

파이썬 기초 with Django

■ 프레임워크

백 엔드 혹은 서버 사이드 프로그래밍의 경우 언어 자체로 웹 프로그래밍을 하는 것은 꽤나 반복적으로 지루한 작업을 많이 해야 하는 데다가 빈도와 양이 많다.

이런 비효율을 해결하기 위해 만들어진 도구

(그래서 python, Ruby, php, java, C# 등의 언어로 웹 서비스를 만드는 경우 언어만 사용하지 않고 프레임워크를 사용하게 되는 것)

■ java - Spring / js - Node.js / Ruby - Ruby on Rails / php - Laravel

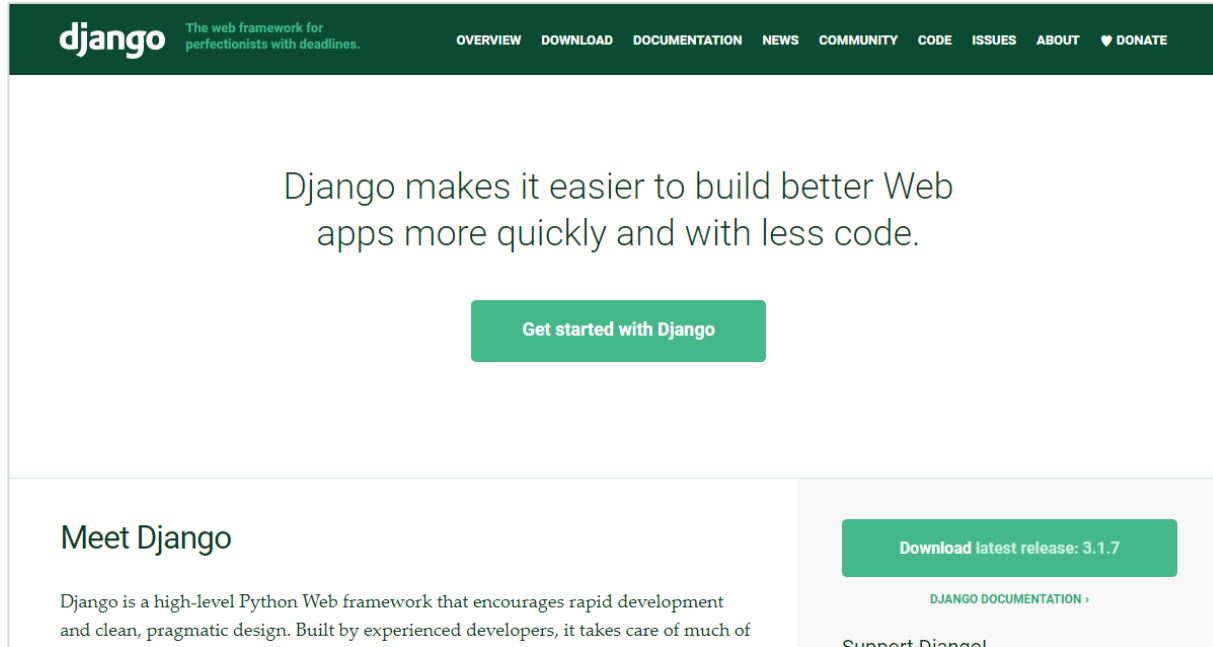
■ 풀 스택 프레임워크

- 웹 서비스를 만드는데 필요한 다양한 기능(데이터베이스, 인증, 템플릿 엔진 등)을 모두 포함하고 한꺼번에 설치하는 형태
- 프레임워크를 설치 후 바로 기본적인 웹 서비스를 할 수 있을 정도로 편리
- 단점 : 기본 기능들이 하나로 뭉쳐 있어 커스터마이징이 비교적 어렵고, 함께 설치되는 코드의 양이 많기 때문에 상대적으로 느리다

■ 마이크로 프레임워크

- 적은 코드가 초기에 설치되고 많은 기능을 갖고 있지 않기 때문에 가볍고 빠르며, 커스터마이징 하기 좋다
- 단점 : 기능 개발에 비교적 시간이 오래 걸릴 수 있다

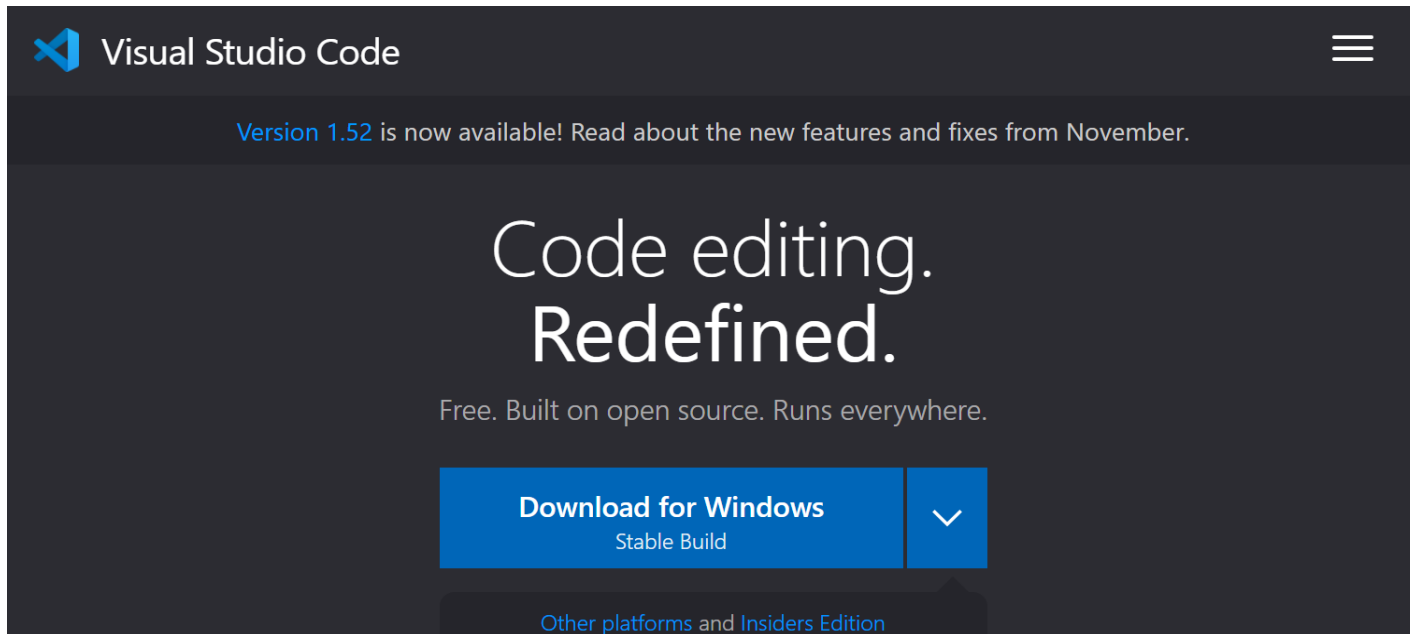
- The web framework for perfectionists with deadlines. (<https://www.djangoproject.com/>)



■ Django - 장고

- 배우기 쉽고, 활용 범위가 넓다.
파이썬 기반 다양한 라이브러리 활용이 가능하다.
- 관리자페이지를 기본으로 제공한다.
프로젝트를 생성하는 순간 기본 관리자페이지를 자동으로 생성하여 현재 프로젝트의 디비 구조 파악을 할 수 있게 한다.
- 기본 보안 기능이 설정되어 있다.
SQL Injection, XXS (cross-site scripting), CSRF (cross-site request forgery), Clickjacking 과 같은 보안 공격을 기본으로 막아준다.
- 여러 기능이 준비되어 있다
웹 프로그램 개발을 위한 도구와 기능이 대부분 준비되어 있다.

- Visual Studio Code (<https://blog.naver.com/emilia96/222180287353>)



장고 기본 명령들

- **django-admin startproject**

장고 프로젝트를 만드는 명령. 웹 서비스를 만들 때마다 한번 실행.

- **startapp**: 프로젝트에 기능 단위인 앱을 새로 만들 때 사용

- **makemigrations**

어플리케이션 변경사항 추적해 DB에 적용할 내용 정리. 앱 안에 있는 model의 변경 사항이 있을 때 주로 사용

- **sqlmigrate**

실행할 SQL명령문 출력. 어떤 명령문을 실행할지 확인할 때 사용, 튜닝이 안된 쿼리나 슬로우 쿼리 여부 확인

- **migrate**: 실제 변경사항을 DB에 반영

- **showmigrations**: 프로젝트의 DB 변경사항 목록과 상태를 출력

- **runserver**: 테스트 서버 실행

- **dumpdata**: 현재 DB 내용 백업 시 사용

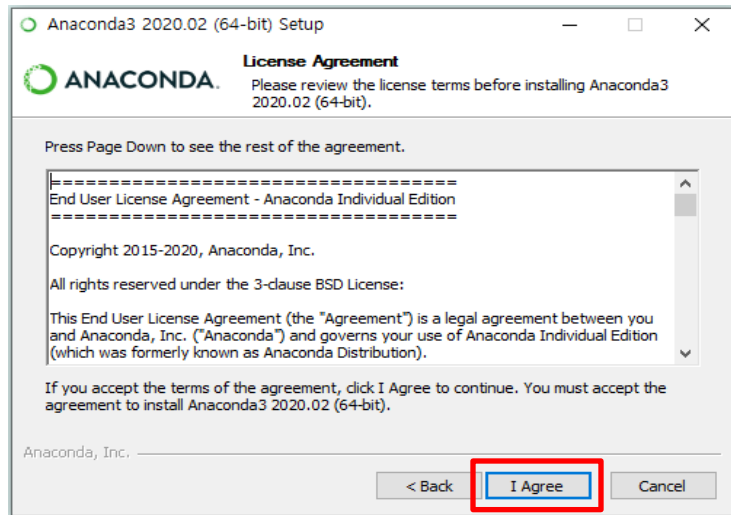
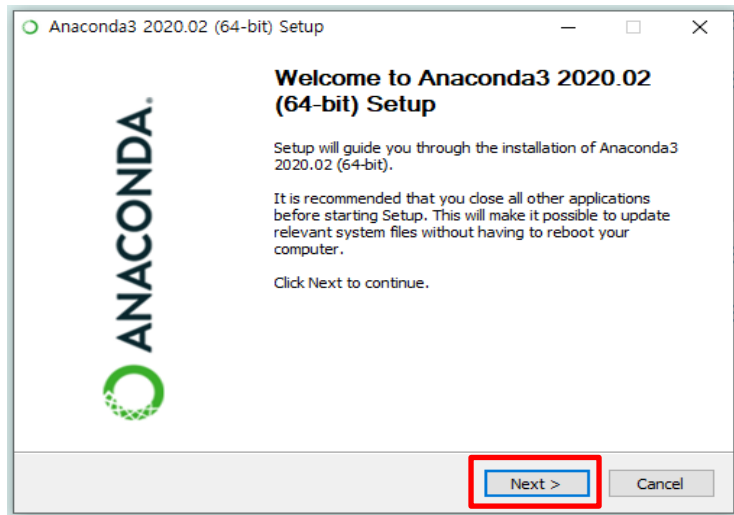
- **loaddata**: 백업 파일에서 DB로 내용을 복구할 때 사용

장고 기본 명령들

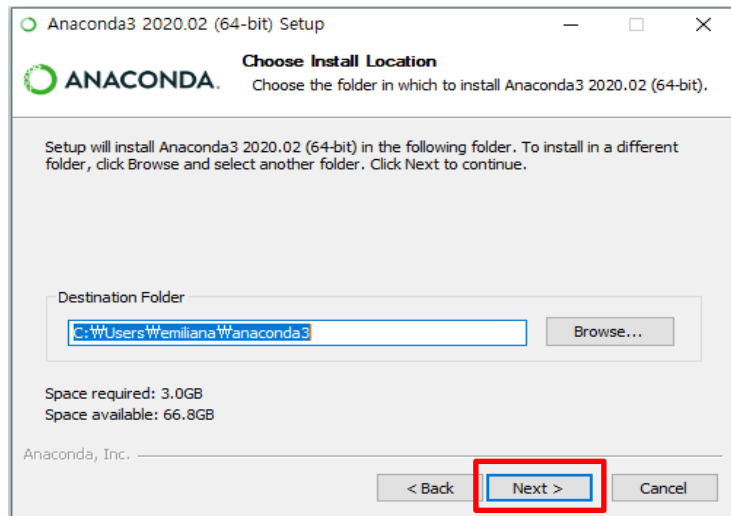
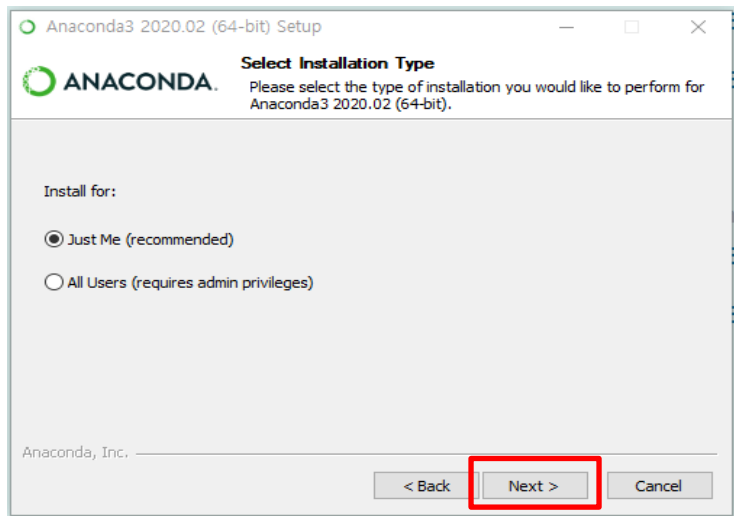
- **flush** : DB테이블은 그대로 두고 테이블의 내용만 전부 삭제
- **shell** : 장고 쉘 실행. 작성한 모델 등을 불러와 실제 테스트 가능
- **dbshell** : DB에 직접 접근할 수 있는 쉘을 실행. SQL구문을 이용해 직접 수정하고 싶을 때 사용
- **createsuperuser** : 관리자 계정 생성
- **changepassword** : 계정 비밀번호 변경

개발 환경 설정 - 아나콘다

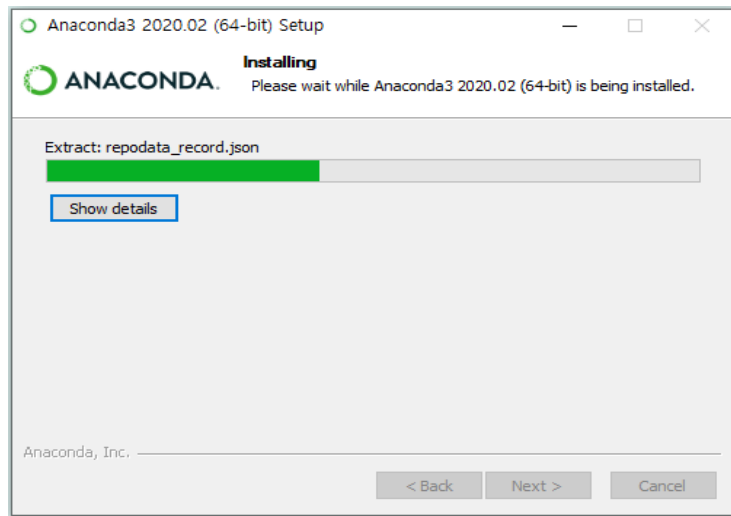
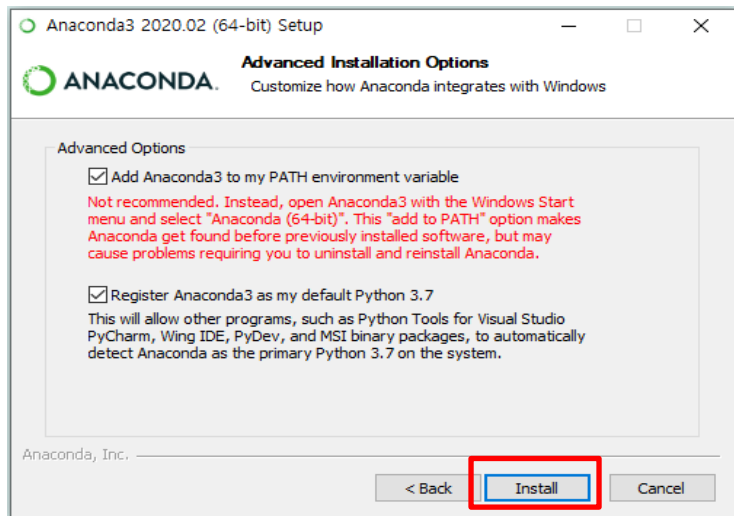
■ 아나콘다 설치



