



Django

- web 이란
- Python,Django설치
- Django구조

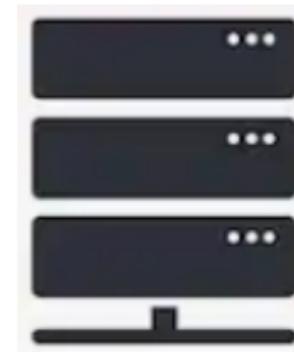
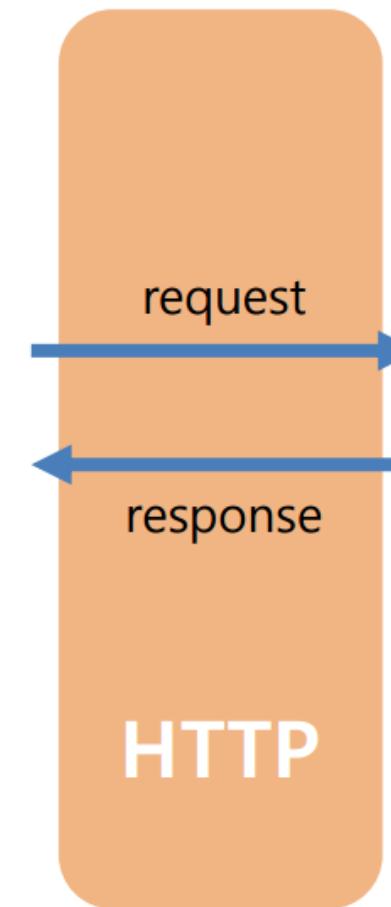




