1 API call 구현**

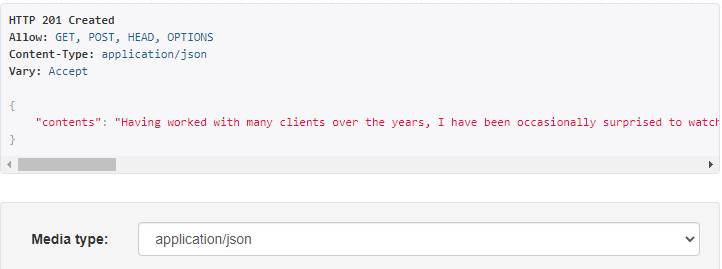
ㄴ-1-1 Django의 Rest framework를 이용하여 get 과 post 콜을 구현했다.



해당 화면은 장고 프레임워크 내부에서 맥의 POSTMAN처럼 만든 GET과 POST콜을 테스트해볼 수 있는 화면이다. 정상적으로 json 통

신이 되는 것을 확인할 수 있다.

이와 같이 포스트콜을 보내서 201 응답을 받음을 확인할 수 있다.



ㄴ-1-2 공통된 직렬화 형식

Json 형식으로

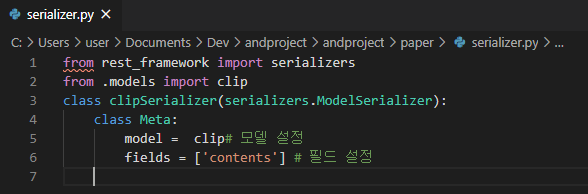
{

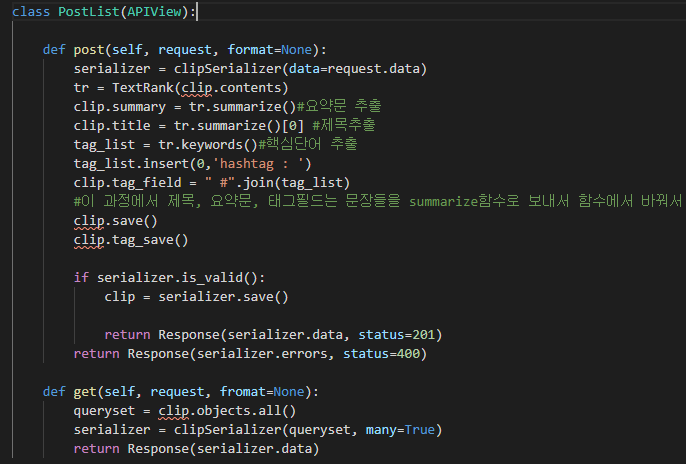
"contents": "Having worked with many clients over the years, I have been occasionally surprised to watch as one of them experienced an amazing “aha!” of recognition and had an internal shift after watching a movie. These clients were able to identify with characters who had struggles similar to their own. But while they identified with the characters, it was also somehow easier to maintain their distance “outside” the experience. In other words, they maintained an observer perspective. In some cases, instead of identifying with a character, the client was highly critical of them. Either way, the same active principles seemed to apply, and combining the movie experience with psychological exploration during our sessions had powerful results."

}

와 같이 {“컨텐츠” : “내용”} 의 json 형식을 취해서 API콜을 구현했다.

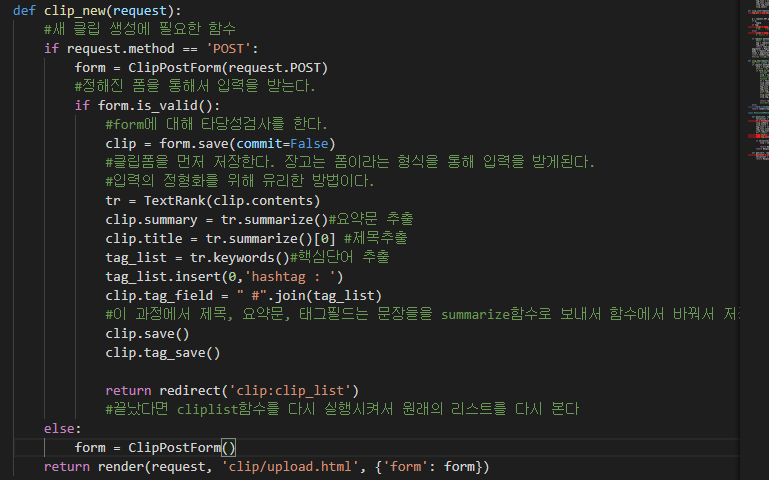
이를 위한 모듈 serializer를 구현했다.