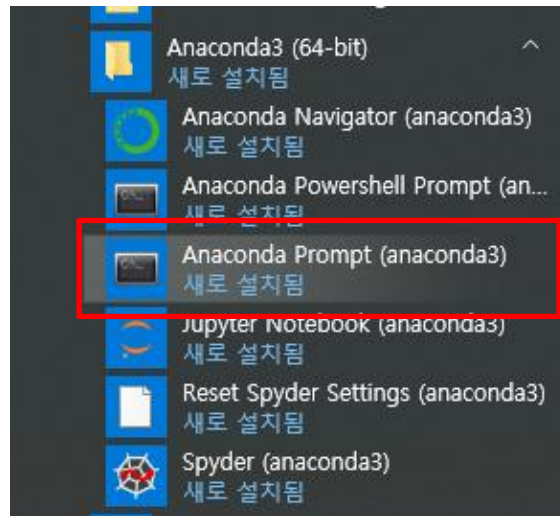
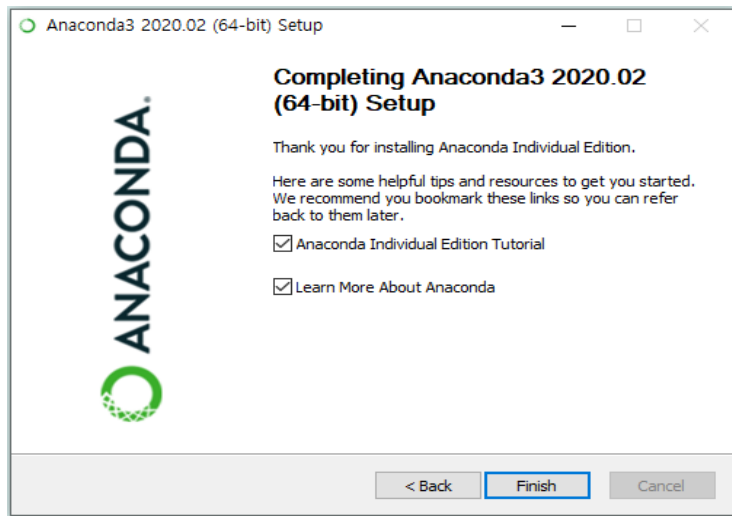
■ 아나콘다 설치



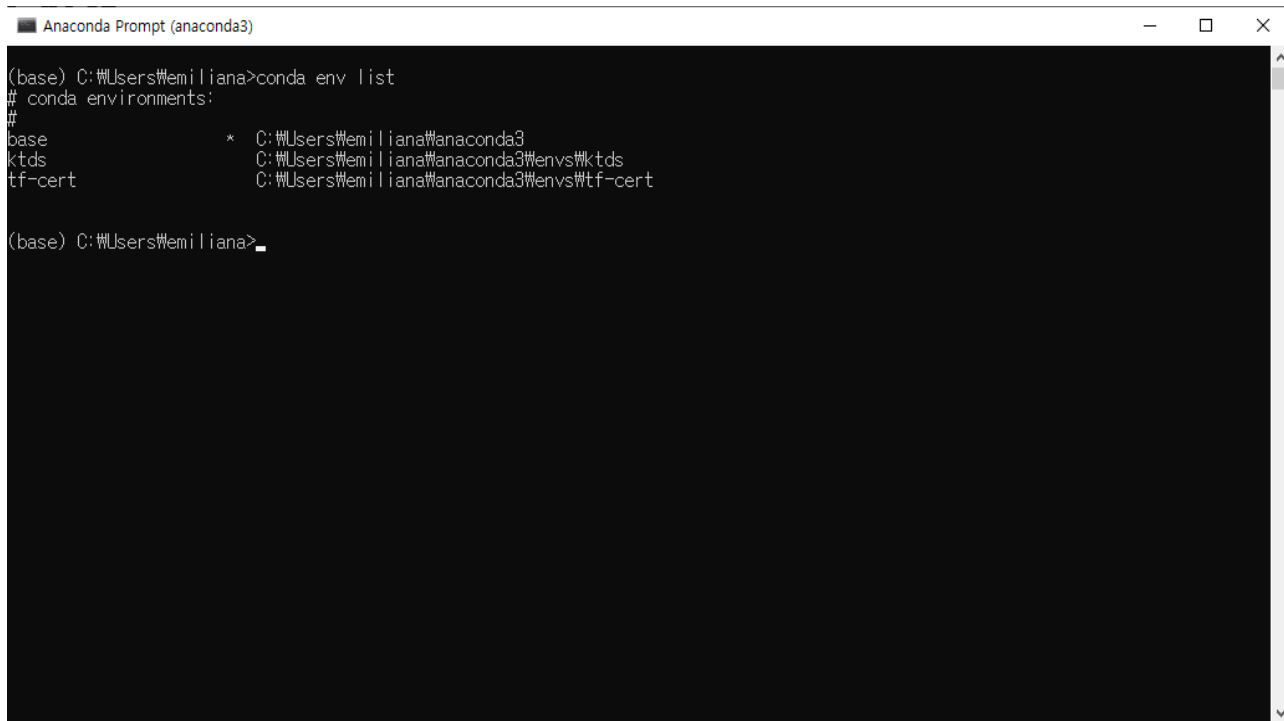
■ 아나콘다 설치



■ 아나콘다 설치



■ 아나콘다 가상환경 목록 확인



```
Anaconda Prompt (anaconda3)
(base) C:\Users\Wemiliana>conda env list
# conda environments:
#
base                * C:\Users\Wemiliana\anaconda3
ktds                 C:\Users\Wemiliana\anaconda3\envs\ktds
tf-cert             C:\Users\Wemiliana\anaconda3\envs\tf-cert

(base) C:\Users\Wemiliana>
```

■ 아나콘다 최신 버전 업데이트 : conda update -n base -c defaults conda

```
선택 Anaconda Prompt (anaconda3)
(base) C:\Users\Wemiliana>conda update -n base -c defaults conda
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\Wemiliana\anaconda3

  added / updated specs:
    - conda

The following packages will be downloaded:

package-----|-----build-----|-----
conda-4.8.3-----|py37_0-----|2.8 MB
conda-package-handling-1.6.1|py37h62dcd97_0|612 KB
-----|-----|-----
Total: 3.4 MB

The following packages will be UPDATED:

conda 4.8.2-py37_0 --> 4.8.3-py37_0
conda-package-handling 1.6.0-py37h62dcd97_0 --> 1.6.1-py37h62dcd97_0

Proceed ([y]/n)? y

Downloading and Extracting Packages
conda-4.8.3 | 2.8 MB | ##### | 100%
conda-package-handling | 612 KB | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

(base) C:\Users\Wemiliana>
```


■ 새로운 가상환경 설치 : conda create -n mysite python=3.8

```
Anaconda Prompt (anaconda3) - conda create -n mysite python=3.8

(base) C:\Users\memiliana>conda create -n mysite python=3.8
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\memiliana\Anaconda3\envs\mysite

added / updated specs:
- python=3.8

The following packages will be downloaded:

package | build | size
-----|-----|-----
ca-certificates-2021.1.19 | haa95532_0 | 122 KB
certifi-2020.12.5 | py38haa95532_0 | 141 KB
openssl-1.1.1j | h2b6ff1b_0 | 4.8 MB
pip-21.0.1 | py38haa95532_0 | 1.8 MB
python-3.8.8 | hdbf38c2_4 | 15.9 MB
setuptools-52.0.0 | py38haa95532_0 | 726 KB
sqlite-3.33.0 | h2a8f88b_0 | 809 KB
vc-14.2 | h21ff451_1 | 8 KB
vs2015_runtime-14.27.29016 | h5e88377_2 | 1007 KB
wheel-0.36.2 | pyhd9eb1b0_0 | 33 KB
wincertstore-0.2 | py38_0 | 15 KB

Total: 25.3 MB

The following NEW packages will be INSTALLED:

ca-certificates pkgs/main/win-64::ca-certificates-2021.1.19-haa95532_0
certifi pkgs/main/win-64::certifi-2020.12.5-py38haa95532_0
openssl pkgs/main/win-64::openssl-1.1.1j-h2b6ff1b_0
pip pkgs/main/win-64::pip-21.0.1-py38haa95532_0
python pkgs/main/win-64::python-3.8.8-hdbf38c2_4
setuptools pkgs/main/win-64::setuptools-52.0.0-py38haa95532_0
sqlite pkgs/main/win-64::sqlite-3.33.0-h2a8f88b_0
vc pkgs/main/win-64::vc-14.2-h21ff451_1
vs2015_runtime pkgs/main/win-64::vs2015_runtime-14.27.29016-h5e88377_2
wheel pkgs/main/win-64::wheel-0.36.2-pyhd9eb1b0_0
wincertstore pkgs/main/win-64::wincertstore-0.2-py38_0
zlib pkgs/main/win-64::zlib-1.2.11-h82c0d37_4

Proceed ([y]/n)?
```

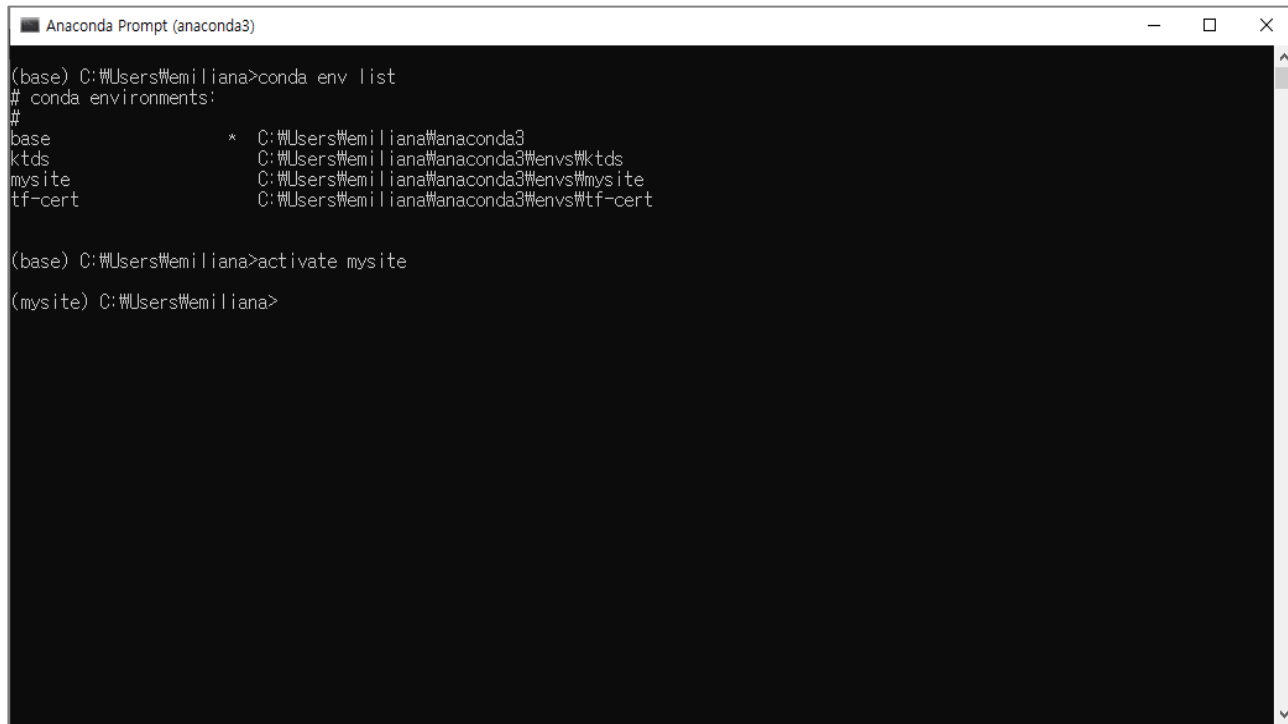
```
Anaconda Prompt (anaconda3)

done

# To activate this environment, use
#
#     $ conda activate mysite
#
# To deactivate an active environment, use
#
#     $ conda deactivate
#

(base) C:\Users\memiliana>
```

■ 가상환경 확인



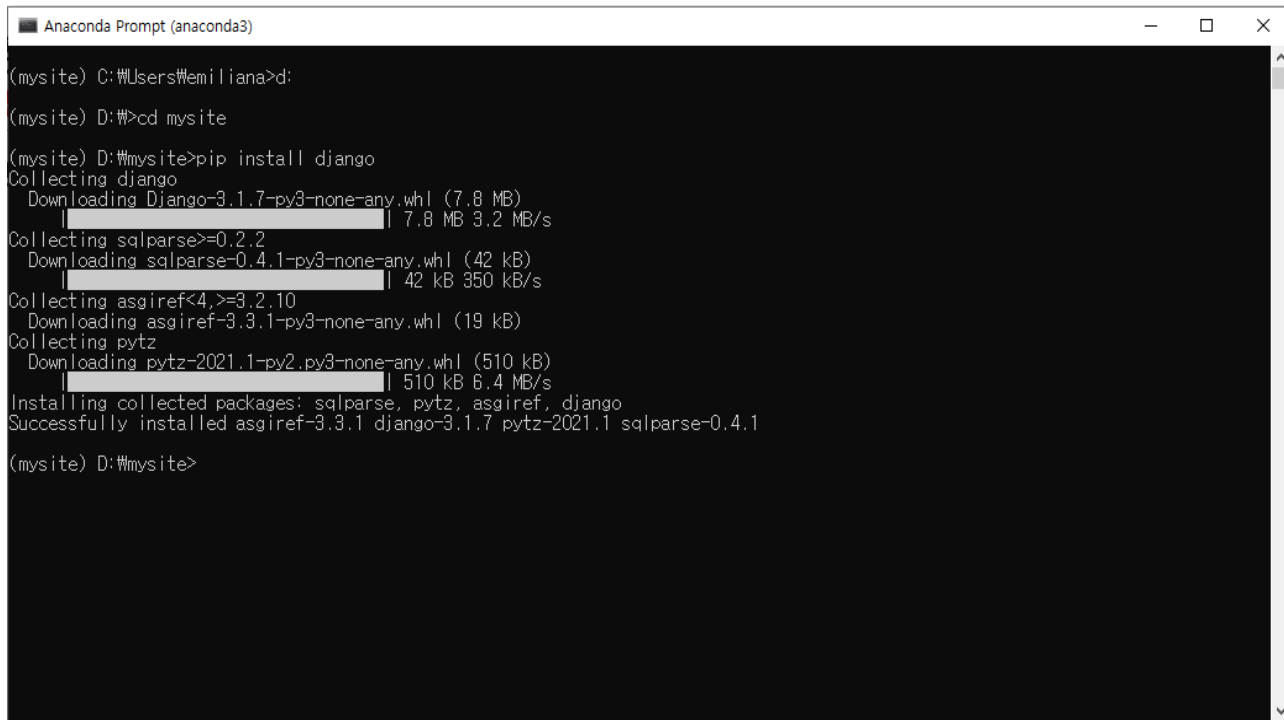
```
Anaconda Prompt (anaconda3)

(base) C:\Users\Wemiliana>conda env list
# conda environments:
#
base                * C:\Users\Wemiliana\anaconda3
ktds                 C:\Users\Wemiliana\anaconda3\envs\ktds
mysite               C:\Users\Wemiliana\anaconda3\envs\mysite
tf-cert              C:\Users\Wemiliana\anaconda3\envs\tf-cert

(base) C:\Users\Wemiliana>activate mysite

(mysite) C:\Users\Wemiliana>
```