device
↓
software



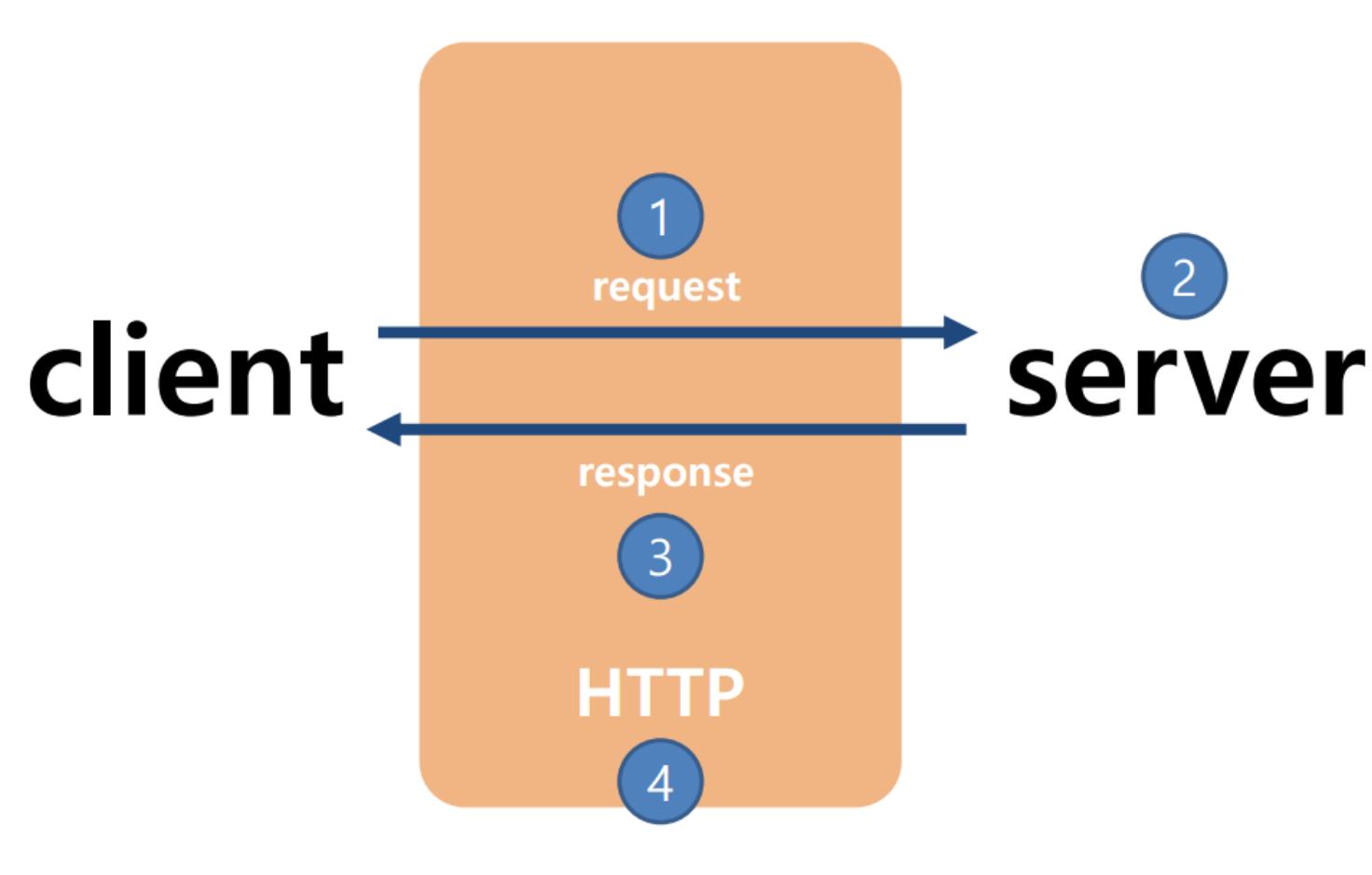
client

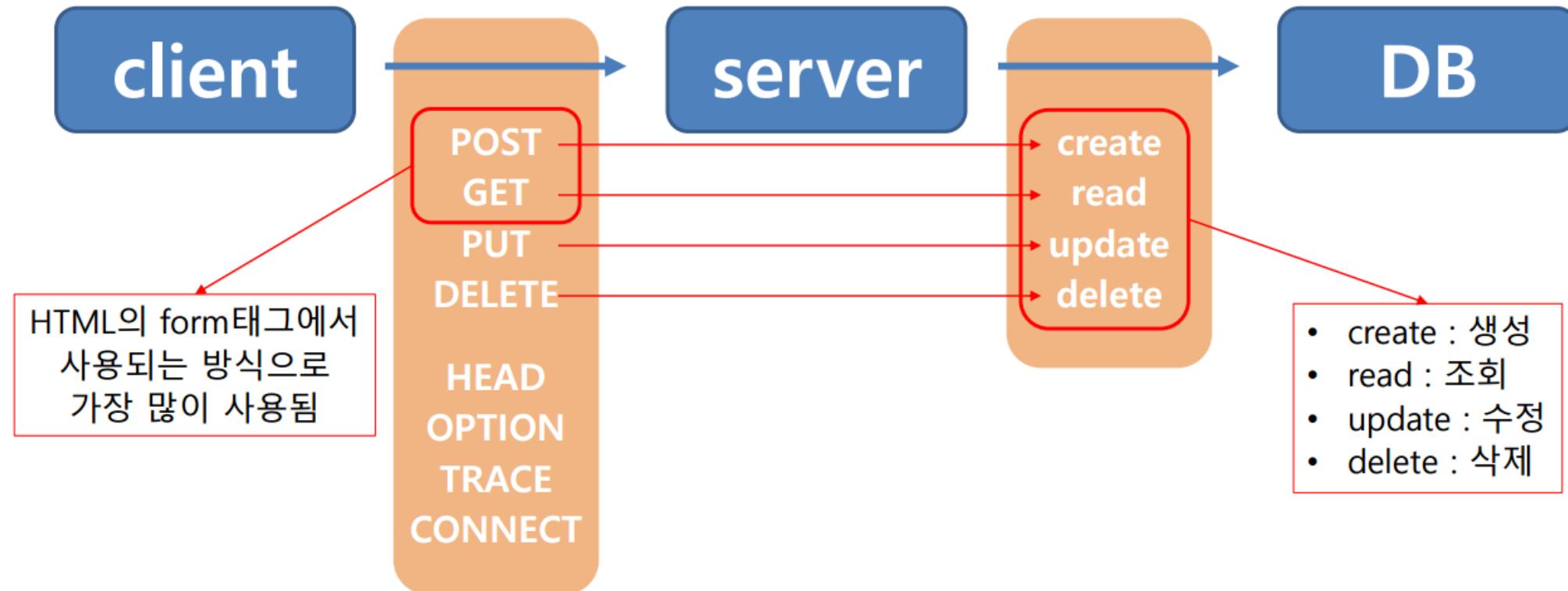


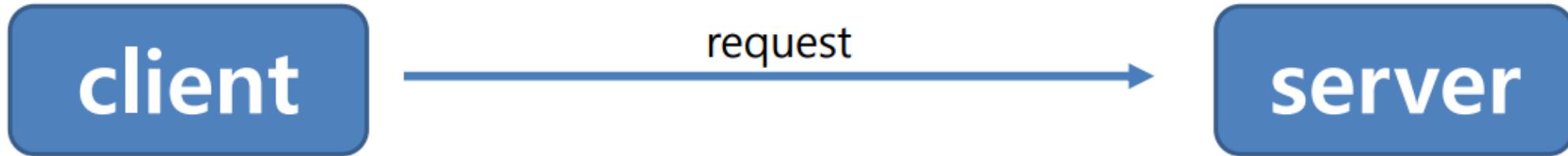
server

- 1 서버에 요청, 연결
- 2 요청에 따른 작업
- 3 클라이언트에 응답
- 4 연결 해제

연결을 유지하기 위해서
쿠키 또는 세션 이용







[GET]

