Ask Company

리액트와 함께 장고 시작하기 / 웹 프론트엔드 기초

장고가 static 파일을 다루는 방법

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

Static & Media 파일

Static 파일

개발 리소스로서의 정적인 파일 (js, css, image 등)

앱 / 프로젝트 단위로 저장/서빙

Media 파일

FileField/ImageField를 통해 저장한 모든 파일

DB필드에는 저장경로를 저장하며, 파일은 파일 스토리지에 저장

프로젝트 단위로 저장/서빙

static 파일

장고 static 파일 경로

장고는 One Project, Multi App 구조

한 App을 위한 static 파일을 **app/static/app**경로에 둡니다.

프로젝트 전반적으로 사용되는 static 파일은 **settings.STATICFILES_DIRS**에 지정된 경로에 둡니다.

다수 디렉토리에 저장된 static 파일은 collectstatic 명령을 통해, settings.STATIC_ROOT에 지정한 경로로 모아서 (복사)해서 서비스에 사용

Static 파일, 관련 settings 예시

각 설정의 디폴트 값

STATIC_URL = None

각 static 파일에 대한 URL Prefix

템플릿 태그 {% static “경로” %} 에 의해서 참조되는 설정

항상 / 로 끝나도록 설정

STATICFILES_DIRS = []

File System Loader에 의해 참조되는 설정

STATIC_ROOT = None

python manage.py collectstatic 명령이 참조되는 설정

여러 디렉토리로 나눠진 static파일들을 이 경로의 디렉토리로 복사하여, 서빙

배포에서만 의미가 있는 설정

https://docs.djangoproject.com/en/2.1/howto/static-files/

추천 settings

```
STATIC_URL = '/static/'

STATIC_ROOT = os.path.join(BASE_DIR, 'static')

STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'askdjango', 'static'),
]
```

Static Files Finders

Template Loader와 유사

설정된 Finders를 통해, static 템플릿이 있을 **디렉토리 목록**을 구성

장고 서버 초기 시작 시에만 1회 작성

디렉토리 목록에서 지정 상대경로를 가지는 static 파일 찾기.

대표적인 2가지 Static Files Finders

App Directories Finder

"장고앱/static/" 경로를 "디렉토리 목록"에 추가

File System Finder

settings.STATICFILES_DIRS 설정값을 "디렉토리 목록"에 추가

```
STATICFILES_FINDERS = [
    'django.contrib.staticfiles.finders.FileSystemFinder’,
    'django.contrib.staticfiles.finders.AppDirectoriesFinder',
]
```

템플릿에서 static URL 처리 예시 (1)

방법1) settings.STATIC_URL, Prefix를 하드코딩하기

하지만, settings.STATIC_URL 설정은 언제라도/프로젝트마다 변경될 수 있음. 하드코딩하는 것이 번거롭기도하고 변경이 되었을 때 하나하나 수정해줘야함.

무엇보다, 배포 시에는 static_url 설정값이 변경됩니다.

클라우드 정적 스토리지나 CDN 사용 시

```
<img src="/static/blog/title.png" />
```

템플릿에서 static URL 처리 예시 (2)