■ Django 설치 : pip install django



```
Anaconda Prompt (anaconda3)

(mysite) C:\Users\emiliana>d:

(mysite) D:\>cd mysite

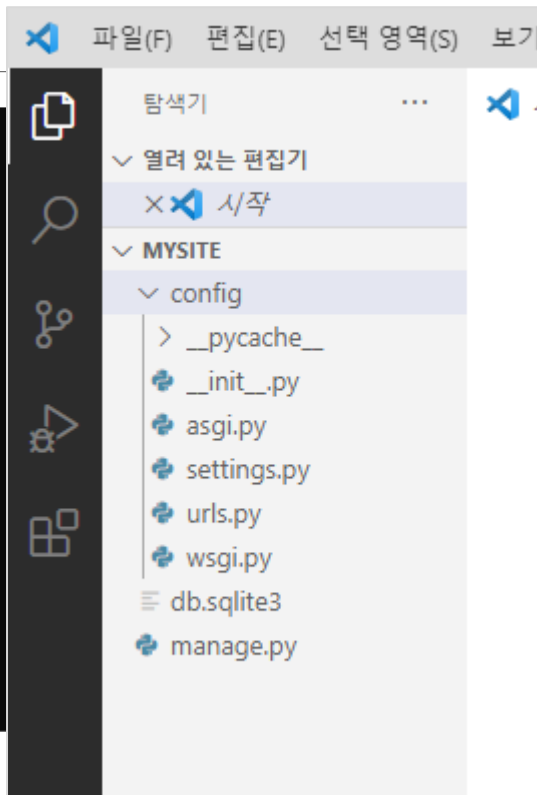
(mysite) D:\mysite>pip install django
Collecting django
  Downloading Django-3.1.7-py3-none-any.whl (7.8 MB)
    | 7.8 MB 3.2 MB/s
Collecting sqlparse>=0.2.2
  Downloading sqlparse-0.4.1-py3-none-any.whl (42 kB)
    | 42 kB 350 kB/s
Collecting asgiref<4,>=3.2.10
  Downloading asgiref-3.3.1-py3-none-any.whl (19 kB)
Collecting pytz
  Downloading pytz-2021.1-py2.py3-none-any.whl (510 kB)
    | 510 kB 6.4 MB/s
Installing collected packages: sqlparse, pytz, asgiref, django
Successfully installed asgiref-3.3.1 django-3.1.7 pytz-2021.1 sqlparse-0.4.1

(mysite) D:\mysite>
```

장고 프로젝트 생성 - Django-admin startproject config .

Anaconda Prompt (anaconda3)

```
(mysite) D:\mysite>django-admin startproject config .  
(mysite) D:\mysite>_
```



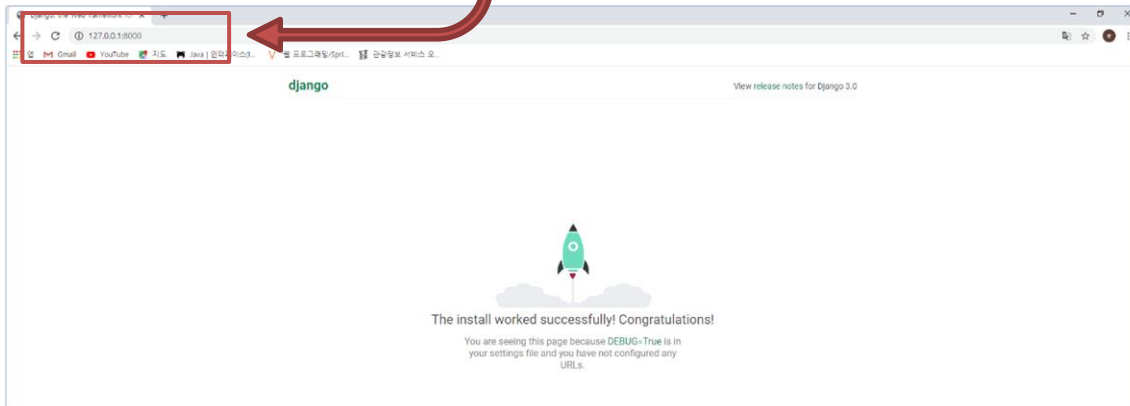
python manage.py runserver

```
Anaconda Prompt (anaconda3) - python manage.py runserver

(mysite) D:\mysite>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
March 08, 2021 - 00:06:15
Django version 3.1.7, using settings 'config.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```



■ 슈퍼 유저 생성하기 : `python manage.py createsuperuser`

```
(mysite) D:\mysite>python manage.py createsuperuser
Username (leave blank to use 'emiliana'): admin
Email address: emily.kyungah.kim@gmail.com
Password:
Password (again):
Superuser created successfully.

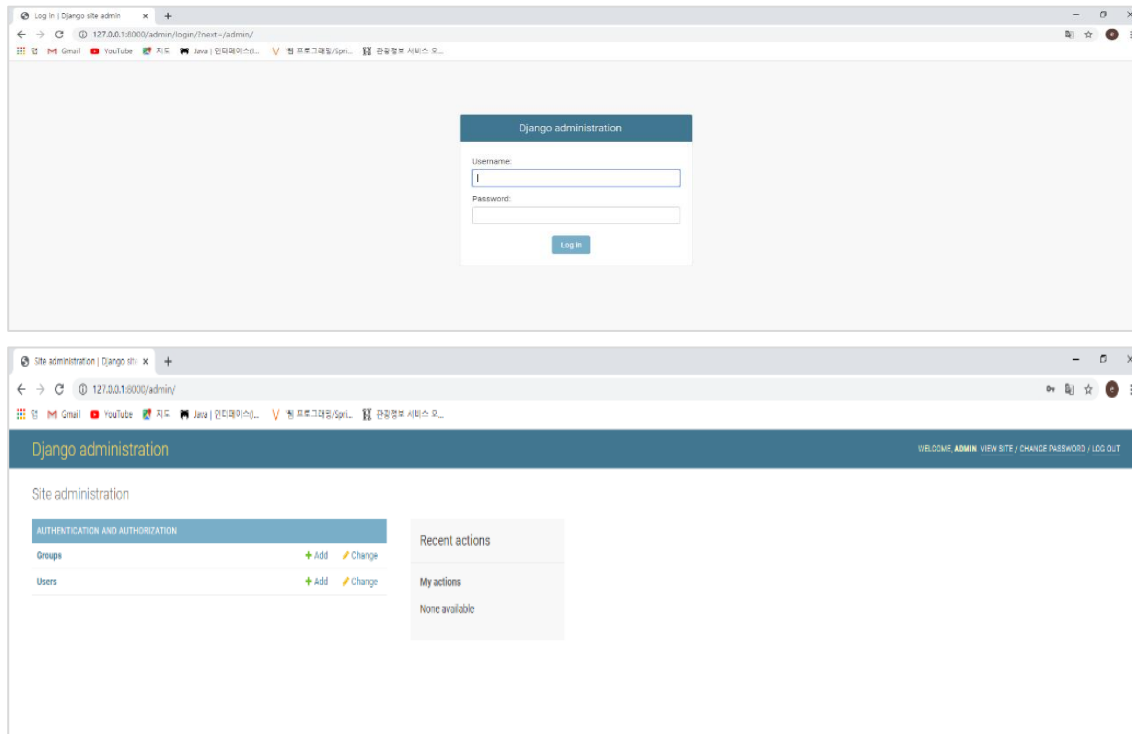
(mysite) D:\mysite>_
```

비밀번호 입력 시 영문과 숫자를 섞고 최소8자리로 해야함.

사이트 확인

kt ds University

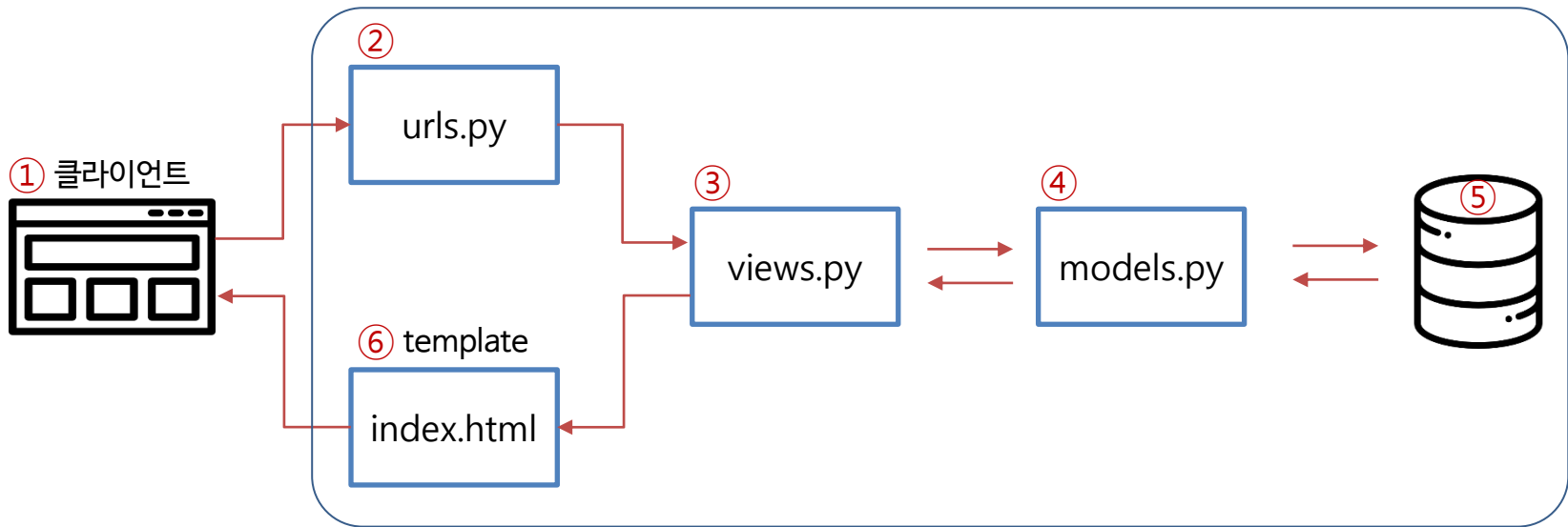
■ <http://127.0.0.1:8000/admin/>



MTV

■ MVC와 MTV

- MVC : 웹 프로그래밍에서 자주 사용되는 디자인 패턴 (Model-View-Controller)
- MVT : 장고 디자인 패턴 (Model-Template-View)
- Model:Model - View:Template - Controller:View



■ models.py

장고에서는 models.py를 통해 DB의 명세를 관리.

모델은 클래스로 만드는데 클래스의 이름이 테이블 이름이 되고 클래스의 속성들이 컬럼이 됨

모델을 이용해 DB의 종류에 상관없이 그리고 프로그래머가 SQL을 모르더라도 DB를 편하게 이용할 수 있도록 ORM 사용.

모델을 이용하면 ORM내부에서 자동으로 데이터베이스에 할 일을 전달하고 결과를 반환

SQL문이나 API사용법을 몰라도 웹 어플리케이션 작성이 가능.

■ admin.py

모델이 제대로 만들어졌는지 확인을 하고 싶다면 장고 기본관리자 페이지를 사용할 수 있음. 모델을 관리자페이지에서 확인하거나 기능을 추가하고 싶다면 admin.py에 써주면 됨

■ views.py

글쓰기, 글보기 등 페이지들을 하나하나 만들 때 views.py에 프로그래밍

이 뷰는 클래스형 뷰와 함수형 뷰 두가지가 있고, 대부분의 프로그래밍 작업은 뷰를 만들고 수정하는 것으로 이루어짐

■ urls.py

어떤 url을 이용해 어떤 view를 동작 시킬지를 결정할 때 이 내용을 urls.py에 기록

■ templates

html이 들어 있는 파일.

특정 폴더 안에 템플릿 파일들을 모아두고 싶다면 파일 위치를 settings.py에 설정해 줘야 함

개발환경

■ config 폴더

프로젝트 설정 파일과 웹 서비스 실행을 위한 파일들

이 폴더 이름은 django-admin startproject 명령을 사용해 프로젝트를 만들 때 정해진 것이며 config라는 이름을 사용할 필요는 없다.

- `__init__.py` : 파이썬 2.x 대 버전과의 호환을 위해 만들어진 비어 있는 파일. 파이썬 3.x대에서는 불필요하지만 계속 생성됨. 지워져도 프로젝트에 영향을 끼치지 않음.
- `settings.py` : 프로젝트 설정에 관한 다양한 내용이 있음.
- `urls.py` : 특정 기능을 수행하기 위해 접속하는 주소를 url이라 하고 이를 설정해 주는데 그 설정을 기록. 한 프로젝트 안에 여러 개의 urls 파일을 만들 것이며, config 폴더 안에 있는 urls 파일이 최초 기준 url 파일. 기준 url파일은 settings.py에서 변경 가능.
- `wsgi.py` : 웹 서비스를 실행하기 위한 wsgi 관련 내용이 있음. 특별히 변경할 일 거의 없음.

■ db.sqlite3

SQLite3 DB파일. SQLite DB를 사용할 경우 임의로 삭제하거나 위치이동 금지.
다른 DB로 변경하는 경우 필요 없는 파일.

■ manage.py

장고의 다양한 명령어를 실행하기 위한 파일. 임의로 변경하지 않아야 함.

■ config/settings.py

settings.py 는 프로젝트에 관련된 다양한 설정이 모두 들어있음. 장고 기본 값으로 사용하게 됨.

- BASE_DIR : 프로젝트 루트 폴더.
- SECRET_KEY : 다양한 보안을 위해 사용
- DEBUG : 디버그 모드 설정. true일 경우 오류 메시지 확인 가능. 실제 배포 시에는 false로 변경
- ALLOWED_HOSTS : 현재 서비스의 호스트 설정. 배포 시 실제 도메인 기록
- INSTALLED_APPS : 장고는 다양한 앱의 결합으로 만들어짐. 현재 프로젝트에서 사용하는 앱의 목록을 기록하고 관리.
- MIDDLEWARE : 장고의 모든 요청/응답 메시지 사이에 실행되는 특수한 프레임워크들
- ROOT_URLCONF : 기준이 되는 urls.py 파일의 경로 설정
- TEMPLATES : 장고에서 사용하는 템플릿 시스템 설정
- WSGI_APPLICATION : 실행을 위한 WSGI 어플리케이션 설정
- DATABASE : DB설정
- AUTH_PASSWORD_VALIDATORS : 비밀번호 검증 설정
- LANGUAGE_CODE 등 : 이하 내용은 다국어에 관한 설정들.

■ config/wsgi.py

웹서버와 장고 애플리케이션 사이에 통신 역할을 담당

웹 서버 프로그램과 장고 웹 애플리케이션 사이에서 미들웨어처럼 동작하면서 웹 서버는 요청이 있을 경우 정보와 콜백함수를 WSGI(Web Server Gateway Interface)에 전달.

그럼 이 정보를 해석하여 장고 웹 어플리케이션 전달.

장고 웹 어플리케이션은 파이썬 스크립트를 이용해 정보를 처리하고 끝낸 결과를 WSGI에 다시 전달.

그럼 이 정보를 콜백함수를 이용해 웹 서버에 다시 전달하는 방식으로 서버, WSGI, 장고 웹 어플리케이션이 상호작용하며 동작하게 됨.

이를 위해 wsgi.py 파일을 이용해 어플리케이션 구동

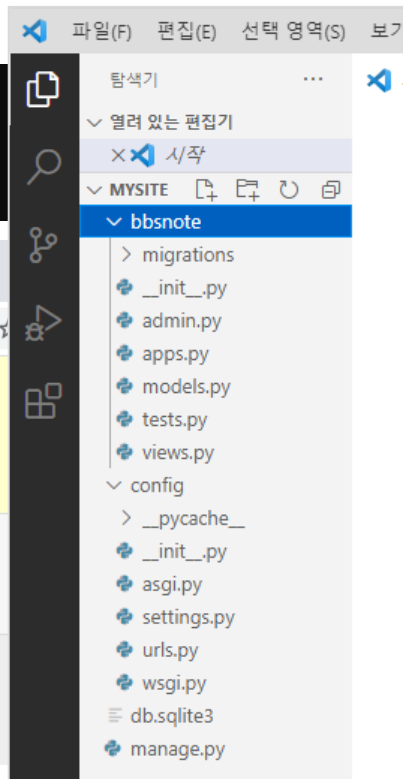
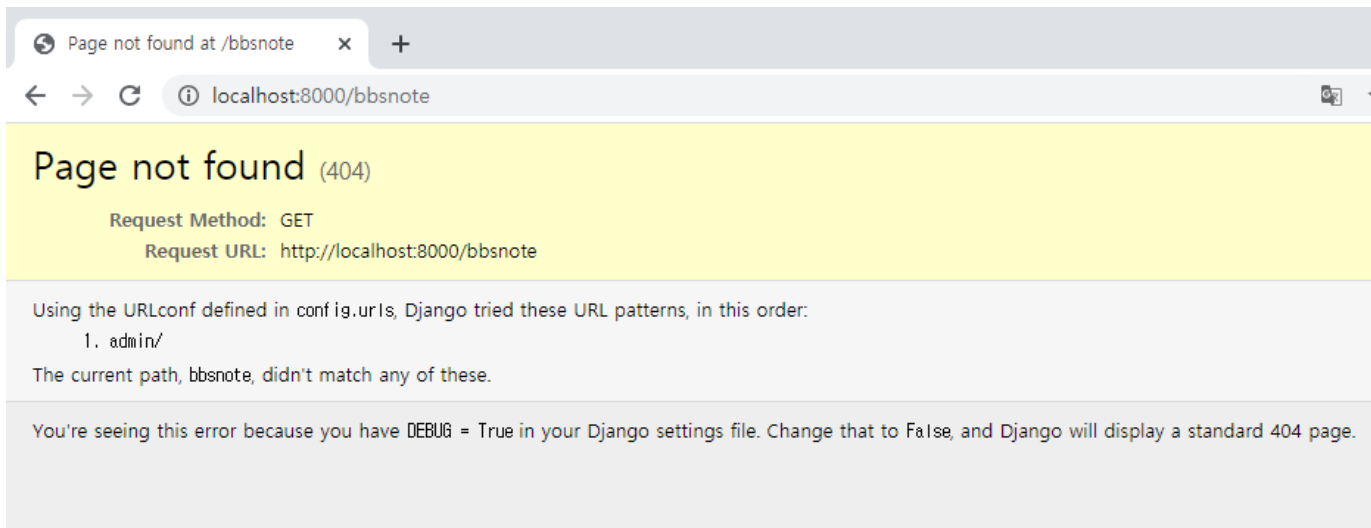
게시판노트 앱 만들기

게시판노트 앱 만들기

kt ds University

■ 기능 추가를 위해 앱을 생성