- URL에 데이터가 노출됨
- 데이터 길이에 제한 있음
- 상대적으로 보안에 취약함

[POST]

- 요청 메시지에 데이터를 담음
- 상대적으로 보안에 강함
- django에서 주로 사용함

URL 기본 형태

https://search.naver.com/search.naver?query=weather

프로토콜

도메인(호스트)

경로

쿼리

REST URL 형태

https://search.naver.com/search/today/weather

프로토콜

도메인(호스트)

URL 맵핑 (데이터)스트링

https://search.naver.com/search/today/weather

URL 맵핑 (데이터)스트링

URLconf(urls.py)

search/<str:today>/weather	→	view.weather(request, today)
search/<str:sport>/movie	→	view.movie(request, sport)
search/<str:korea>/food	→	view.food(request, korea)
search/<str:map>/seoul	→	view.seoul(request, map)
함수 호출		



[웹서버]

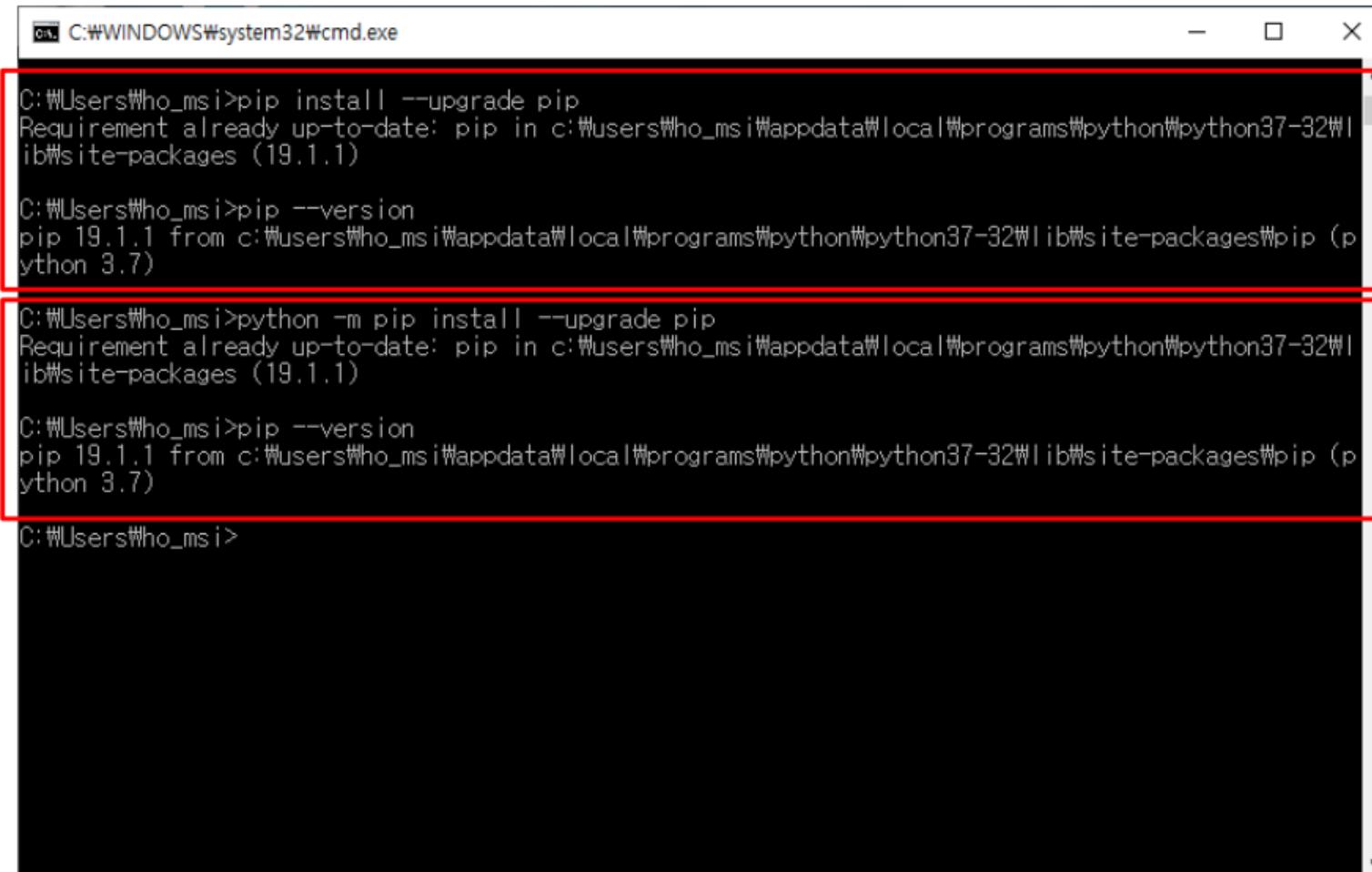
- 주로 정적인 데이터 요청 처리
- 동적인 데이터 요청 시 애플리케이션 서버에 전달