방법2) Template Tag를 통한 처리

프로젝트 설정에 따라, 유연하게 static url prefix가 할당됩니다.

~~~~

```
{% load static %}
<img src="{% static "blog/title.png" %}" />
```

개발환경에서의 static 파일 서빙 (1/2)

개발서버를 쓰고, and settings.DEBUG = True 일 때에만, 지원

프로젝트/urls.py에 Rule이 명시되어 있지 않아도, 자동 Rule 추가

이는 순수 **개발목적**으로만 제공

개발서버를 쓰지 않거나, settings.DEBUG = False 일 때에는

별도로 static 서빙 설정을 해줘야합니다.

개발환경에서의 static 파일 서빙 (2/2)

myproj/static/**main.css** => http://localhost:8000/static/**main.css** 경로로 접근 가능

myproj/static/**jquery/jquery-3.4.1.min.js** => http://localhost:8000/static/**jquery/jquery-3.4.1.min.js**

myproj/static/**bootstrap/4.3.1/css/bootstrap.min.css** => http://localhost:8000/static/**bootstrap/4.3.1/css/bootstrap.min.css**

blog/static/**blog/style.css** => http://localhost:8000/static/**blog/style.css** 경로로 접근 가능

blog/static/**blog/blog.js** => http://localhost:8000/static/**blog/blog.js** 경로로 접근 가능

shop/static/**shop/shop.js** => http://localhost:8000/static/**shop/shop.js** 경로로 접근 가능

URL을 통해 파일시스템에 직접 접근하는 것이 아니라, 지정 이름의 STATIC 파일을 장고의 StaticFiles Finder에서 대신 찾아 그 내용을 읽어서 응답하는 것

static 서빙을 하는 여러가지 방법

1. 클라우드 정적 스토리지나 CDN 서비스를 활용
2. apache/nginx 웹서버 등을 통한 서빙
3. 장고를 통한 서빙

whitenoise 라이브러리를 활용해서 가능 → http://whitenoise.evans.io → Heroku 배포에 필요

collectstatic 명령

실 서비스 배포 전에는 필히 본 명령을 통해, 여러 디렉토리에 나눠져있는 static 파일들을 한 곳으로 복사

복사하는 대상 디렉토리 : settings.STATIC_ROOT

왜냐하면, 여러 디렉토리에 나눠 저장된 static 파일들의 위치는 “현재 장고 프로젝트” 만이 알고 있음. 외부 웹서버는 전혀 알지 못함.

외부 웹서버에서 Finder의 도움없이도 static 파일을 서빙하기 위함.

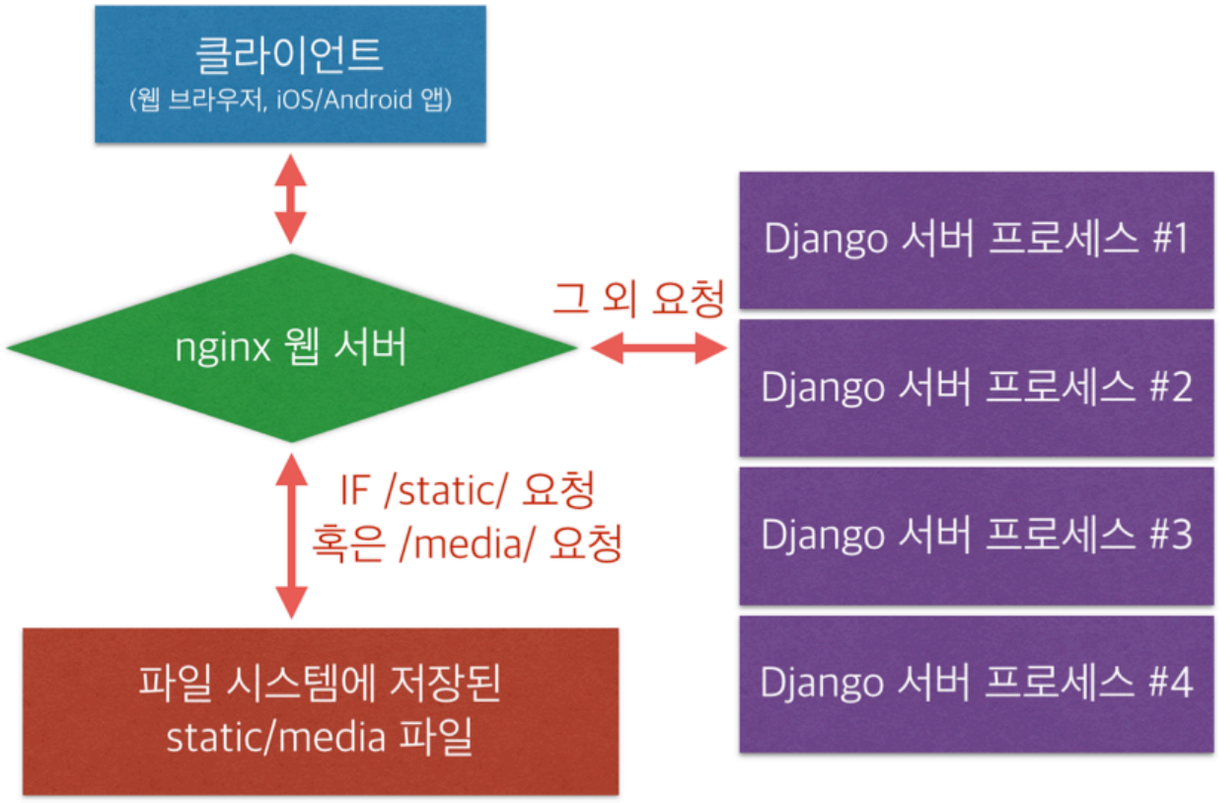
한 디렉토리에 모두 모여있기에, Finder의 도움이 필요가 없음.

외부 웹서버에 의한 static/media 컨텐츠 서비스

정적인 컨텐츠는, 외부 웹서버를 통해 처리하면, 효율적인 처리

정적 컨텐츠만의 최적화 방법 사용

- memcache/redis 캐시 등
- CDN (Content Delivery Network)



nginx 웹서버에서의 static 설정 예시