```
(mysite) D:\mysite>python manage.py startapp bbsnote  
(mysite) D:\mysite>
```



■ urls.py 수정하기 - URL 매핑

config/urls.py

```
from django.contrib import admin
from django.urls import path
from bbsnote import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('bbsnote/', views.index),
]
```

■ views.py 작성

config/urls.py

```
from django.shortcuts import render
from django.http import HttpResponse

# Create your views here.
def index(request):
    return HttpResponse("bbsnote에 오신것을 환영합니다!");
```

- URL 분리 - 프로젝트 짜임새를 고려하여 URL 매핑 관리를 한다.

config/urls.py

```
from django.contrib import admin
from django.urls import path, include
from bbsnote import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('bbsnote/', include('bbsnote.urls')),
]
```

bbsnote/urls.py

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index),
]
```

게시판노트 앱 만들기

■ 모델로 DB 관리하기

```
(mysite) D:\mysite>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
```

```
System check identified no issues (0 silenced).
```

```
You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
```

```
Run 'python manage.py migrate' to apply them.
```

```
March 14, 2021 - 16:02:49
```

```
Django version 3.1.7, using settings 'config.settings'
```

```
Starting development server at http://127.0.0.1:8000/
```

```
Quit the server with CTRL-BREAK.
```

```
[14/Mar/2021 16:02:54] "GET /bbsnote/ HTTP/1.1" 200 40
```

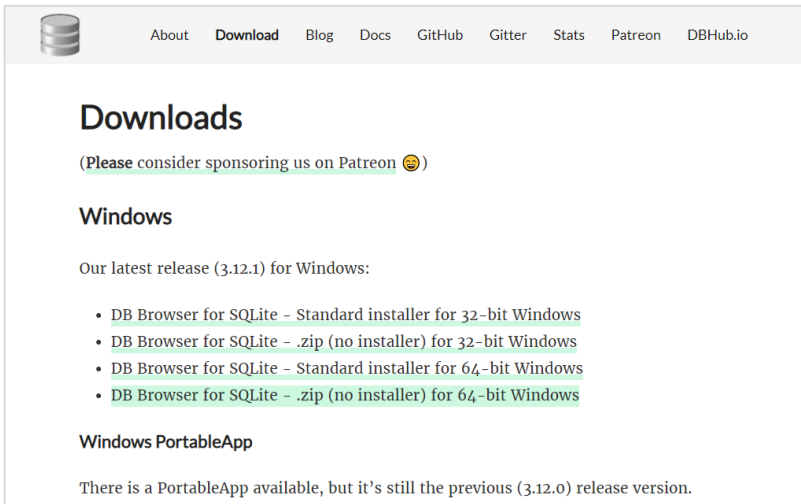
- django는 모델로 데이터 관리가 가능하다. SQL을 사용하지 않고도 모델만으로 데이터를 저장,조회 등 관리할 수 있다.
- 이를 위해 migrate 명령어로 앱들이 필요로 하는 테이블을 생성해주어야 한다. `python manage.py migrate`

```
(mysite) D:\mysite>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
```

게시판노트 앱 만들기

■ DB Browser for SQLite로 생성된 테이블 살펴보기

- django로 작업할 때에 ORM으로 작업이 가능하기 때문에 SQL을 알지 못해도 사용하는 데에 문제 없다.
- 기본 db인 SQLite의 내용을 살펴보고 싶다면, GUI도구를 다운로드 받아 확인할 수 있다. (<https://sqlitebrowser.org/dl/>)



The screenshot shows the official website for DB Browser for SQLite. The navigation bar includes links for About, Download, Blog, Docs, GitHub, Gitter, Stats, Patreon, and DBHub.io. The main heading is "Downloads", followed by a note to consider sponsoring on Patreon. Under the "Windows" section, it states the latest release (3.12.1) and lists four download options: Standard installer for 32-bit Windows, .zip (no installer) for 32-bit Windows, Standard installer for 64-bit Windows, and .zip (no installer) for 64-bit Windows. A "Windows PortableApp" section mentions a previous version (3.12.0) is available.

Downloads

(Please consider sponsoring us on Patreon 🙏)

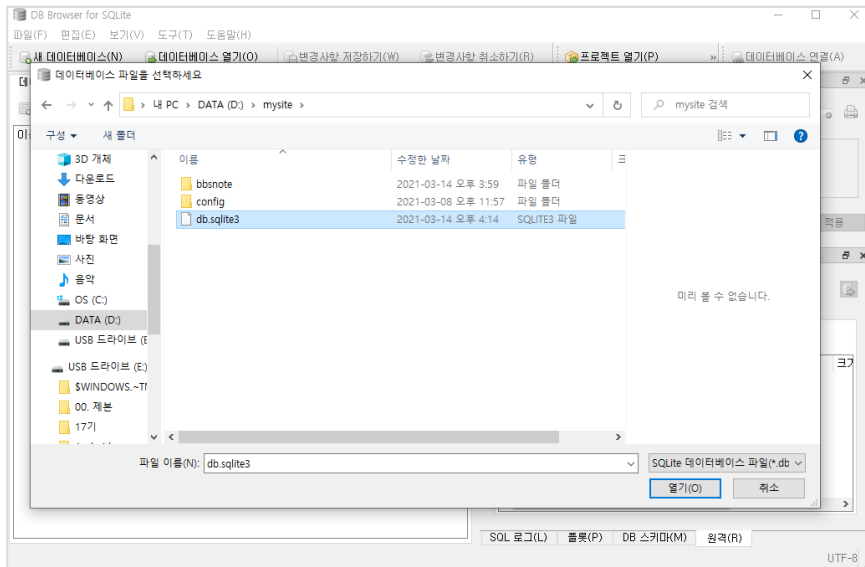
Windows

Our latest release (3.12.1) for Windows:

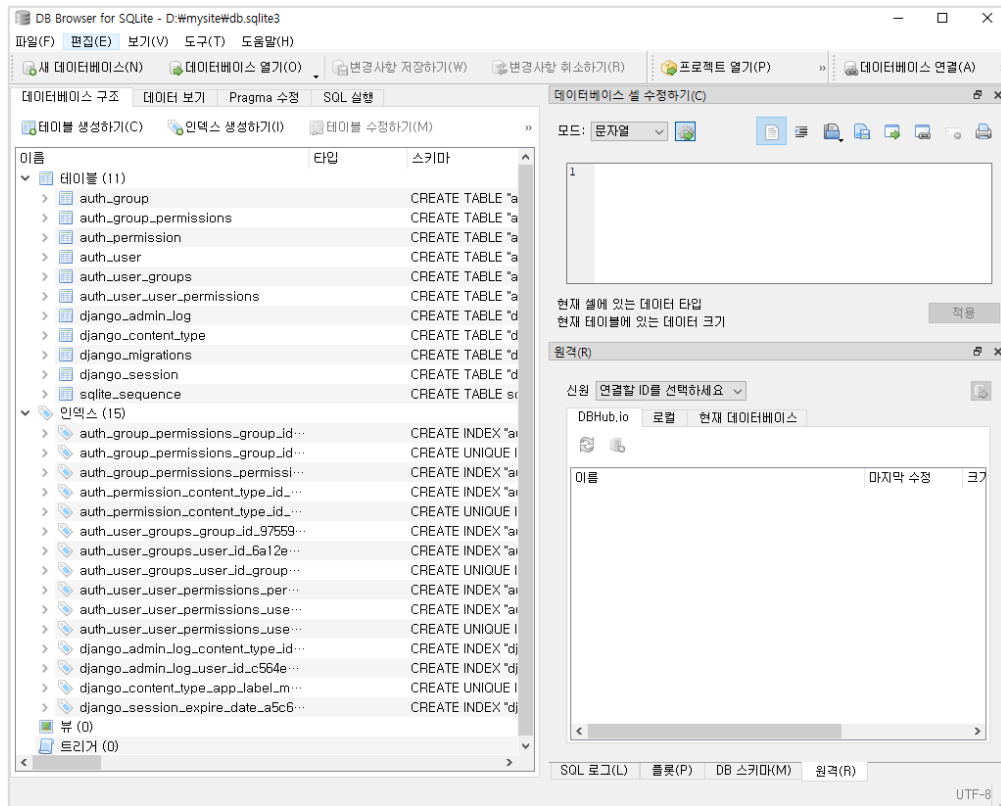
- DB Browser for SQLite - Standard installer for 32-bit Windows
- DB Browser for SQLite - .zip (no installer) for 32-bit Windows
- DB Browser for SQLite - Standard installer for 64-bit Windows
- DB Browser for SQLite - .zip (no installer) for 64-bit Windows

Windows PortableApp

There is a PortableApp available, but it's still the previous (3.12.0) release version.



DB Browser for SQLite로 생성된 테이블 살펴보기



■ 게시판 모델 만들기

- 게시판 본문과 댓글 작성을 위한 모델을 구성한다

bbsnote/models.py

```
from django.db import models

# Create your models here.
class Board(models.Model): #게시글모델
    subject = models.CharField(max_length=200)
    content = models.TextField()
    create_date = models.DateTimeField()

class Answer(models.Model): #댓글모델
    board = models.ForeignKey(Board, on_delete=models.CASCADE)
    content = models.TextField()
    create_date = models.DateTimeField()
```

- <https://docs.djangoproject.com/en/3.0/ref/models/fields/#filed-types>

■ 게시판 모델 만들기

- 작성된 모델이 테이블로 생성되어 등록되려면 config/settings.py 에서 bbsnote 앱을 추가해야 한다.

```
...  
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'bbsnote.apps.BbsnoteConfig',  
]  
...
```

- bbsnote/apps.py에 BbsnoteConfig 자동 등록된 클래스가 있는 것을 확인할 수 있다.

게시판노트 앱 만들기

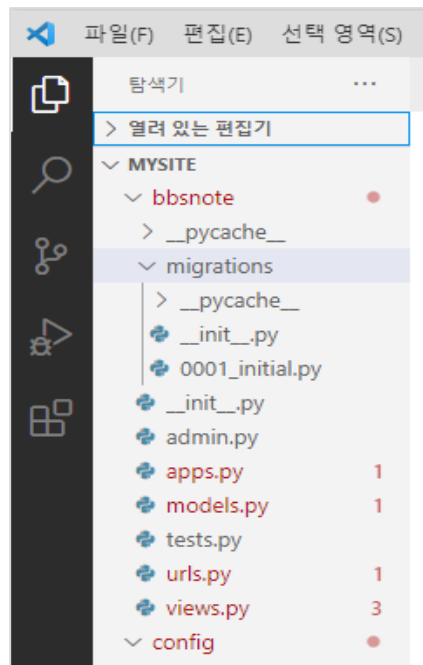
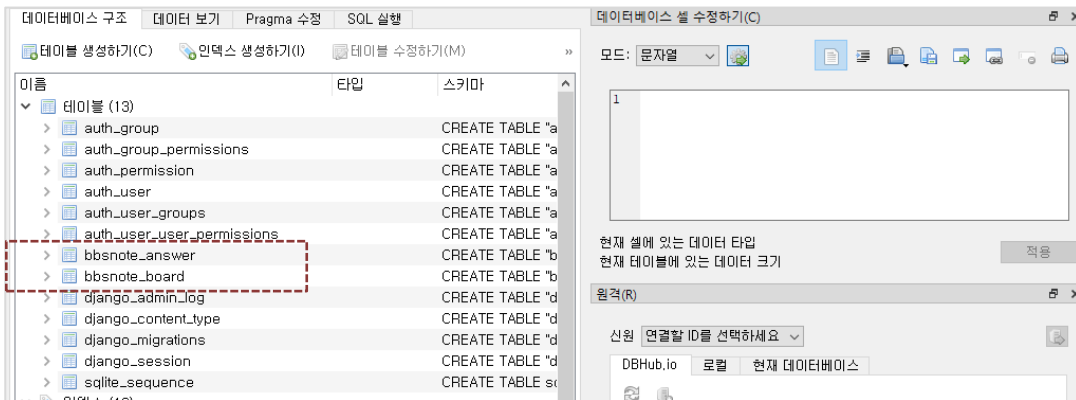
kt ds University

■ 게시판 모델 만들기

```
(mysite) D:\mysite>python manage.py makemigrations
Migrations for 'bbsnote':
  bbsnote\migrations\0001_initial.py
    - Create model Board
    - Create model Answer

(mysite) D:\mysite>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, bbsnote, contenttypes, sessions
Running migrations:
  Applying bbsnote.0001_initial... OK
```

- makemigrations는 장고가 테이블 작업을 수행하기 위한 파일들을 생성하고, 실제 테이블 생성은 migrate가 한다.