[애플리케이션서버]

- 주로 동적인 데이터 요청 처리
- Database 연동

2 python 패키지 | pip 업그레이드

pip : Python Install Packag의 약자로 PyPI(Python Package Index)의 SW패키지를 사용하기 위한 명령어



The screenshot shows a Windows Command Prompt window titled 'cmd.exe' with the path 'C:\WINDOWS\system32'. It contains two sets of command-line outputs, each enclosed in a red box and connected by a red arrow to its respective operating system name.

linux (Top Red Box):

```
C:\#Users\ho_ms\>pip install --upgrade pip
Requirement already up-to-date: pip in c:\#Users\ho_ms\#appdata\local\#programs\#python\#python37-32\lib\site-packages (19.1.1)

C:\#Users\ho_ms\>pip --version
pip 19.1.1 from c:\#Users\ho_ms\#appdata\local\#programs\#python\#python37-32\lib\site-packages\#pip (p
ython 3.7)
```

windows (Bottom Red Box):

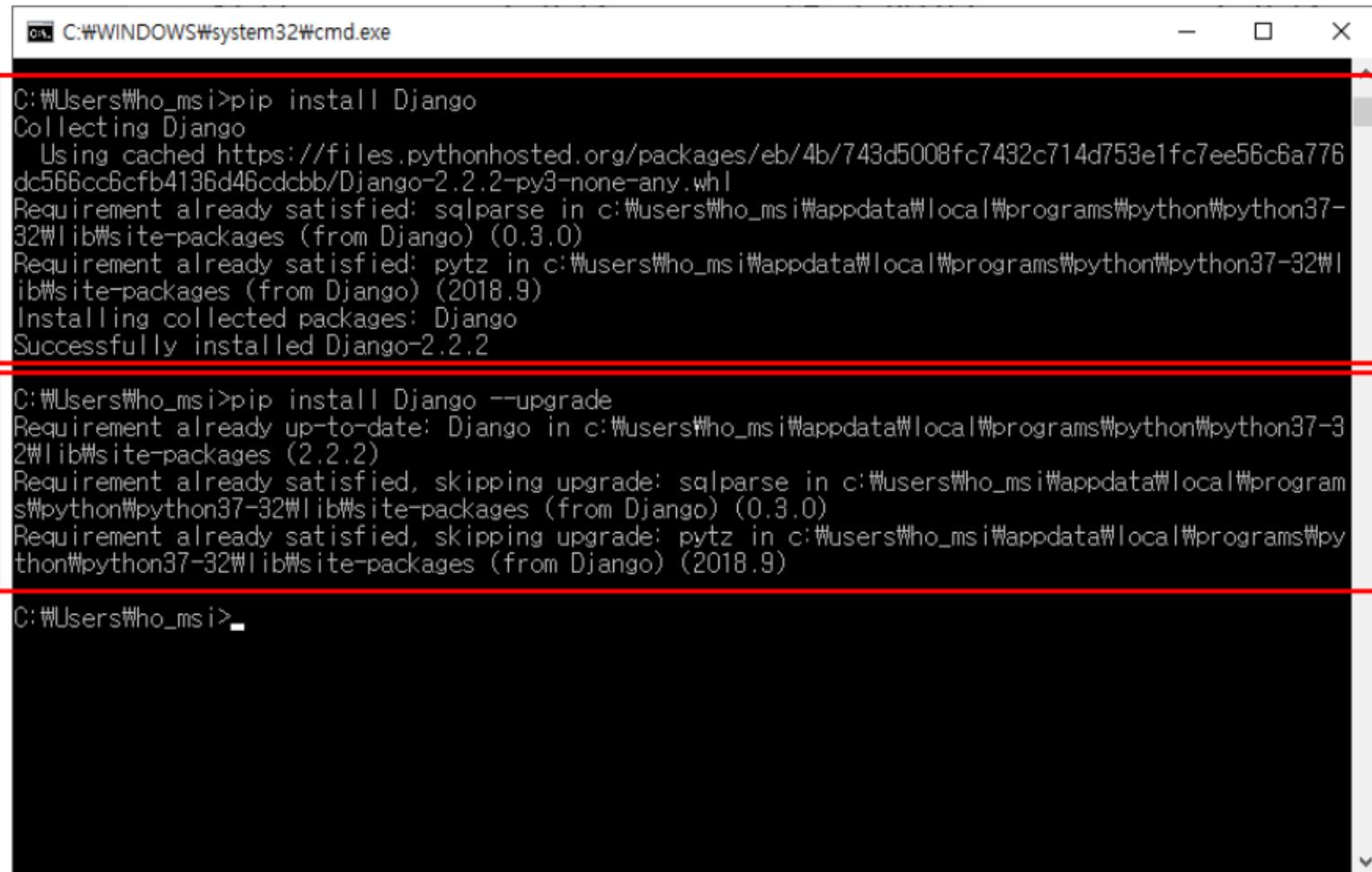
```
C:\#Users\ho_ms\>python -m pip install --upgrade pip
Requirement already up-to-date: pip in c:\#Users\ho_ms\#appdata\local\#programs\#python\#python37-32\lib\site-packages (19.1.1)

C:\#Users\ho_ms\>pip --version
pip 19.1.1 from c:\#Users\ho_ms\#appdata\local\#programs\#python\#python37-32\lib\site-packages\#pip (p
ython 3.7)

C:\#Users\ho_ms\>
```