  
이와 함께 사용하는 함수 또한 구현했다.



전체 소스코드는 기존 함수 뷰와 크게 다르지 않다. 이에 대한 비교는 아래에 웹콜을 함께 만들어두었다. Getcall 보다는 Postcall에 집중해서 구현했다.

**ㄴ-2 TextRank 알고리즘과의 연동**



기존함수에서 텍스트랭크를 이용해 요약문, 제목, 핵심단어를 추출해 정해진 양식에 삽입하는 함수로 바꾸었다.

**ㄷ. 배포**

AWS위에 올리는 배포작업을 거쳐야 어플리케이션과 API통신을 구현할 수 있기에 배포작업을 진행했다. 전체적인 구조는 다음과 같다.

사용자 <-> Nginx 웹서버 <-> uWSGI <-> Django

**ㄷ-0 AWS**

. 이 모든 과정을 소화하는 클라우드 서비스인 AWS에 업로드 작업을 진행했으며, EC2 프리티어 우분투 16.04를 이용해서 구성했다. 인바운드 8080포트를 여는 보안규칙을 구성했으며 기본적인 우분투 기반위에서 SSH콜과 SCP 콜을 이용하여 서버를 조작한다.

**ㄷ-1 uWSGI**

**uwsgi \**

**--http :8080 \**

**--home /home/ubuntu/pyenv/versions/summary \**

**--chdir /srv/project/andproject \**

**-w config.wsgi**

위와 같은 구성을 이용해서 wsgi를 구동시킴

**ㄷ-2 Nginx**

server {

listen 8080; //8080번 포트를 사용하겠다는 뜻이다

server\_name \*.[compute.amazonaws.com](http://compute.amazonaws.com/); //아래와 같은 주소로 접속하는걸 받겠다는 뜻이다.

charset utf-8;

client\_max\_body\_size 128M;

location / {

uwsgi\_pass unix:///tmp/andproject.sock;

include uwsgi\_params;

}

}

Nginx는 다음과 같이 설정하여 SCP 통신으로 AWS에 업로드 한다.

**ㄷ-3 nginx와 uWSGI의 연동**

nginx <-> wsgi 통신은 기본적으로 socket 방식을 사용한다. HTTP 방식을 사용할 수 있지만 통신에서 로스가 많이 발생하는 단점이 있다. 그에 비해서 wsgi app은 서버 안쪽에서 서버의 요청을 django 어플리케이션에 중계하는 역할을 하기 때문에 무거운 HTTP 요청을 보다는 socket이라는 전송방식을 사용한다.

[Unit]

Description=uWSGI service

After=syslog.target

[Service]

ExecStart=/home/ubuntu/.pyenv/versions/uwsgi-env/bin/uwsgi -i /srv/ec2\_deploy\_project/.config/uwsgi/mysite.ini