게시판노트 앱 만들기

■ 관리자 페이지에서 모델 관리하기

- 해당 모델을 확인하기 위해 admin.py에 등록한다.

bbsnote/admin.py

```
from django.contrib import admin
from .models import Board

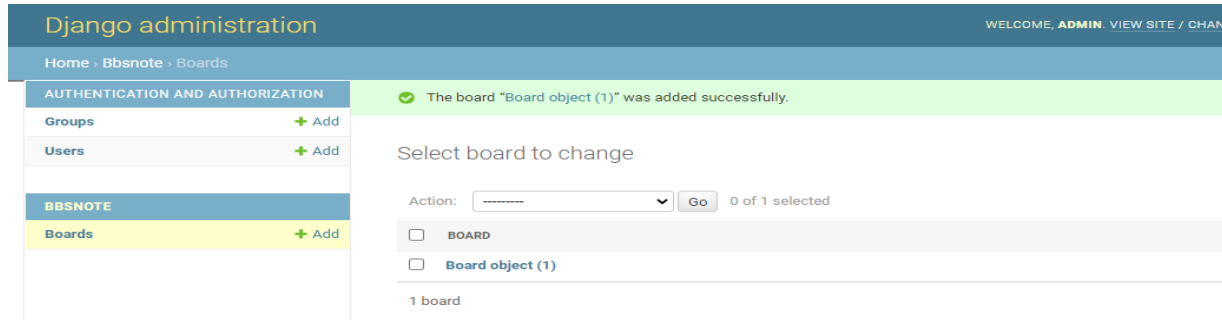
# Register your models here.
admin.site.register(Board)
```

The screenshot shows the Django administration interface. At the top, there's a header bar with 'Django administration' on the left and 'WELCOME, ADMIN. VIEW SITE / CH...' on the right. Below the header, the 'Site administration' section is visible. It contains three main sections: 'AUTHENTICATION AND AUTHORIZATION' with 'Groups' and 'Users' links, each with '+ Add' and 'Change' buttons; 'BBSNOTE' with a 'Boards' link, also with '+ Add' and 'Change' buttons. The 'Boards' link and its associated buttons are highlighted with a red dashed box. To the right of the 'Site administration' section, there's a 'Recent actions' section with a 'My actions' link and the text 'None available'.

게시판노트 앱 만들기

■ 관리자 페이지에서 모델 관리하기

- makemigrations는 장고가 테이블 작업을 수행하기 위한 파일들을 생성하고, 실제 테이블 생성은 migrate가 한다.



bbsnote/models.py

```
...  
class Board(models.Model): #게시글모델  
    subject = models.CharField(max_length=200)  
    content = models.TextField()  
    create_date = models.DateTimeField()  
  
    def __str__(self):  
        return self.subject
```

게시판노트 앱 만들기

■ 관리자 페이지에서 모델 관리하기

- 데이터 검색을 위해 항목을 추가한다.

bbsnote/admin.py

```
...  
class BoardAdmin(admin.ModelAdmin):  
    search_fields = ['subject']  
  
admin.site.register(Board, BoardAdmin)
```

The screenshot shows the Django administration interface for the 'bbsnote' application. The top navigation bar is dark blue with 'Django administration' on the left and 'WELCOME, ADMIN.' on the right. Below this is a light blue breadcrumb trail: 'Home > Bbsnote > Boards'. The main content area has a title 'Select board to change' followed by a search bar with a magnifying glass icon and a 'Search' button. Below the search bar is an 'Action:' dropdown menu with a 'Go' button and the text '0 of 1 selected'. A table lists the available boards, with one entry: 'BOARD' with a checkbox. Below the table, it says '1 board'. A red dashed box highlights the search bar and the 'Action:' dropdown area.

■ 템플릿 생성하기

- board 모델 데이터를 템플릿 파일을 사용하여 화면에 출력할 수 있도록 views.py에서 render 함수를 활용한다.
render 함수는 context에 있는 Board 모델 데이터 board_list를 템플릿(bbsnote/board_list.html)에 적용하여 출력한다.

bbsnote/views.py

```
from django.shortcuts import render
from django.http import HttpResponse
from .models import Board
```

```
# Create your views here.
```

```
def index(request):
```

```
    board_list = Board.objects.order_by('-create_date')
```

```
    context = {'board_list': board_list}
```

```
    #return HttpResponse("bbsnote에 오신것을 환영합니다!");
```

```
    return render(request, 'bbsnote/board_list.html', context)
```

■ 템플릿 생성하기

- 모델 데이터를 렌더링 하기 위해 작성된 html 템플릿 파일을 저장하기 위해 디렉터리를 만들고, 위치를 등록한다.
(프로젝트 폴더 내부에 templates 폴더가 위치해야 하며 그 하위에 앱 이름으로 폴더를 한번 더 만들어서 템플릿을 저장한다.)

config/settings.py

```
...
TEMPLATES = [
    {
        ...
        'DIRS': [BASE_DIR/'templates'],
        ...
    },
]
...
```

■ 템플릿 생성하기

templates/bbsnote/board_list.html

```
{% if board_list %}
    <ul>
        {% for board in board_list %}
            <li><a href="/bbsnote/{{board.id}}/">{{board.subject}}</a></li>
        {% endfor %}
    </ul>
{% else %}
    <p>글이 없습니다.</p>
{% endif %}
```

- 템플릿 태그 : {% 와 %} 로 둘러싸인 문장

<https://docs.djangoproject.com/en/3.1/topics/templates/>

- 템플릿 변수 : {{ }}로 둘러싸인 값

■ 상세 페이지 기능 구현하기

bbsnote/urls.py

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index),
    path('<int:board_id>/', views.detail),
]
```

- int는 컨버터라 부르는 기능. board_id는 컨버터를 통해 반환 받은 값을 의미한다.
(int는 board_id에 숫자가 매핑되었음을 의미)
- 기본 제공되는 컨버터 종류
 - 1) str : 비어있지 않은 모든 문자와 매칭. 단 / 는 제외. 컨버터 설정하지 않을 경우 기본 컨버터
 - 2) int : 0을 포함한 양의 정수와 매칭
 - 3) slug : 아스키 문자나 숫자, 하이픈, 언더스코어를 포함한 슬러그 문자열과 매칭
 - 4) uuid : UUID 와 매칭. 같은 페이지에 여러 URL이 연결되는 것을 막으려고 사용
 - 5) path : 기본적으로 str와 같은 기능이나 / 도 포함. URL의 부분이 아닌 전체에 대한 매칭을 하고 싶을 때 사용

■ 상세 페이지 기능 구현하기

bbsnote/views.py

```
...  
def detail(request, board_id):  
    board = Board.objects.get(id=board_id)  
    context = {'board': board}  
    return render(request, 'bbsnote/board_detail.html', context)
```

templates/bbsnote/board_detail.html

```
<h1>{{board.subject}}</h1>  
<div>  
    {{board.content}}  
</div>
```

■ URL 별칭 사용하기

- app_name 변수에 네임스페이스 - 각각의 앱이 관리하는 독립된 이름공간 - 를 정의한다.
- path 함수에 있는 URL 매핑에 name속성을 부여한다.
네임스페이스를 이용함으로써 각 앱 별로 동일한 name이 부여되어도 구분이 된다.

bbsnote/urls.py

```
...
app_name = 'bbsnote'

urlpatterns = [
    path('', views.index, name='index'),
    path('<int:board_id>/', views.detail, name='detail'),
]
```

templates/bbsnote/board_list.html

```
...
<li><a href="{% url 'bbsnote:detail' board.id %}">{{board.subject}}</a></li>
...
```

게시판노트 앱 만들기

kt ds University

■ 댓글 구현하기

bbsnote/urls.py

```
...
urlpatterns = [
    path('', views.index, name='index'),
    path('<int:board_id>/', views.detail, name='detail'),
    path('comment/create/<int:board_id>/', views.answer_create, name='comment_create'),
]
```

bbsnote/views.py

```
from django.shortcuts import render, redirect
from django.http import HttpResponse
from .models import Board, Answer
from django.utils import timezone
...
def comment_create(request, board_id):
    board = Board.objects.get(id=board_id)
    comment = Comment(board=board, content=request.POST.get('content'), create_date=timezone.now())
    comment.save()
    return redirect('bbsnote:detail', board_id=board.id)
```

■ 댓글 구현하기

- foreign_key로 연결된 모델의 경우 **연결모델명_set** 형태로 연결된 데이터를 조회하거나 등록을 할 수 있다.

Board 모델을 통해 Comment 모델 데이터 생성을 위해 board.comment_set.create를 사용한다.

bbsnote/views.py

```
from django.shortcuts import render, redirect
from django.http import HttpResponse
from .models import Board, Answer
from django.utils import timezone
...
def comment_create(request, board_id):
    board = Board.objects.get(id=board_id)
    board.comment_set.create(content=request.POST.get('content'), create_date=timezone.now())
    return redirect('bbsnote:detail', board_id=board.id)
```

■ 댓글 구현하기

- csrf_token(Cross Site Request Forgery)

csrf 공격을 막기 위해 form 태그 아래에 작성.

form 태그를 통해 전송된 데이터가 실제 브라우저에서 작성된 데이터인지 판단.

templates/bbsnote/board_detail.html

```
...
<h5>{{board.comment_set.count}}개의 댓글이 있습니다.</h5>
<div>
  <ul>
    {% for comment in board.comment_set.all %}
      <li>{{comment.content}}</li>
    {% endfor %}
  </ul>
</div>
<form action="{% url 'bbsnote:comment_create' board.id %}" method="post">
  {% csrf_token %}
  <textarea name="content" id="content" rows="15"></textarea>
  <input type="submit" value="댓글달기">
</form>
```

■ 스타일 적용하기

- 정적파일인 css를 적용하기 위해 디렉터리를 추가하고, settings.py 에 경로를 추가한다.

config/settings.py

```
...  
STATIC_URL = '/static/'  
STATICFILES_DIRS = [  
    BASE_DIR / 'static',  
]
```

- 스타일 적용을 하기 위해 부트스트랩 파일을 static 폴더에 추가하고, 템플릿에 스타일을 적용한다.

templates/bbsnote/board_list.html

```
{% load static %}  
<link rel="stylesheet" type="text/css" href="{% static 'bootstrap.min.css' %}">  
{% if board_list %}  
...
```

■ 스타일 적용하기

- 장고는 템플릿 상속 기능을 제공하여, 기본 틀을 만들어 활용이 가능하다.

body 태그 안 block content 템플릿 태그는 base.html을 상속받은 템플릿 파일에서 구현해야 하는 영역이 된다.

templates/base.html

```
{% load static %}
<!doctype html>
<html lang="ko">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">
  <link rel="stylesheet" type="text/css" href="{% static 'bootstrap.min.css' %}">
  <title>게시판 노트</title>
</head>
<body>
<!-- 기본 템플릿 안에 삽입될 내용 Start -->
{% block content %}
{% endblock %}
<!-- 기본 템플릿 안에 삽입될 내용 End -->
</body>
</html>
```


■ 스타일 적용하기

- 앞서 작성한 base.html을 상속받기 위한 템플릿 태그를 추가하고, block content 사이에 들어갈 내용을 구성한다.

templates/bbsnote/board_list.html

```
{% extends 'base.html' %}
{% block content %}
<div class="container my-3">
  <table class="table">
    ...
    <tbody>
      {% if board_list %}
      {% for board in board_list %}
      <tr class="text-center">
        <td class="text-left">
          <a href="{% url 'bbsnote:detail' board.id %}">{{ board.subject }}</a>
        </td>
        <td>{{ board.create_date }}</td>
      </tr>
      {% endfor %}
      {% else %}
      ...
      {% endif %}
    </tbody>
  </table>
</div>
{% endblock %}
```

■ 게시글 등록 기능 만들기

bbsnote/forms.html

```
from django import forms
from bbsnote.models import Board

class BoardForm(forms.ModelForm):
    class Meta:
        model = Board
        fields = ['subject', 'content']
```

- 모델 폼 : forms.ModelForm 을 상속받는 폼.
모델과 연결된 폼이며, 모델 폼 객체를 저장하면 연결된 모델의 데이터를 저장 할 수 있다.
Meta 클래스를 내부 클래스에 반드시 가져가야 하며, 모델 폼이 사용할 모델과 모델의 필드를 작성해야 한다.
- 폼 : forms.Form을 상속받는 폼.
모델 폼과 다르게 직접 필드를 정의하고, 위젯 설정을 해줘야한다.

bbsnote/urls.py

```
urlpatterns = [
    ...
    path('board/create/', views.board_create, name='board_create'),
]
```

■ 게시글 등록 기능 만들기

bbsnote/views.py

```
...
from .forms import BoardForm
...
def board_create(request):
    if request.method == 'POST':
        form = BoardForm(request.POST)
        if form.is_valid():
            board = form.save(commit=False)
            board.create_date = timezone.now()
            board.save()
            return redirect('bbsnote:index')
    else:
        form = BoardForm()
    return render(request, 'bbsnote/board_form.html', {'form': form})
```

- BoardForm 클래스로 생성한 form을 사용하여, 템플릿에게 form을 전달하게 되고, 템플릿을 그 form으로 폼 엘리먼트를 생성할 때 사용한다.

■ 게시글 등록 기능 만들기

- board_create함수에서 전달한 BoardForm 객체가 form.as_p 의 form이다.
form.as_p는 subject, content에 값을 입력할 수 있는 HTML코드를 자동으로 만들어 준다.

templates/bbsnote/question_form.html

```
{% extends 'base.html' %}
{% block content %}
<div class="container">
  <h5 class="my-3 border-bottom pb-2">질문 등록</h5>
  <form method="post" class="post-form my-3">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit" class="btn btn-primary">저장</button>
  </form>
</div>
{% endblock %}
```

templates/bbsnote/question_form.html

```
...
</table>
<a href="{% url 'bbsnote:board_create' %}" class="btn btn-primary">등록</a>
</div>
```

게시판노트 앱 만들기

■ 로그인 • 로그아웃 구현하기

- 장고는 Django.contrib.auth 앱을 제공하여, 로그인 • 로그아웃을 쉽게 구현할 수 있게 해준다. (settings.py에서 확인가능)
- 공통 기능인 로그인 • 로그아웃은 common 앱에 별도 구현한다.

```
(mysite) D:\mysite>django-admin startapp common
```

config/settings.py

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'bbsnote.apps.BbsnoteConfig',  
    'common.apps.CommonConfig',  
]  
...  
#로그인 로그아웃 성공 시 자동으로 이동할 URL  
LOGIN_REDIRECT_URL = '/'  
LOGOUT_REDIRECT_URL = '/'
```

■ 로그인 • 로그아웃 구현하기

- common URL을 사용하기 위해 추가하고, 앞에 설정한 '/'에 대응하기 위한 URL path도 추가한다.

config/urls.py

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('bbsnote/', include('bbsnote.urls')),  
    path('common/', include('common.urls')),  
    path('', views.index, name='index'),  
]
```

- Django.contrib.auth 앱 LoginView클래스를 사용할 것이므로, views.py 추가 없이, urls.py 를 수정한다.

common/urls.py

```
from django.contrib.auth import views as auth_views  
from django.urls import path  
  
app_name = 'common'  
  
urlpatterns = [  
    path('login/', auth_views.LoginView.as_view(template_name='common/login.html'), name='login'),  
    path('logout/', auth_views.LogoutView.as_view(), name='logout'),  
]
```

■ 로그인 · 로그아웃 구현하기

- 아이디와 비밀번호를 입력받는 페이지와 로그인에 실패했을 경우 오류 메시지를 표현할 부분을 추가하여 준다.

templates/common/login.html

```
{% extends "base.html" %}
{% block content %}
<div class="container my-3">
  ...
  <form method="post" class="post-form" action="{% url 'common:login' %}">
    {% csrf_token %}
    <input type="hidden" name="next" value="{{ next }}"> <!-- 로그인 성공후 이동되는 URL -->
    {% include "form_errors.html" %}
    <div class="form-group">
      <label for="username">사용자ID</label>
      <input type="text" class="form-control" name="username" id="username"
        value="{{ form.username.value|default_if_none:'' }}">
    </div>
    <div class="form-group">
      <label for="password">비밀번호</label>
      <input type="password" class="form-control" name="password" id="password"
        value="{{ form.password.value|default_if_none:'' }}">
    </div>
    <button type="submit" class="btn btn-primary">로그인</button>
  </form>
</div>
{% endblock %}
```

■ 로그인 · 로그아웃 구현하기

- 아이디와 비밀번호를 입력받는 페이지와 로그인에 실패했을 경우 오류 메시지를 표현할 부분을 추가하여 준다.

templates/form_errors.html