```
server {
    # 중략
    location /static {
        autoindex off;
        alias /var/www/staticfiles;  # settings.STATIC_ROOT
    }
    location /media {
        autoindex off;
        alias /var/www/media;        # settings.MEDIA_ROOT
    }
}
```

배포 시에 static 처리 프로세스

1. “서비스용settings”에 배포 static 설정
2. 관련 클라우드 스토리지 설정, 혹은 아파치/nginx static 설정
3. 개발이 완료된 static파일을, 한 디렉토리로 복사

   python manage.py collectstatic --settings=서비스용settings

   Storage 설정에 따라, 한 번에 클라우드 스토리지로의 복사를 수행되기도 함.

   settings.STATIC_ROOT 경로로 복사됨.

4. settings.STATIC_ROOT경로에 복사된 파일들을 배포서버로 복사

   대상 : 클라우드 스토리지, 혹은 아파치/nginx에서 참조할 경로

5. static 웹서버를 가리키토록 sesttings.STATIC_URL 수정

static 관련 라이브러리

django-storages

https://django-storages.readthedocs.io

Azure Storage, Amazon S3, Google Cloud Storage, FTP 등 지원

django-storages-azure

https://pypi.org/project/django-storages-azure/

브라우저 캐싱

브라우저 캐싱

브라우저 캐시 기간을 설정해 주면 그 기간 동안은 웹브라우저가 해당 파일을 다시 다운받지 않고 캐싱된 내용을 사용하기 때문에 트래픽이 줄어들고, 속도도 빨라집니다.

Expires 헤더 : 만료일시를 지정

- Expires: Wed, 21 Oct 2015 07:28:00 GMT
- 응답 내에 "max-age" 혹은 "s-maxage" directive를 지닌 Cache-Control 헤더가 존재할 경우, Expires 헤더는 무시

Cache-Control : 보다 다양한 캐싱 정책 지정

blog/style.css 조회 후에 변경된 파일이 반영되지 않을 경우

유저는 /blog/ 페이지에 방문하면서 브라우저에 /static/blog/style.css 파일이 다운로드되었습니다. 이때 이 파일이 24시간 동안 브라우저 캐싱이 되어있다고 생각해봅시다.

그런데, 개발하면서 CSS파일이 변경되었습니다. 파일경로는 바뀌지 않았습니다. 변경된 CSS파일이 유저페이지에 적용되길 원하지만 적용되지 않습니다. 캐싱된 이전 파일에 계속 접근하게 됩니다.

해결하기

- 방법1) 해당 파일의 캐싱이 만료될 때까지 기다립니다.
- 방법2) 브라우저 설정에서 캐싱된 내용을 삭제합니다. 크롬 브라우저에서는 "강력 새로고침"을 통해 수행 가능.
- 방법3) **해당 STATIC 리소스의 URL을 변경합니다.**

윈도우 단축키 Ctrl+Shift+R
맥 단축키 Command+Shift+R

https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/http-caching?hl=ko

클라이언트측 캐싱과 빠른 업데이트를 할려면

리소스의 URL을 변경 → 사용자가 새로이 다운로드

1. GET인자 붙이기 : 실제 파일명은 변경하지 않으면서, 브라우저가 인지하는 URL만 변경
   - **개발 시에 유용**
   - 버전을 숫자로 붙이거나
   - http://localhost:8000/static/main.css?**v=1**
   - 버전을 날짜로 붙이거나
   - http://localhost:8000/static/main.css?**v=20190503**
   - 더미로 현재시각의 timestamp을 붙입니다.
   - http://localhost:8000/static/main.css?**_=1556908299**
2. 파일명 변경하기
   - **서비스 배포 시에 유용**

커스텀 템플릿태그를 통해 STATIC URL에 더미 GET인자 붙이기

myapp/templatetags/static_tags.py

```python
import time
from django import template
from django.conf import settings
from django.templatetags.static import StaticNode

register = template.Library()

class TimestampStaticNode(StaticNode):
    def url(self, context):
        url = super().url(context)
        if settings.DEBUG:
            t = str(int(time.time()))
            if '?' not in url:
                url += '?_=' + t
            else:
                url += '&_=' + t
        return url

@register.tag('static_t')
def static_t(parser, token):
    return TimestampStaticNode.handle_token(parser, token)
```

```
{% load static_tags %}

{% static_t "blog/style.css" %}
<link href="{% static_t "blog/style.css" %}" />
<script src="{% static_t "blog/editor.js" %}"></script>
```

ex) /static/blog/style.css?_=1556908299

다양한 STATIC 리소스

직접 생성/등록한 CSS/JavaScript/Image 파일들

외부 CSS/JavaScript 라이브러리

- CDN (Contents Delivery Network) 배포판 활용
- 직접 다운로드&서빙
- 자바스크립트 팩키지 관리자를 활용하여 다운로드&서빙

CDN 배포판 활용

유명 라이브러리일 경우, 대개 CDN 배포판을 제공

개발 시에 빠른 적용을 위해서는 편리

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" />
<script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>
```

안정적인 실서비스 제공을 위해서는 다운로드&서빙을 추천

- 정적 파일 서빙을 "관리할 수 없는 외부 서비스"에 의존할 경우, 특정 유저의 해외망 접속이 원활하지 않거나, 해당 서비스 장애일 경우, 의도치않게 서비스 이용에 차질이 발생하게 됩니다.

직접 다운로드&서빙

프로젝트 전반적으로 사용될 파일들이므로, filesystem static finder에서 접근하는 경로에 넣어두고, 버전관리 대상에도 추가

- "프로젝트/static/" 경로

수집된 파일들은 버전관리 대상에 넣지 않습니다.

Life is short.
You need Python and Django.

I will be your pacemaker.