Django

django basics

- MVT; Model-View-Template
  - 프로젝트 뼈대 만들기 → 모델 코딩하기 → URLconf 코딩하기 → 뷰 코딩하기 → 템플릿 코딩하기
- settings.py
  - 데이터베이스 설정: 디폴트로 SQLite3 데이터베이스 엔진을 사용하는 것으로 지정
  - 애플리케이션 등록: 개발하는 앱, 즉 프로젝트에 포함되는 애플리케이션들은 모두 설정 파일에 등록
  - 템플릿 항목 설정: TEMPLATES 항목으로 지정
  - 정적 파일 항목 설정: STATIC_URL 등 관련 항목을 지정
  - 타임존 지정: 최초에는 세계표준시(UTC)로 설정되어 있는데, 한국 시간으로 변경
- models.py; ORM(Object Relational Mapping) 기법 사용, makemigration/migrate 명령
- URLconf; urls.py 프로젝트 전체 URL을 정의하는 프로젝트 URL과 앱마다 정의하는 앱 URL
- views.py; Function-based view, Class-based view
- templates; 프로젝트 베이스(루트) 디렉토리, 프로젝트 디렉토리, 프로젝트 템플릿 디렉토리, 앱 템플릿 디렉토리
- admin site
- runserver; 상용화를 고려할 때 runserver 대신 apache / nginx 선택

Virtual Environments

- 파이썬 라이브러리들 간 충돌을 방지
- ~~virtualenv~~ / venv

1. 작업할 디렉토리 생성
2. 가상 환경 생성
3. 생성된 가상 환경으로 진입 → 프롬프트 변경 확인
4. 작업
5. 가상 환경에서 빠져나옴

- 가상 환경에서 패키지 복제

```
(myvenv)$ pip3 freeze > requirenments.txt  # myvenv 가상 환경에 설치된 패키지 목록을 requirenments.txt로 저장
(new_venv)$ pip3 install -r requirenments.txt  #
```

- 가상 환경에 장고 패키지 설치

```
$ source /home/alex/VENV/djprj/bin/activate  # djprj 가상 환경으로 진입
(djprj)$ pip3 install Django  # 최신 버전의 장고 설치
(djprj)$ pip3 list  # 설치된 패키지 리스트 확인

# 파이썬 기본 라이브러리들의 위치 /usr/local/lib/python3.7/
```

```
# 파이썬 외부 라이브러리들의 위치 /home/alex/VENV/djprj/lib/python3.7/site-packages/
```

- 파이썬용 패키지 검색 사이트 https://pypi.python.org/
- 장고용 패키지 검색 사이트 https://www.djangopackages.com/

- 타임존 관리 패키지 pytz

```
(djprj)$ pip3 install pytz
```

- 패키지 설치 툴 업그레이드

```
(djprj)$ pip install -U pip setuptools wheel
```

- InsecurePlatformWarning 해결

```
(djprj)$ pip3 install pyopenssl ndg-httpsclient pyasn1
```

develop with django

- design
  - UI
  - table
  - logic
  - url

작업 순서	관련 명령/파일	필요한 작업 내용
뼈대 만들기	startproject	프로젝트 생성
	settings.py	프로젝트 설정 항목 변경
	migrate	User/Group 테이블 생성
	createsuperuser	프로젝트 관리자인 슈퍼유저를 만듦
	startapp	앱 생성
	settings.py	앱 등록
모델 코딩하기	models.py	모델(테이블) 정의
	admin.py	admin 사이트에 모델 등록
	makemigrations	모델의 변경사항 추출
	migrate	변경사항을 데이터베이스에 반영
URLconf 코딩하기	urls.py	URL 정의
뷰 코딩하기	views.py	뷰 로직 작성
템플릿 코딩하기	templates 디렉토리	템플릿 파일 작성
그 외 코딩하기	-	(없음)

- coding
  - project 생성
  - 프로젝트 설정 파일 변경
  - 기본 테이블 생성
  - 슈퍼유저 생성
  - 애플리케이션 생성
  - 애플리케이션 등록

```
(djprj)$ django-admin startproject mysite .
```

```
# settings.py

# ALLOWED_HOST 지정
ALLOWED_HOSTS = ['192.168.0.1', 'localhost', '127.0.0.1']

# 애플리케이션 등록

# 템플릿 설정, DIRS 변경
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],  # 수정
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.message.context_processors.messages',
            ],
        },
    },
]

# 데이터베이스 엔진
# Database
# https://docs.djangoproject.com/en/2.0/ref/settings/#databases
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}

# 타임존 설정
# TIME_ZONE = 'UTC'
TIME_ZONE = 'Asia/Seoul'

# 정적 파일 설정
STATIC_URL = '/static/'

STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static')]  # 추가

# 미디어 관련 사항 지정
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