```
{% if form.errors %}
  {% for field in form %}
    {% for error in field.errors %} <!-- 필드 오류를 출력한다. -->
      <div class="alert alert-danger">
        <strong>{{ field.label }}</strong>
        {{ error }}
      </div>
    {% endfor %}
  {% endfor %}
  {% for error in form.non_field_errors %} <!-- 년필드 오류를 출력한다. -->
    <div class="alert alert-danger">
      <strong>{{ error }}</strong>
    </div>
  {% endfor %}
{% endif %}
```


■ 회원가입 기능 추가하기

- Django.contrib.auth 앱을 이용하여 구현한다.
common/forms.html

```
from django import forms
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User

class UserForm(UserCreationForm):
    email = forms.EmailField(label="이메일")

    class Meta:
        model = User
        fields = ("username", "email")
```

- UserCreationForm 은 기본적으로 username, password1, password2를 가지고 있고, email속성을 추가하기 위해 UserCreationForm을 상속받는 UserForm을 생성한다.

■ 회원가입 기능 추가하기

common/forms.html

```
from django.contrib.auth import authenticate, login
from django.shortcuts import render, redirect

from .forms import UserForm

def signup(request):
    if request.method == "POST":
        form = UserForm(request.POST)
        if form.is_valid():
            form.save()
            username = form.cleaned_data.get('username')
            raw_password = form.cleaned_data.get('password1')
            user = authenticate(username=username, password=raw_password)
            login(request, user)
            return redirect('index')
    else:
        form = UserForm()
    return render(request, 'common/signup.html', {'form': form})
```

- 회원가입 완료된 후 자동 로그인에 가능하도록 authenticate와 login 함수를 사용한다.

■ 회원가입 기능 추가하기

common/forms.html

```
...
<form method="post" class="post-form">
  {% csrf_token %}
  {% include "form_errors.html" %}
  <div class="form-group">
    <label for="username">사용자 이름</label>
    <input type="text" class="form-control" name="username" id="username"
      value="{{ form.username.value|default_if_none:'' }}">
  </div>
  <div class="form-group">
    <label for="password1">비밀번호</label>
    <input type="password" class="form-control" name="password1" id="password1"
      value="{{ form.password1.value|default_if_none:'' }}">
  </div>
  <div class="form-group">
    <label for="password2">비밀번호 확인</label>
    <input type="password" class="form-control" name="password2" id="password2"
      value="{{ form.password2.value|default_if_none:'' }}">
  </div>
  <div class="form-group">
    <label for="email">이메일</label>
    <input type="text" class="form-control" name="email" id="email"
      value="{{ form.email.value|default_if_none:'' }}">
  </div>
  <button type="submit" class="btn btn-primary">생성하기</button>
</form>
...
```

■ 회원가입 기능 추가하기

templates/common/login.html

```
...
<div class="row">
  <div class="col-4">
    <h4>로그인</h4>
  </div>
  <div class="col-8 text-right">
    <span>또는 <a href="{% url 'common:signup' %}">계정을 만드세요.</a></span>
  </div>
</div>
...
```

common/urls.py

```
...
urlpatterns = [
    path('login/', auth_views.LoginView.as_view(template_name='common/login.html'), name='login'),
    path('logout/', auth_views.LogoutView.as_view(), name='logout'),
    path('signup/', views.signup, name='signup'),
]
```

게시판노트 앱 만들기

kt ds University

■ 게시판 페이징 적용하기

bbsnote/views.py

```
...
from django.core.paginator import Paginator

def index(request):
    #입력 인자
    page = request.GET.get('page', 1)
    #조회
    board_list = Board.objects.order_by('-create_date')
    #페이징 처리
    paginator = Paginator(board_list, 5)
    page_obj = paginator.get_page(page)

    context = {'board_list' : page_obj}
    #return HttpResponse("bbsnote에 오신것을 환영합니다!");
    return render(request, 'bbsnote/board_list.html', context)
...
```

■ 게시판 페이징 적용하기

- 장고의 Paginator 클래스를 이용하면 다음 속성들 사요잉 가능하여 페이징 처리가 간단해 진다.
- 아래 속성들은 템플릿에서 페이징 처리시에도 사용된다.

항목	설명
paginator.count	전체 게시물 수
paginator.per_page	페이지당 보여줄 게시물 개수
paginator.page_range	페이지 범위
number	현재 페이지 번호
previous_page_number	이전 페이지 번호
next_page_number	다음 페이지 번호
has_previous	이전 페이지 유무
has_next	다음 페이지 유무
start_index	현재 페이지 시작 인덱스(1부터 시작)
end_index	현재 페이지의 끝 인덱스(1부터 시작)

■ 게시판 페이징 적용하기

templates/bbsnote/board_list.html

```
...
<!-- 페이지리스트 -->
{% for page_number in board_list.paginator.page_range %}
{% if page_number >= board_list.number|add:-5 and page_number <= board_list.number|add:5 %}
    {% if page_number == board_list.number %}
        <li class="page-item active" aria-current="page">
            <a class="page-link" data-page="{{ page_number }}" href="#">{{ page_number }}</a>
        </li>
    {% else %}
        <li class="page-item">
            <a class="page-link" data-page="{{ page_number }}" href="#">{{ page_number }}</a>
        </li>
    {% endif %}
{% endif %}
{% endfor %}
...
```

- board_list가 views.py의 page_obj. 페이지 표시 제한 기능을 이용하여 현재 페이지 기준으로 좌우 5개씩 보이도록 설정한다.

■ 템플릿 필터 적용하기

- 템플릿 필터 : 템플릿 태그에서 | 문자 뒤에 사용하는 필터
- 게시글 앞에 번호를 붙이고 페이징 결과를 확인해봅니다.

templates/bbsnote/board_list.html

```
...  
    <tr class="text-center thead-dark">  
        <th></th>  
        <th style="width:50%">제목</th>  
        <th>작성일시</th>  
    </tr>  
...  
    {% for board in board_list %}  
    <tr class="text-center">  
        <td>{{ forloop.counter }}</td>  
        <td class="text-left">  
            <a href="{% url 'bbsnote:detail' board.id %}">{{ board.subject }}</a>  
        </td>  
        <td>{{ board.create_date }}</td>  
    </tr>  
    {% endfor %}  
...
```


■ 템플릿 필터 적용하기

- 게시물 번호 공식 : 일련번호 = 전체 게시물 개수 - 시작인덱스 - 현재 인덱스 + 1
더하기 필터에는 변수를 적용할 수 없기에 빼기 필터를 만듭니다.

bbsnote/templatetags/bbsnote_filter.py

```
from django import template

register = template.Library()

@register.filter
def sub(value, arg):
    return value - arg
```

- 애너테이션 적용하면 템플릿에서 함수를 필터로 사용할 수 있게 된다.

■ 템플릿 필터 적용하기

- 템플릿 필터를 사용하기 위해 템플릿 맨 위 - extends문 아래 - 에 템플릿 필터 파일을 로드해야 한다.
- 기존에 있던 forloop.counter를 교체한다.

templates/bbsnote/board_list.html

```
{% extends 'base.html' %}
{% load bbsnote_filter %}
{% block content %}
...
<td>{{ board_list.paginator.count|sub:board_list.start_index|sub:forloop.counter0|add:1 }}</td>
...
```

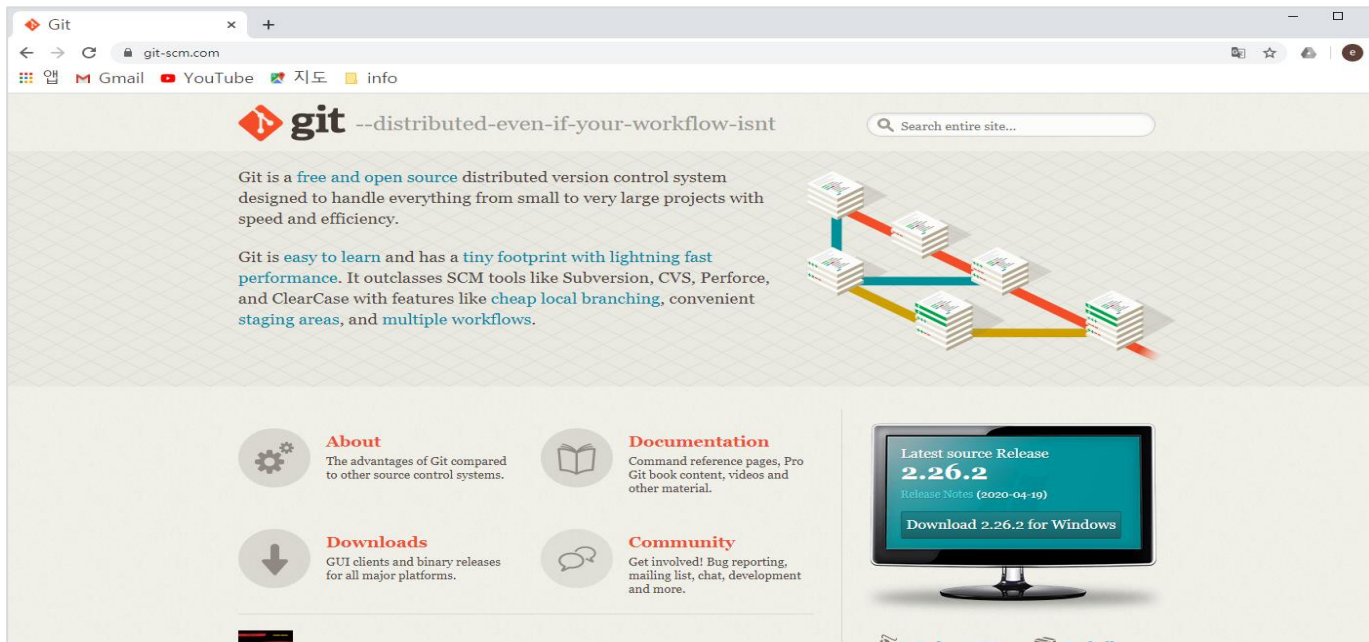
공식요소	설명
question_list.paginator.count	전체 게시물 수
question_list.start_index	시작 인덱스(1부터 시작)
forloop.counter0	forloop내의 현재 인덱스 forloop.counter0은 0부터, forloop.counter는 1부터 시작

배포하기

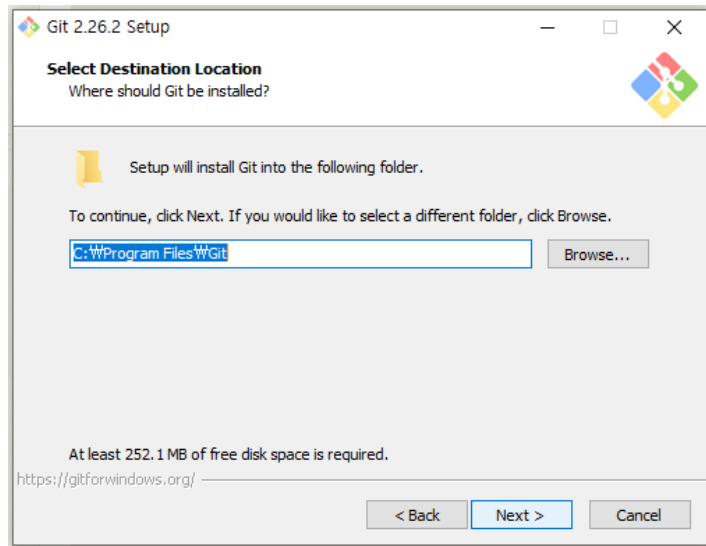
■ git 설치

깃 허브등 깃을 사용하는 서비스를 통해 소스코드를 관리하려면 깃 명령 관련 프로그램 설치가 필요

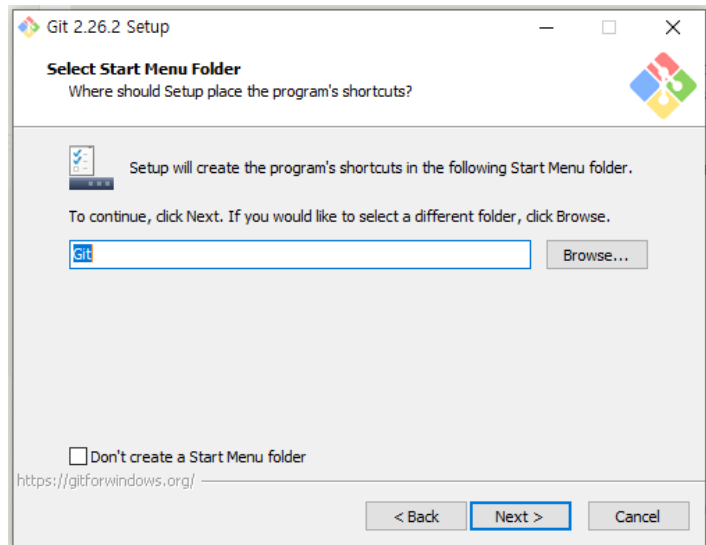
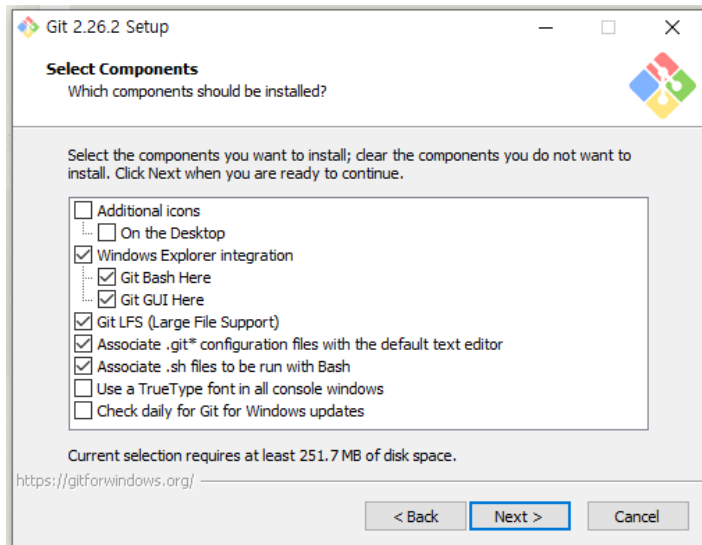
<https://git-scm.com/>



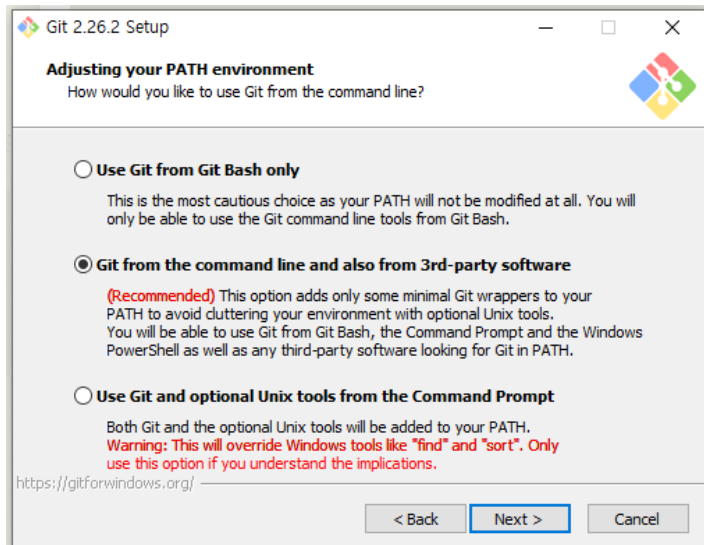
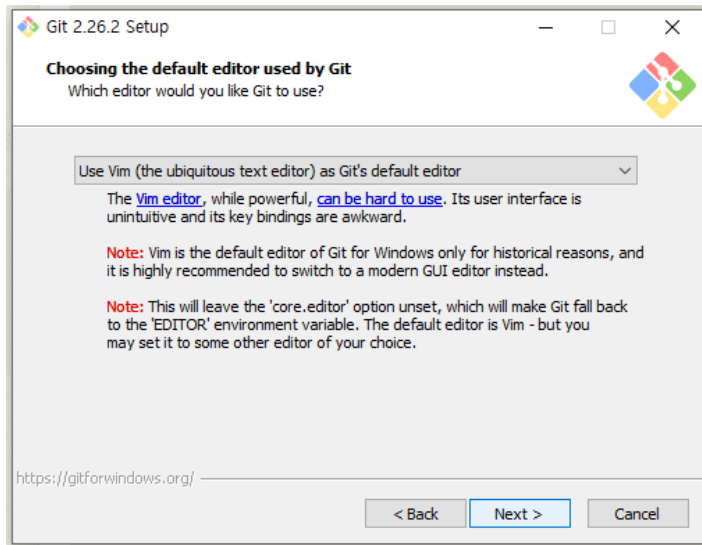
■ git 설치



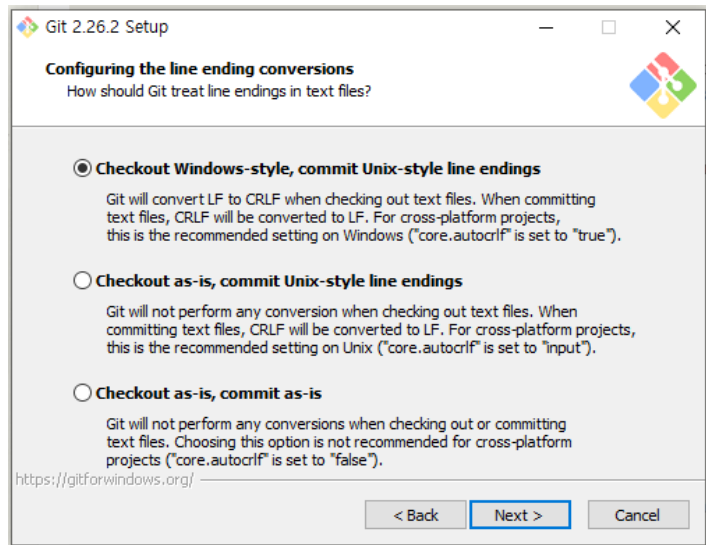
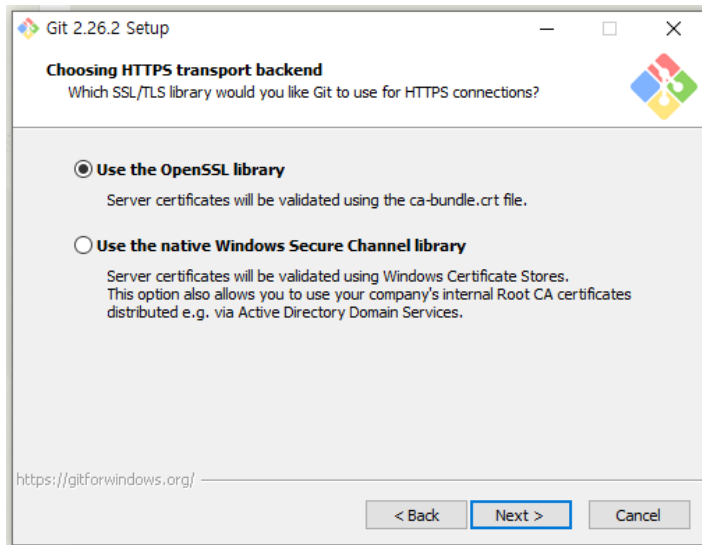
■ git 설치



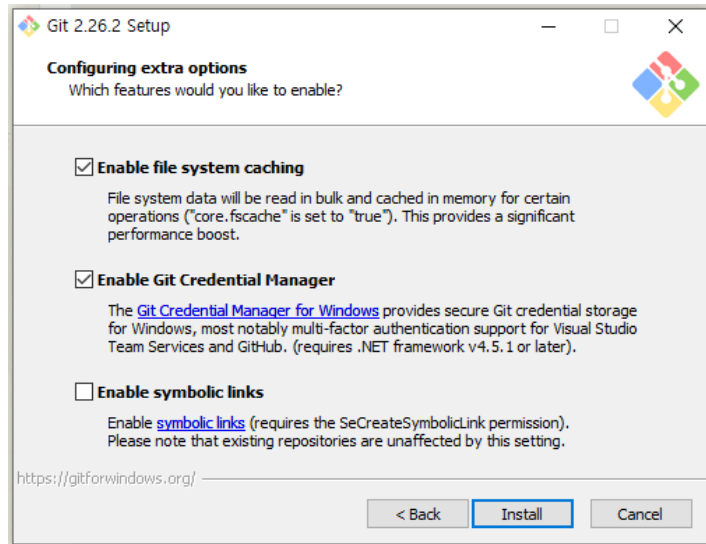
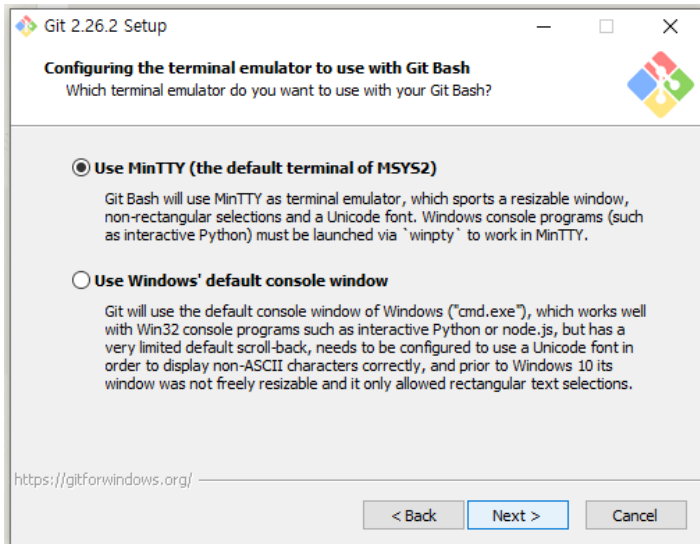
■ git 설치



■ git 설치



■ git 설치



■ 저장소 만들기

- 아나콘다에서 해당 디렉토리를 깃을 이용해 관리하겠다는 의미로 명령을 입력