Restart=always

KillSignal=SIGQUIT

Type=notify

StandardError=syslog

NotifyAccess=all

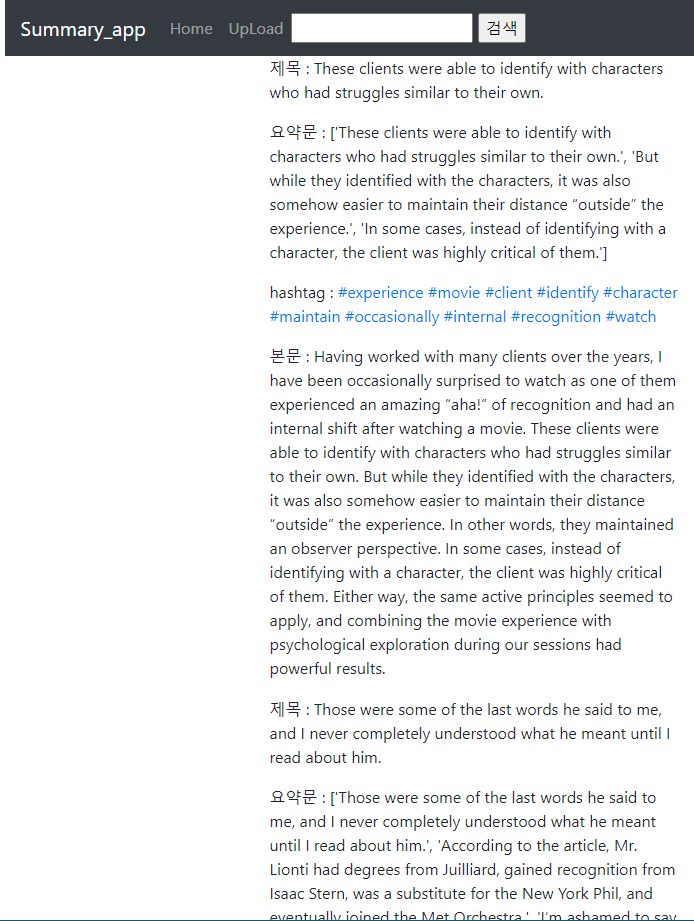
[Install]

WantedBy=multi-user.target

명세는 다음과 같이 작성하였다. 이를 데몬에 넣어 서버컴퓨터 부팅될 때, 자동으로 백그라운드에서 동작하게 만들어준다.