```
# 한국 시간대만 사용하는 경우
# USE_TZ = True
USE_TZ = False

# 장고 사용 언어
#LANGUAGE_CODE = 'en-us'
LANGUAGE_CODE = 'ko-kr'
```

- models
  - 테이블 정의
  - admin 사이트에 테이블 반영
  - 데이터베이스 변경 사항 반영
  - 테이블 확인
- URLconf
- View
- Templates
  - bootstrap
  - 상속; base.html, home.html, footer.html, ...
- 인증

AWS, Heroku

- AWS S3 서비스 연동을 위한 썸네일용 패키지
  - sorl-thumbnail: S3 서비스와 연동 가능하며 원격 저장소에 대한 쿼리 성능 좋음
  - django-imagekit: S3 서비스와 연동을 위해 일부 SW 변경 필요
  - easy-thumbnails: S3 서비스와 연동되지 않음
- RDS, MySQL DB 연동
  - AWS 사이트에서 MySQL 생성 → 장고 프로그램에서 사용
  - AWS RDS로 MySQL 생성; 마스터 사용자이름, 마스터 암호, 엔드 포인트, 포트 확인
  - 패키지; mysqlclient 설치 → settings.py 파일에서 DATABASES 부분 수정 → migrate 실행 → createsuperuser 실행 → 기존 SQLite3의 데이터 export/import (manage.py dumpdata 및 loaddata)
- S3 서비스 활용(Storage 서버 연동)
  - S3(Simple Storage Service) 버킷 생성
  - IAM(Identity and Access Management) 사용자 생성; IAM메뉴 → 사용자 → 사용자추가 → 사용자이름, 프로그래밍 방식 엑세스 → 기존 정책 직접 연결 → 정책 필터: s3, 정책 목록 : AmazonS3FullAccess → 사용자 자격 증명(액세스 키와 비밀 엑세스키) csv 다운로드
  - boto3, django-storages 패키지 설치
  - settings.py 수정; INSTALLED_APPS에 'storage' 추가 → 파일 끝에 STORAGE관련, AWS 관련 내용 추가
  - urls.py 수정;
  - storage.py 추가; S3Boto3Storage 클래스 상속 받아 정의
  - 정적 파일 모으기(collectstatic 명령) → S3 사이트에서 버킷 내용 확인

```
(djprj)$ python manage.py collectstatic
```

- Elastic Beanstalk 서비스 활용(웹 서버 활용); 브라우저 UI, EB CLI 명령

    - EB 애플리케이션 및 환경 생성; Elastic Beanstalk → 시작하기 → 애플리케이션 이름, 플랫폼, 애플리케이션 코드 → url 확인 후 접속 확인
    - 장고 프로젝트 배포; requirements.txt 생성 → .ebextensions 폴더 만들기 → django.config(변경 가능) 파일 생성 (YAML) → settings.py 수정 → 프로젝트 압축
    - Elastic Beanstalk 메뉴 → 생성한 애플리케이션 이름 → 애플리케이션 버전 → 업로드 → 버전 레이블 입력 → 업로드 → 작업 → 배포 → 환경 선택 → 배포 → 이벤트 페이지 → 에러 발생시 → 환경명/로그/로그 요청/마지막 100줄 → django.config 파일 수정 후 압축/업로드/배포 과정 성공할 때까지 반복 → EB에서 정해준 도메인 URL로 접속 확인
- Heroku 서비스 활용(웹 서버 활용)
    - 프로그램 설치; Heroku CLI, git
    - 장고 프로젝트 배포 준비; gunicorn 패키지 설치 → 가상환경 맞추기 (requirements.txt) → Procfile, runtime.txt, .gitignore 생성 → settings.py 수정
    - Heroku CLI; git init → heroku login → heroku create 앱이름 → 비밀 데이터를 앱의 환경 변수로 등록 heroku config:set DJANGO_SECRET_KEY, DATABASE_NAME,.... → git add -A → git commit -m “..” → git push heroku master → heroku run python manage.py migrate → heroku run python manage.py createsuperuser → heroku open → url 확인 후 브라우저에서 실행 확인

- EC2 서비스 참고 사항

^ ^ 장점 ^ 단점 ^

EC2(Elastic Comput Cloud) 서비스	구성 자유도 높음, 비용 낮음	작업 난이도 높음
Heroku 서비스	작업 난이도 낮음	구성 자유도 낮음, 비용 높음

- ref; IaaS(Infra as a Service), Paas(Platform as a Service)

From:
https://theta5912.net/ - **reth**

Permanent link:
https://theta5912.net/doku.php?id=public:computer:django&rev=1628008110

Last update: **2021/08/04 01:28**