```
(mysite) D:\mysite>git init  
Initialized empty Git repository in D:/mysite/.git/
```

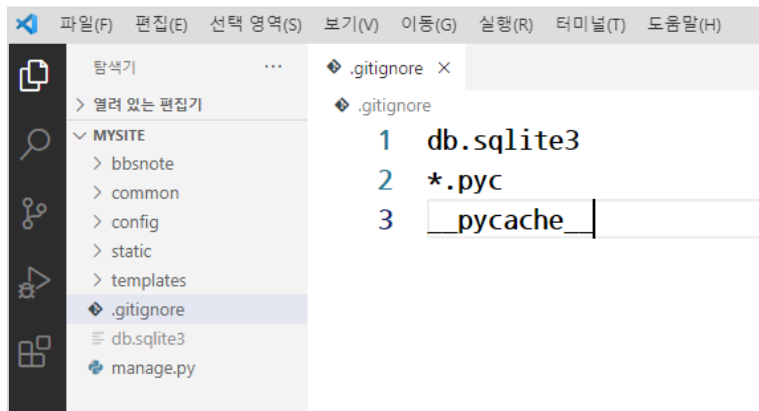
■ 깃 제외파일 관리

- .gitignore - 깃에 업로드 하지 않을 파일 목록을 설정하는 파일

보통 데이터베이스 파일이나 비밀번호가 들어있는 파일 혹은 캐시 파일 등

업로드 하면 안되거나 굳이 필요없는 파일들의 목록을 작성

프로젝트 루트디렉토리에 .gitignore 파일 생성



■ 깃 등록

- 깃을 통해 관리할 파일 목록을 등록

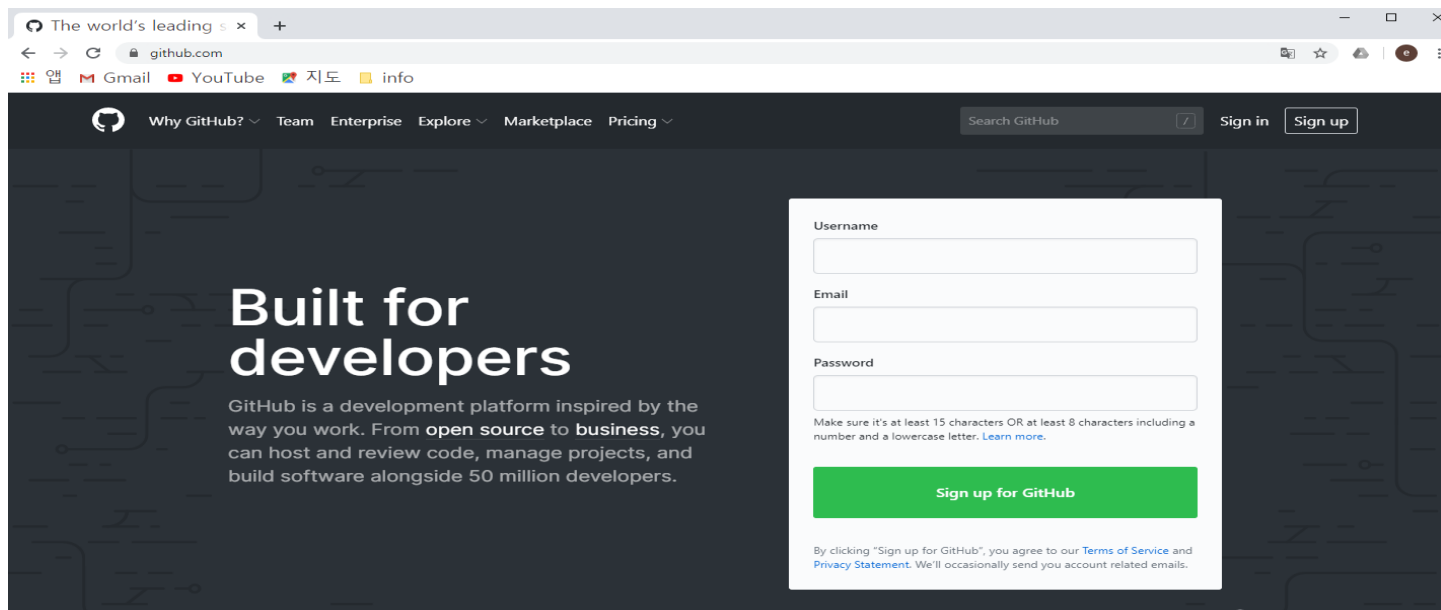
```
(mysite) D:\mysite>git add *  
warning: LF will be replaced by CRLF in static/bootstrap.min.css.  
The file will have its original line endings in your working directory  
warning: LF will be replaced by CRLF in static/bootstrap.min.js.  
The file will have its original line endings in your working directory  
warning: LF will be replaced by CRLF in static/jquery-3.4.1.min.js.  
The file will have its original line endings in your working directory
```

■ 깃 커밋

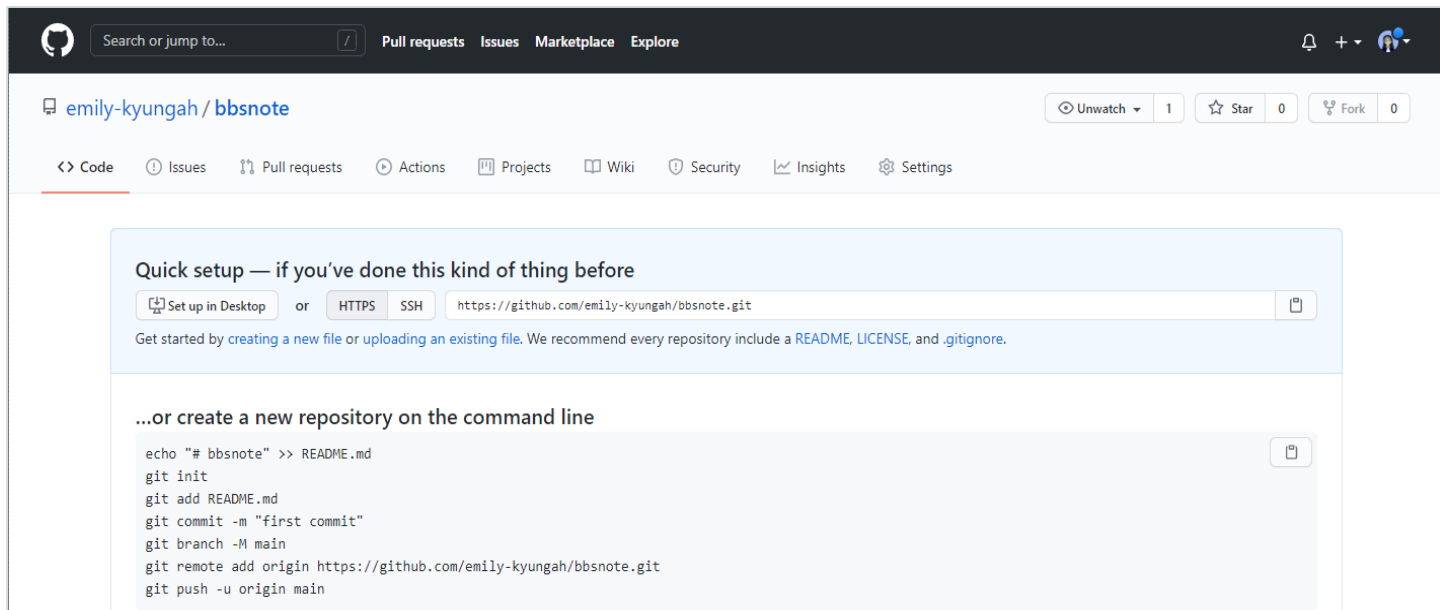
- 변경사항 확정을 위해 commit 을 진행

```
(mysite) D:\mysite>git commit -m "bbsnote first commit"  
[master (root-commit) 0e10c6d] bbsnote first commit  
41 files changed, 735 insertions(+)  
create mode 100644 .gitignore  
create mode 100644 bbsnote/__init__.py  
create mode 100644 bbsnote/admin.py  
create mode 100644 bbsnote/apps.py  
create mode 100644 bbsnote/forms.py  
create mode 100644 bbsnote/migrations/0001_initial.py  
create mode 100644 bbsnote/migrations/0002_comment.py  
create mode 100644 bbsnote/migrations/0003_delete_answer.py  
create mode 100644 bbsnote/migrations/__init__.py  
create mode 100644 bbsnote/models.py  
create mode 100644 bbsnote/templatetags/bbsnote_filter.py
```

■ 깃 허브(<https://github.com/>)



■ 깃 허브(<https://github.com/>)



■ 깃 허브 - 로컬 저장소와 원격 저장소 연결

소스 코드를 업로드할 리포지토리를 origin으로 등록

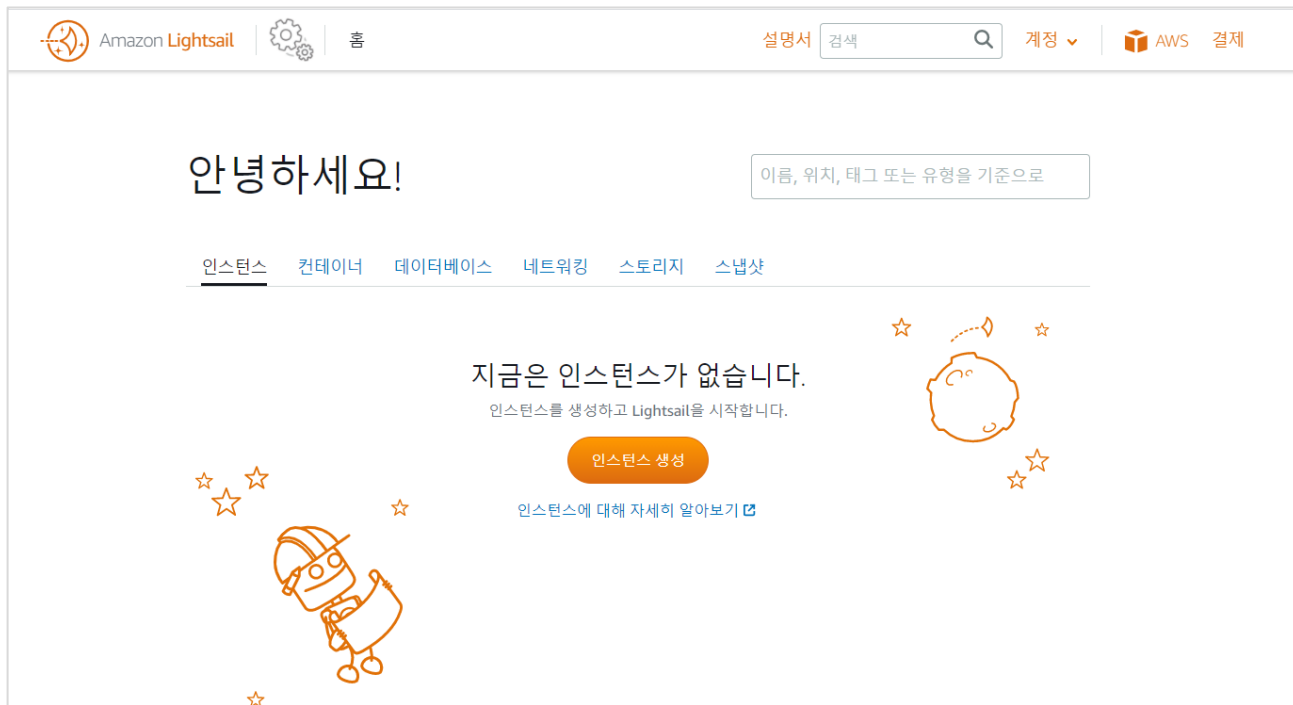
```
(mysite) D:\mysite>git remote add origin https://github.com/emily-kyungah/bbsnote.git
```

push 명령으로 코드 업로드 진행(처음이라면 깃허브 아이디와 비밀번호 입력하면 됨)

