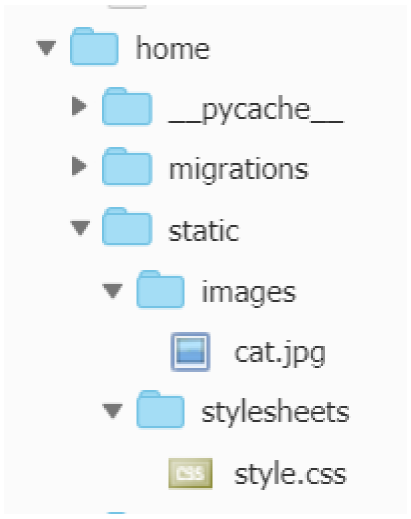
190513

Static 파일 사용하기

1. home/static/stylesheets, home/static/images 생성



2. static 파일 연결

[static_example.html]

```
{% load static %} # static 로드

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="{% static 'stylesheets/style.css' %}" type="text/css" /> #
static 가져옴
    <title>Document</title>
</head>
<body>
    <h1>STATIC 파일 실습</h1>
    <img src="{% static 'images/cat.jpg' %}"></img> # static 가져옴
</body>
</html>
```

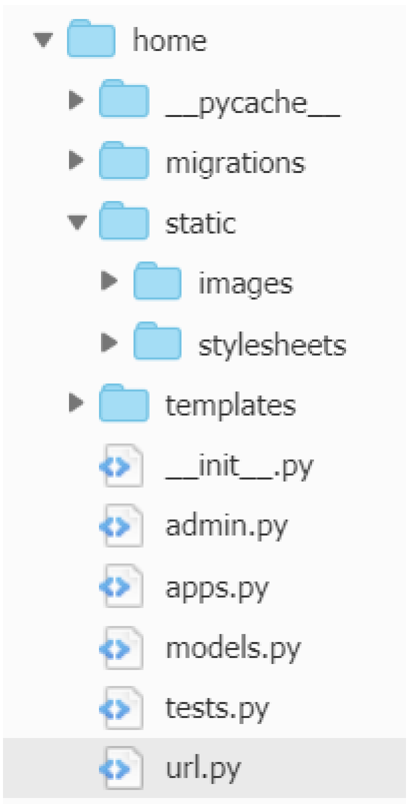
최상단에 {% load static %} 하고, 불러오는 곳에서도 {% static '경로' %}로 사용

STATIC 파일 실습

URL 나누기

앱이 늘어남에 따라, URL 을 나눌 필요가 생김.

1. home/urls.py 생성

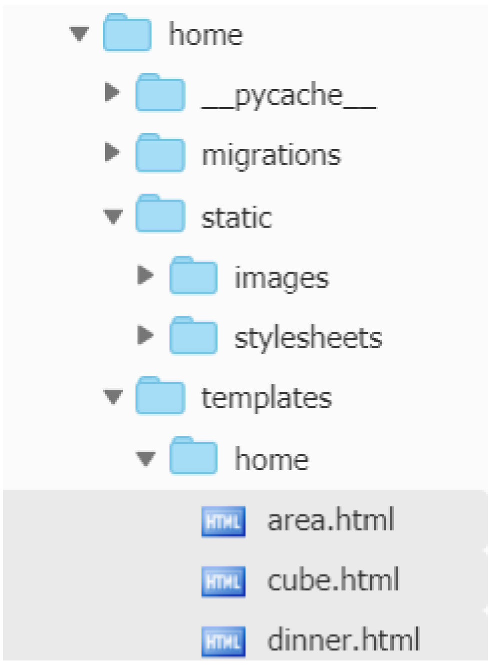

home
__pycache__
migrations
static
images
stylesheets
templates
__init__.py
admin.py
apps.py
models.py
tests.py
url.py

```
[home/urls.py]

from django.urls import path

# 현재 위치를 받아오는거니까 .
from . import views

urlpatterns = [
    path('index/',views.index), # sub url만 넣으면 됨
    path('lotto/',views.lotto),
    ...
]
```

2. /urls.py

```
from django.urls import path, include

urlpatterns = [
    path('home/', include('home.urls')), # home/ -> home.urls로 이동
    path('admin/', admin.site.urls),
]
```

