

장고 차근차근 시작하기

Second Edition

당신의 파이썬/장고 페이스메이커가 되겠습니다. ;)

EP25. static 파일을 다루는 방법

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

Static 파일, 관련 settings 예시 ([공식문서](#))

- 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 나뉜 static파일들을 이 경로의 디렉토리로 복사하여, 서빙
 - 배포에서만 의미가 있는 설정

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

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

- Template Loader와 유사

- 설정된 Finders를 통해, static 템플릿이 있을 **디렉토리 목록**을 구성
 - 장고 서버 초기 시작 시에만 1회 작성
- **디렉토리 목록**에서 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

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

```

```

템플릿에서 static URL 처리 예시 (2)

- 방법2) Template Tag를 통한 처리
 - 프로젝트 설정에 따라, 유연하게 static url prefix가 할당됩니다.

~~~~

```
{% load static %}  

```



# 개발환경에서의 static 파일 서빙

- 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.

# static 서빙을 하는 여러가지 방법

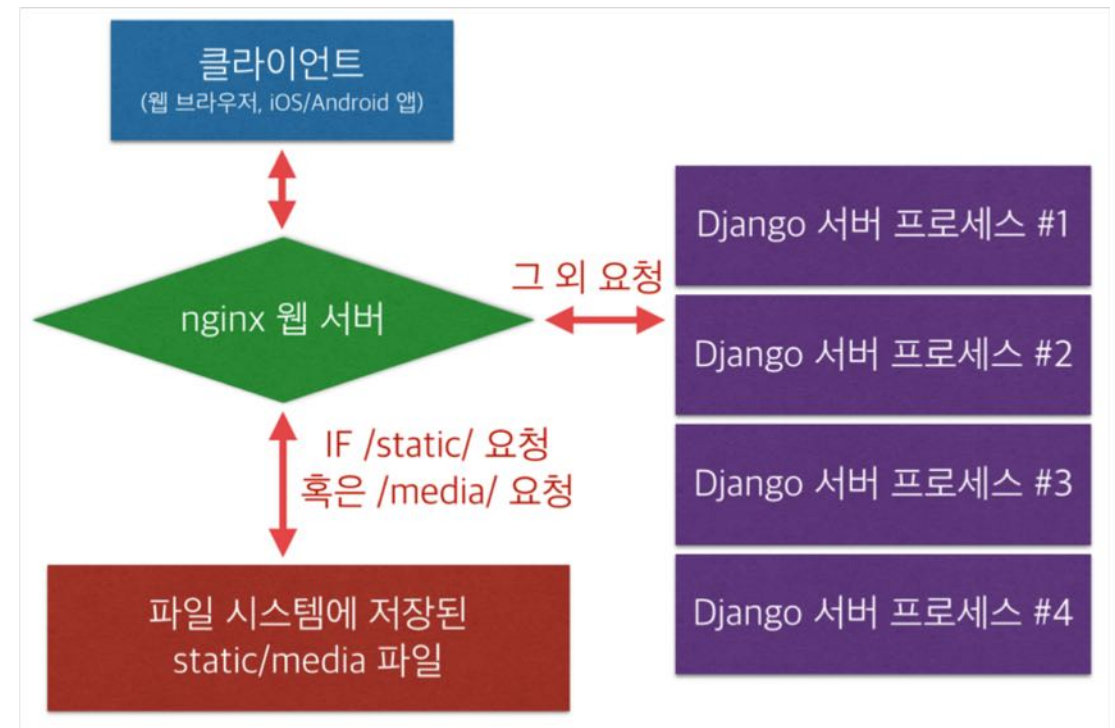
1. 클라우드 정적 스토리지나 CDN 서비스를 활용
2. apache/nginx 웹서버 등을 통한 서빙
3. 장고를 통한 서빙
  - whitenoise 라이브러리 활용 ([공식문서](#))