```
(mysite) D:\mysite>git push -u origin master
Enumerating objects: 48, done.
Counting objects: 100% (48/48), done.
Delta compression using up to 8 threads
Compressing objects: 100% (44/44), done.
Writing objects: 100% (48/48), 80.27 KiB | 3.09 MiB/s, done.
Total 48 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/emily-kyungah/bbsnote.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

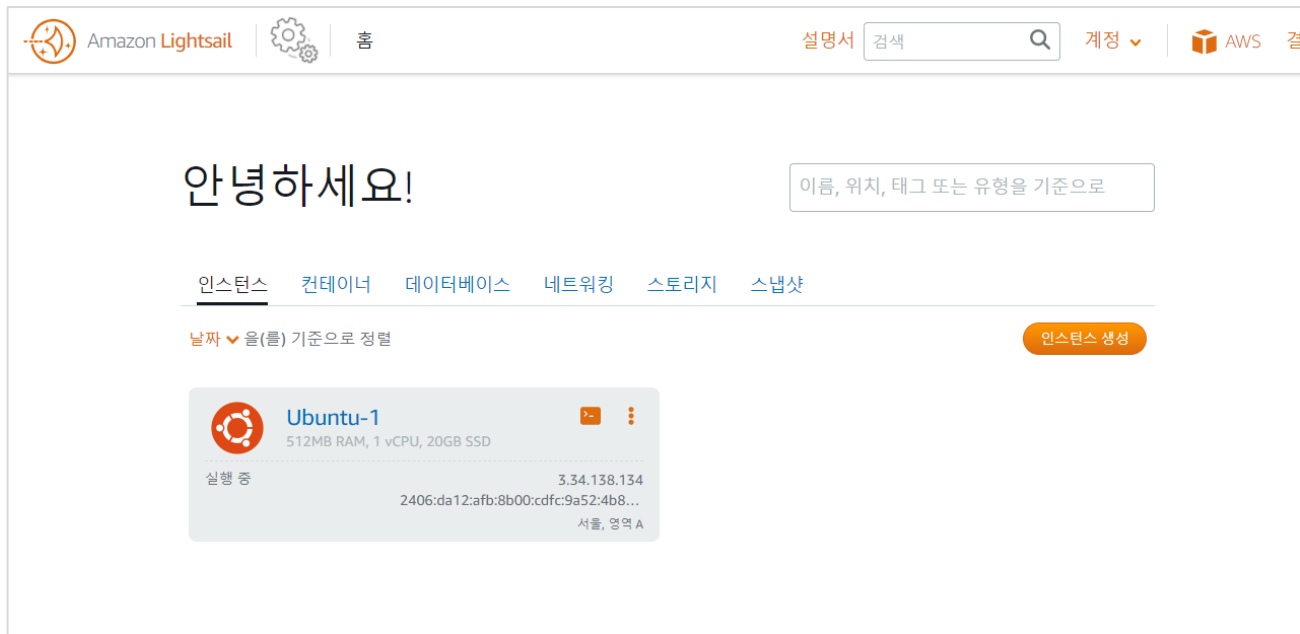
■ AWS 라이트세일 (<https://lightsail.aws.amazon.com/>)

- 아마존에서 운영하는 웹 서비스에 특화된 클라우드 서비스



■ AWS 라이트세일 (<https://lightsail.aws.amazon.com/>)

- 간단한 운영환경 구성을 통해 한달 무료로 사용이 가능하다.



■ AWS 터미널 접속하기

AWS 터미널

```
ubuntu@ip-172-26-12-184:~$ date
Thu Mar 18 15:44:45 UTC 2021
ubuntu@ip-172-26-12-184:~$ sudo ln -sf /usr/share/zoneinfo/Asia/Seoul /etc/localtime
ubuntu@ip-172-26-12-184:~$ date
Fri Mar 19 00:45:26 KST 2021
ubuntu@ip-172-26-12-184:~$ python
```

Command 'python' not found, did you mean:

```
command 'python3' from deb python3
command 'python' from deb python-is-python3
```

```
ubuntu@ip-172-26-12-184:~$ python3
Python 3.8.2 (default, Apr 27 2020, 15:53:34)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
```

■ AWS 터미널 접속하기

AWS 터미널

```
ubuntu@ip-172-26-12-184:~$ sudo apt update
...
Reading package lists... Done
Building dependency tree
Reading state information... Done
178 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-172-26-12-184:~$ sudo apt install python3-venv
...
ubuntu@ip-172-26-12-184:~$ mkdir projects
ubuntu@ip-172-26-12-184:~$ mkdir venvs
ubuntu@ip-172-26-12-184:~$ ls
projects  venvs
ubuntu@ip-172-26-12-184:~$ cd venvs
ubuntu@ip-172-26-12-184:~/venvs$ python3 -m venv bbsnote
ubuntu@ip-172-26-12-184:~/venvs$ cd bbsnote
ubuntu@ip-172-26-12-184:~/venvs/bbsnote$ cd bin
ubuntu@ip-172-26-12-184:~/venvs/bbsnote/bin$ . activate
(bbsnote) ubuntu@ip-172-26-12-184:~/venvs/bbsnote/bin$
(bbsnote) ubuntu@ip-172-26-12-184:~/venvs/bbsnote/bin$ deactivate
ubuntu@ip-172-26-12-184:~/venvs/bbsnote/bin$
```

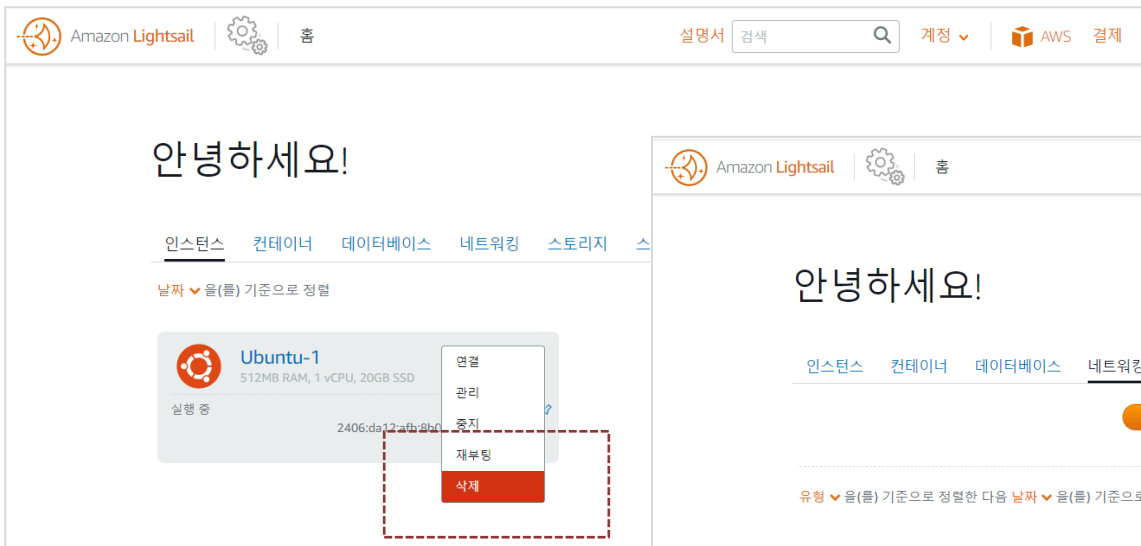
■ AWS 터미널 접속하기

AWS 터미널

```
(bbsnote) ubuntu@ip-172-26-12-184:~/venvs/bbsnote/bin$ pip install wheel
Collecting wheel
  Downloading wheel-0.36.2-py2.py3-none-any.whl (35 kB)
Installing collected packages: wheel
Successfully installed wheel-0.36.2
(bbsnote) ubuntu@ip-172-26-12-184:~/venvs/bbsnote/bin$ pip install django==3.1.3
(bbsnote) ubuntu@ip-172-26-12-184:~/venvs/bbsnote/bin$ cd ~/projects
(bbsnote) ubuntu@ip-172-26-12-184:~/projects$ git clone https://github.com/emily-kyungah/bbsnote.git bbsnote
Cloning into 'bbsnote'...
remote: Enumerating objects: 48, done.
remote: Counting objects: 100% (48/48), done.
remote: Compressing objects: 100% (42/42), done.
remote: Total 48 (delta 2), reused 48 (delta 2), pack-reused 0
Unpacking objects: 100% (48/48), 80.25 KiB | 454.00 KiB/s, done.
(bbsnote) ubuntu@ip-172-26-12-184:~/projects$ ls
bbsnote
(bbsnote) ubuntu@ip-172-26-12-184:~/projects$ cd bbsnote
(bbsnote) ubuntu@ip-172-26-12-184:~/projects/bbsnote$ python manage.py runserver
(bbsnote) ubuntu@ip-172-26-12-184:~/projects/bbsnote$ python manage.py migrate
(bbsnote) ubuntu@ip-172-26-12-184:~/projects/bbsnote$ python manage.py runserver
```

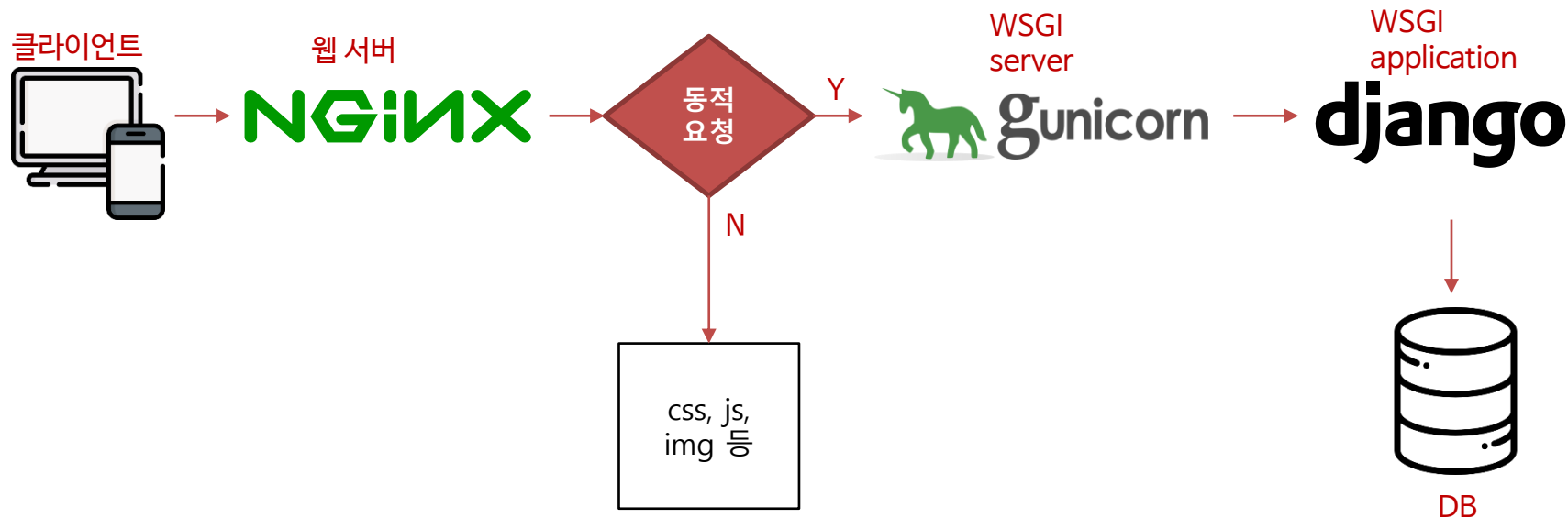
■ AWS 라이트세일 (<https://lightsail.aws.amazon.com/>)

- 더 이상 사용하지 않는 경우 삭제해야 비용 발생을 막을 수 있다.



■ WSGI (Web Server Gateway Interface)

- 웹 서버와 웹 애플리케이션의 인터페이스를 위한 파이썬 프레임워크



■ Gunicorn 설치 및 확인

AWS 터미널

```
(bbsnote) ubuntu@ip-172-26-12-184:~/projects/bbsnote$ pip install gunicorn
Collecting gunicorn
  Downloading gunicorn-20.0.4-py2.py3-none-any.whl (77 kB)
    |████████████████████████████████████████| 77 kB 2.0 MB/s
Requirement already satisfied: setuptools>=3.0 in /home/ubuntu/venvs/bbsnote/lib/python3.8
/site-packages (from gunicorn) (44.0.0)
Installing collected packages: gunicorn
Successfully installed gunicorn-20.0.4
(bbsnote) ubuntu@ip-172-26-12-184:~/projects/bbsnote$ gunicorn --bind 0:8000 config.wsgi:application
(bbsnote) ubuntu@ip-172-26-12-184:~/projects/bbsnote$ nano /home/ubuntu/venvs/bbsnote.env
(bbsnote) ubuntu@ip-172-26-12-184:~/projects/bbsnote$ sudo nano bbsnote.service
(bbsnote) ubuntu@ip-172-26-12-184:~/projects/bbsnote$ sudo nano /etc/systemd/system/bbsnote.service
(bbsnote) ubuntu@ip-172-26-12-184:~/projects/bbsnote$ sudo systemctl start bbsnote.service
(bbsnote) ubuntu@ip-172-26-12-184:~/projects/bbsnote$ sudo systemctl status bbsnote.service
(bbsnote) ubuntu@ip-172-26-12-184:~/projects/bbsnote$ sudo systemctl enable bbsnote.service
```

■ Nginx 설치 및 확인

AWS 터미널

```
(bbsnote) ubuntu@ip-172-26-12-184:~/projects/bbsnote$ sudo apt install nginx
(bbsnote) ubuntu@ip-172-26-12-184:~/projects/bbsnote$ cd /etc/nginx/sites-available/
(bbsnote) ubuntu@ip-172-26-12-184:/etc/nginx/sites-available$ sudo nano bbsnote
(bbsnote) ubuntu@ip-172-26-12-184:/etc/nginx/sites-available$ cd /etc/nginx/sites-enabled/
(bbsnote) ubuntu@ip-172-26-12-184:/etc/nginx/sites-enabled$ ls
default
(bbsnote) ubuntu@ip-172-26-12-184:/etc/nginx/sites-enabled$ sudo rm default
(bbsnote) ubuntu@ip-172-26-12-184:/etc/nginx/sites-enabled$ sudo ln -s /etc/nginx/sites-available/bbsnote
(bbsnote) ubuntu@ip-172-26-12-184:/etc/nginx/sites-enabled$ ls
bbsnote
(bbsnote) ubuntu@ip-172-26-12-184:/etc/nginx/sites-enabled$ sudo systemctl restart nginx
(bbsnote) ubuntu@ip-172-26-12-184:/etc/nginx/sites-enabled$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

수고하셨습니다.