3. App 이름과 동일한 Templates 하위 디렉터리 만들기



```
▼ home
    ▶ __pycache__
    ▶ migrations
    ▼ static
        ▶ images
        ▶ stylesheets
    ▼ templates
        ▼ home
            area.html
            cube.html
            dinner.html
```

다른 App과 url적으로 구분하기 위함

4. 기존에 root 정해준거에 /home/ 추가

[home/views.py]

```
def index(request):
    # return HttpResponse("hi?")
    return render(request, 'home/index.html')
```

기본 css 설정? 헤더 설정? 해주기 (base.html)

[django_intro/templates/base.html]

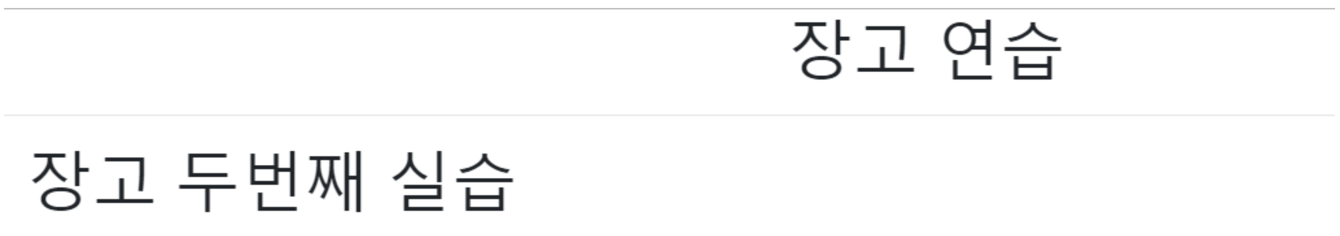
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
    <title>{% block title %}{% endblock %}</title>
    {% block css %}{% endblock %}
</head>
<body>
    <h1 class='text-center'>장고 연습</h1>
    <hr>
    <div class="container">
        {% block body %}
        {% endblock %}
    </div>
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous">
</script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
crossorigin="anonymous"></script>
</body>
</html>
```

[setting.py]

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'django_intro', 'templates')],
        ...
```

[utilites/templates/utilities/index.html]

```
{% extends 'base.html' %}

{% block title %}
장고 두번째 실습
{% endblock %}

{% block body %}
    <h1>장고 두번째 실습</h1>
{% endblock %}
```



장고 연습

장고 두번째 실습

Model

1. 새로운 프로젝트 설정

디렉토리 새로 만든 후, $ pyenv local django-venv -> 기존 가상환경 설정 가져올 수 있음

Q. 그럼 두 디렉토리의 환경은 연동되나?

1. 프로젝트 만들기
    1. $ django-admin startproject crud .

2. settings.py 설정

```
...
ALLOWED_HOSTS = ['python-example-ssoso27.c9users.io'] # Allowed Hosts
...
INSTALLED_APPS = [
    ...
    'boards', # app 정보 추가
]
...
TIME_ZONE = 'Asia/Seoul' # 서버에 영향을 주는 시간

USE_I18N = True

USE_L10N = True

USE_TZ = False # 모델에 영향을 주는 시간
```

3. app 생성
    1. $ django-admin startapp boards
4. git ignore 생성

2. Database

1. boards/models.py 설정

```
from django.db import models

# Create your models here.
# id는 장고 ORM이 자동으로 생성함.
class Board(models.Model): # table
    title = models.CharField(max_length=20) # column
    content = models.TextField() # CharField : max lenth 제한 o / TextField : 제한 x
    created_at = models.DateTimeField(auto_now_add=True)
```

2. migration

   $ python manage.py makemigrations

   - boards/migrations 폴더 생성 확인

   $ python manage.py migrate

   `migrate` 명령은 `INSTALLED_APPS` 의 설정을 탐색하여, `mysite/settings.py` 의 데이터베이스 설정과 app 과 함께 제공되는 데이터베이스 migrations(나중에 설명하겠습니다) 에 따라, 필요한 데이터베이스 테이블을 생성합니다.