# collectstatic 명령

- 실 서비스 배포 전에는 필히 본 명령을 통해, 여러 디렉토리에 나뉘져있는 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 콘텐츠 서비스

- 정적인 콘텐츠는, 외부 웹서버를 통해 처리하면, 효율적인 처리
- 정적 콘텐츠만의 최적화 방법 사용
  - memcache/redis 캐시 등
  - CDN (Content Delivery Network)



# nginx 웹서버에서의 static 설정 예시

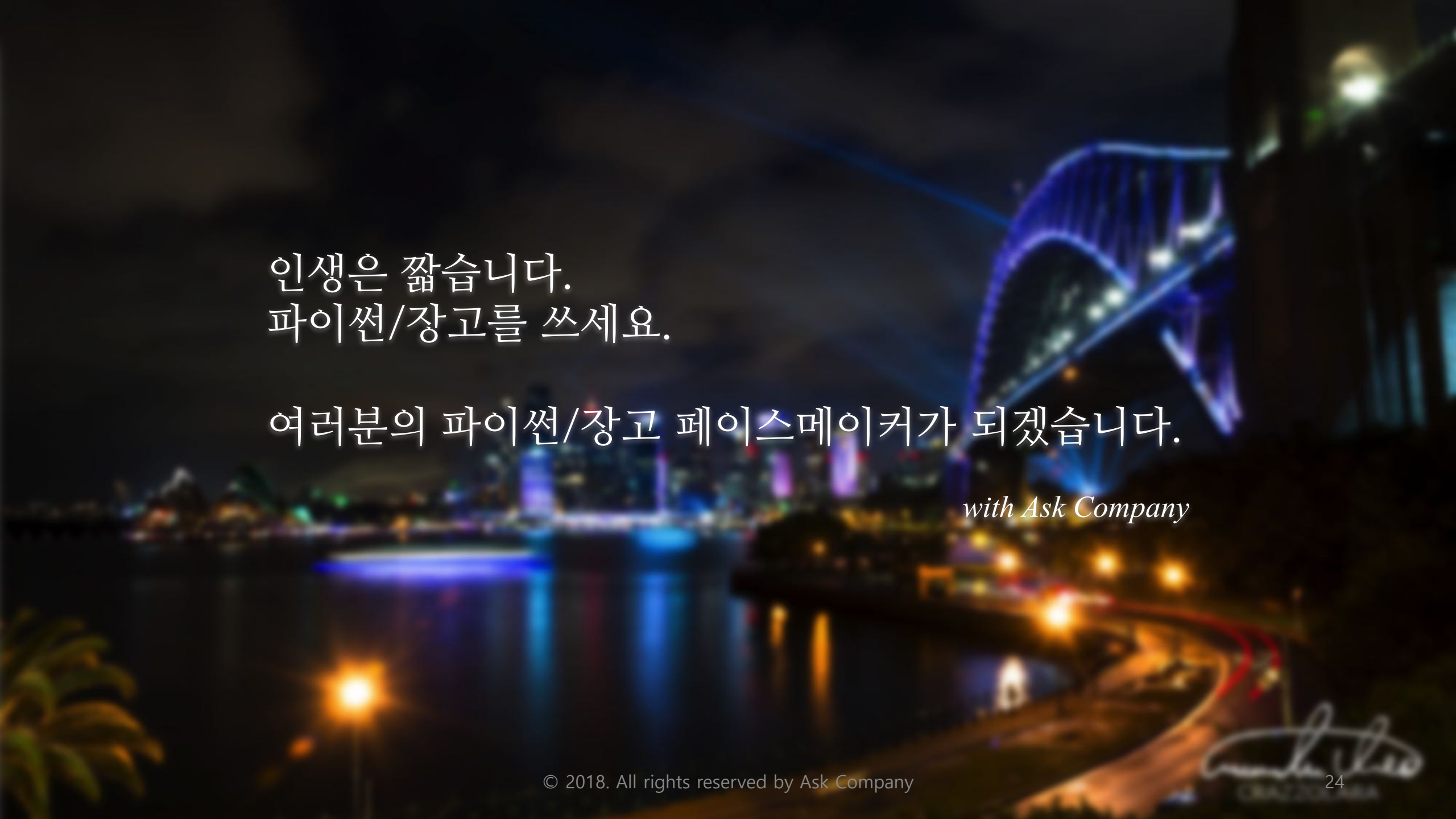
```
server {  
    # 중략  
    location /static {  
        autoindex off;  
        alias /var/www/staticfiles; # settings.STATIC_ROOT  
    }  
    location /media {  
        autoindex off;  
        alias /var/www/media;      # settings.MEDIA_ROOT  
    }  
}
```

# 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_ROOT` 경로에 복사된 파일들을 배포서버로 복사
  - 대상 : 클라우드 스토리지, 혹은 아파치/nginx에서 참조할 경로
5. static 웹서버를 가리키도록 `settings.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/>

A nighttime photograph of a cityscape featuring a bridge with blue and white lights, reflected in the water. The scene is dark, with the city lights providing the primary illumination.

인생은 짧습니다.  
파이썬/장고를 쓰세요.

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

*with Ask Company*