**ㄹ. 결과화면**

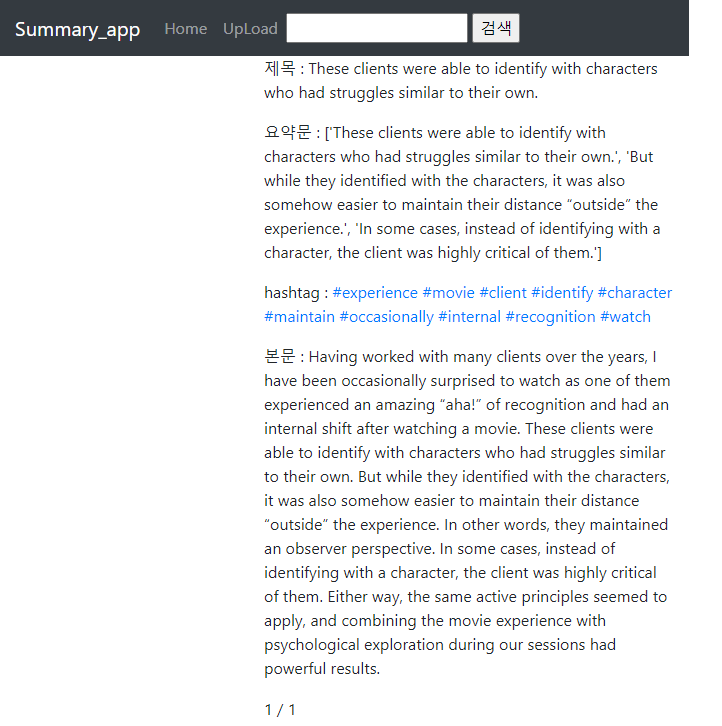
**ㄹ-1 사용자 화면**

**🡨-가장 중요한 문장을 제목으로 사용**

**🡨가장 중요한 세 문장을 요약문으로 구현**

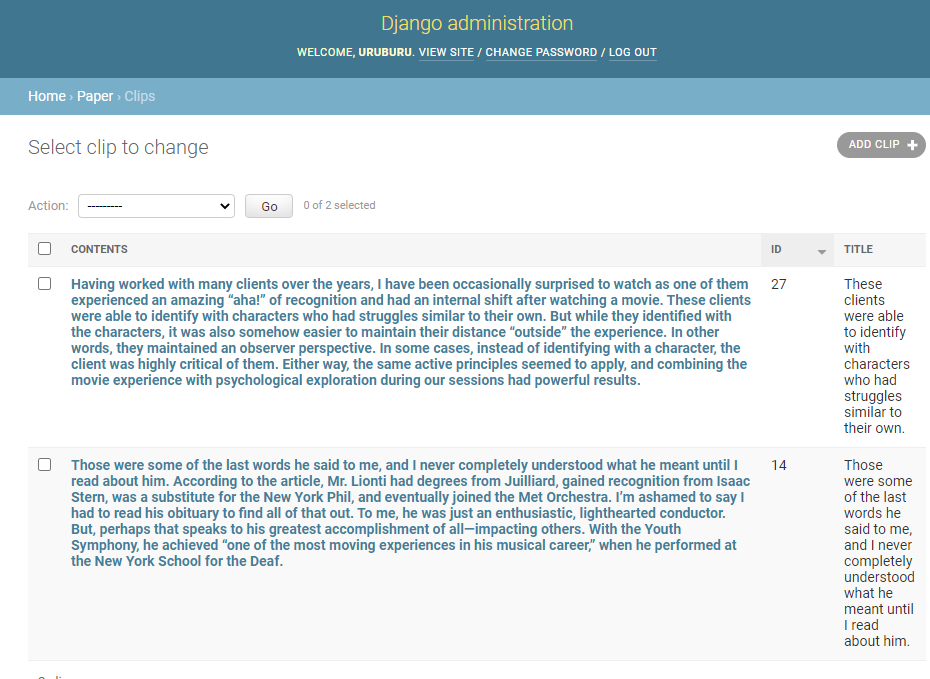
**🡨----해쉬태그 구현(가장 중요도가 높은 10개 단어)**

**🡨이후 입력받은 본문 출력**

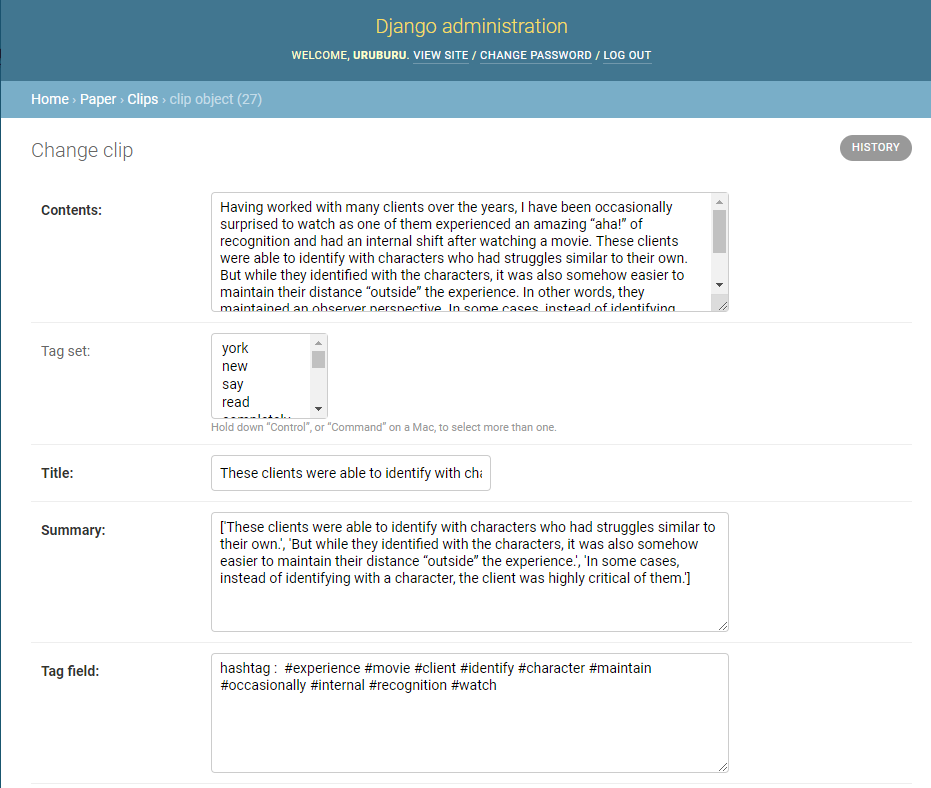


🡨 해쉬태그를 이용한 검색기능 활용시 Experience를 검색하면 나오는 화면이다.

**ㄹ-2 관리자 화면**



이와 같이 클립리스트를 관리자 화면에서 관리하는 작업이 가능



내부 내용을 수정하는 작업도 가능하다