3. sqlite3 -> 현재 db 확인

```
$ sqlite3 db.sqlite3
sqllite> .tables
sqlite> .schema boards_board
```

4. django shell을 활용해서 query 날리기

```
$ python manage.py shell
>>> from boards.models import Board
>>> Board.objects.all()
<QuerySet []> # 응답은 QuerySet(Query를 가지고 Model을 조작할 수 있도록 도와줌)의 형태로 온다.

# insert 방법 1
```

```
>>> board = Board() # instance 생성
>>> board.title = 'first titile'
>>> board.content = 'first content'
>>> board.save() # 실제 DB에 저장
>>> board
<Board: Board object (1)>

# insert 방법 2
>>> board = Board(title='second title', content='second content')
>>> board.save()
>>> board
<Board: Board object (2)>

# insert 방법 3
>>> Board.objects.create(title='third', content='third')
<Board: Board object (3)>

>>> Board.objects.all()
<QuerySet [<Board: Board object (1)>, <Board: Board object (2)>, <Board: Board object
(3)>]>

>>> board = Board()
>>> board.title = 'fourth'
>>> board.content = 'fourth'
>>> board.id # 아직 id가 null
>>> board.title
'fourth'
>>> board.save()
>>> board.id # save 하면서 id 생성
4
>>> board.pk
4

# 데이터 유효성 검사
>>> board = Board()
>>> board.title = 'dsfasdfadsfadsfsd'
>>> board.full_clean() # 유효성 검사 (not null, max lenth 등)
Traceback (most recent call last):
  File "<console>", line 1, in <module>
  File "/home/ubuntu/.pyenv/versions/django-venv/lib/python3.6/site-
packages/django/db/models/base.py", line 1152, in full_clean
    raise ValidationError(errors)
django.core.exceptions.ValidationError: {'content': ['This field cannot be blank.']} #
content field : not null
```

5. toString 설정하기

[models.py]

```python
from django.db import models

# Create your models here.
class Board(models.Model):
    title = models.CharField(max_length=20)
    content = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self): # toString
        return '[' + str(self.id) + ']' + self.title + ':' + self.content
```

[shell]

```
>>> from boards.models import Board
>>> Board.objects.all()
<QuerySet [<Board: [1]first titile:first content>, <Board: [2]second title:second
content>, <Board: [3]third:third>, <Board: [4]fourth:fourth>]>
```

6. filtering

```
>>> Board.objects.create(title='third', content='haha')

<Board: [7]third:haha>
>>> boards = Board.objects.filter(title='third') # filtering
>>> boards
<QuerySet [<Board: [3]third:third>, <Board: [7]third:haha>]>

>>> board = Board.objects.filter(title='third').first() # method chaining. 메소드를 이음.
>>> board
<Board: [3]third:third>
```

7. indexing과 slicing

```
# indexing
>>> boards = Board.objects.all()[2] # .all()의 결과는 list라, 인덱싱과 슬라이싱이 가능.
>>> boards
<Board: [3]third:third>

# slicing
>>> boards = Board.objects.all()[1:4]
>>> boards
<QuerySet [<Board: [2]second title:second content>, <Board: [3]third:third>, <Board:
[4]fourth:fourth>]>
```

8. get, update, delete

```
# get
>>> board = Board.objects.get(pk=1)
>>> board
<Board: [1]first titile:first content>

# update
>>> board.title = 'abab'
>>> board.save()
>>> board = Board.objects.get(pk=1)
>>> board
<Board: [1]abab:first content>

# delete
>>> board.delete()
(1, {'boards.Board': 1})
>>> board = Board.objects.get(pk=1)
Traceback (most recent call last):
  File "<console>", line 1, in <module>
  File "/home/ubuntu/.pyenv/versions/django-venv/lib/python3.6/site-
packages/django/db/models/manager.py", line 82, in manager_method
    return getattr(self.get_queryset(), name)(*args, **kwargs)
  File "/home/ubuntu/.pyenv/versions/django-venv/lib/python3.6/site-
packages/django/db/models/query.py", line 399, in get
    self.model._meta.object_name
```

```
boards.models.Board.DoesNotExist: Board matching query does not exist.
```

9. Modeling 변경

   1. [models.py]

   ```python
   from django.db import models

   # Create your models here.
   class Board(models.Model):
       title = models.CharField(max_length=20)
       content = models.TextField()
       created_at = models.DateTimeField(auto_now_add=True)
       updated_at = models.DateTimeField(auto_now=True) # 추가

       def __str__(self):
           return '[' + str(self.id) + ']' + self.title + ':' + self.content
   ```

   2. migrate

   ```
   $ python manage.py makemigrations # model 설계도 생성
   $ python manage.py migrate # 설계도를 바탕으로 db에 적용
   ```