2 Django 설치 및 업그레이드

pip를 이용한 설치 및 업그레이드



```
C:\Users\ho_ms\>pip install Django
Collecting Django
  Using cached https://files.pythonhosted.org/packages/eb/4b/743d5008fc7432c714d753e1fc7ee56c6a776dc566cc6cfb4136d46cdccb/Django-2.2.2-py3-none-any.whl
Requirement already satisfied: sqlparse in c:\Users\ho_ms\AppData\Local\Programs\Python\Python37-32\lib\site-packages (from Django) (0.3.0)
Requirement already satisfied: pytz in c:\Users\ho_ms\AppData\Local\Programs\Python\Python37-32\lib\site-packages (from Django) (2018.9)
Installing collected packages: Django
Successfully installed Django-2.2.2

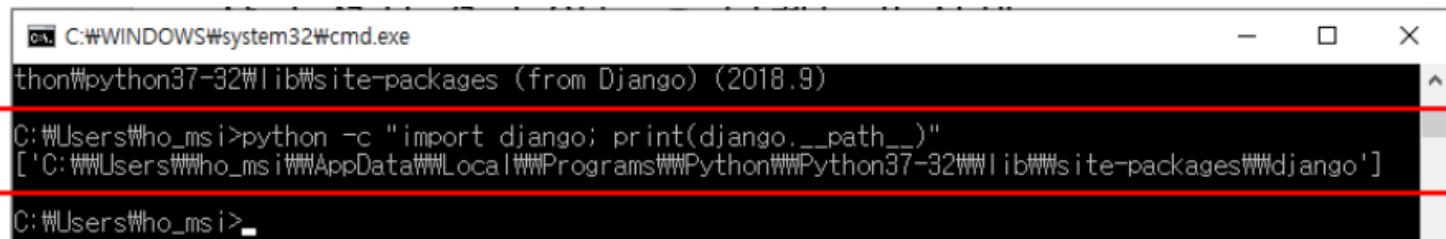
C:\Users\ho_ms\>pip install Django --upgrade
Requirement already up-to-date: Django in c:\Users\ho_ms\AppData\Local\Programs\Python\Python37-32\lib\site-packages (2.2.2)
Requirement already satisfied, skipping upgrade: sqlparse in c:\Users\ho_ms\AppData\Local\Programs\Python\Python37-32\lib\site-packages (from Django) (0.3.0)
Requirement already satisfied, skipping upgrade: pytz in c:\Users\ho_ms\AppData\Local\Programs\Python\Python37-32\lib\site-packages (from Django) (2018.9)

C:\Users\ho_ms\>
```

Django 설치

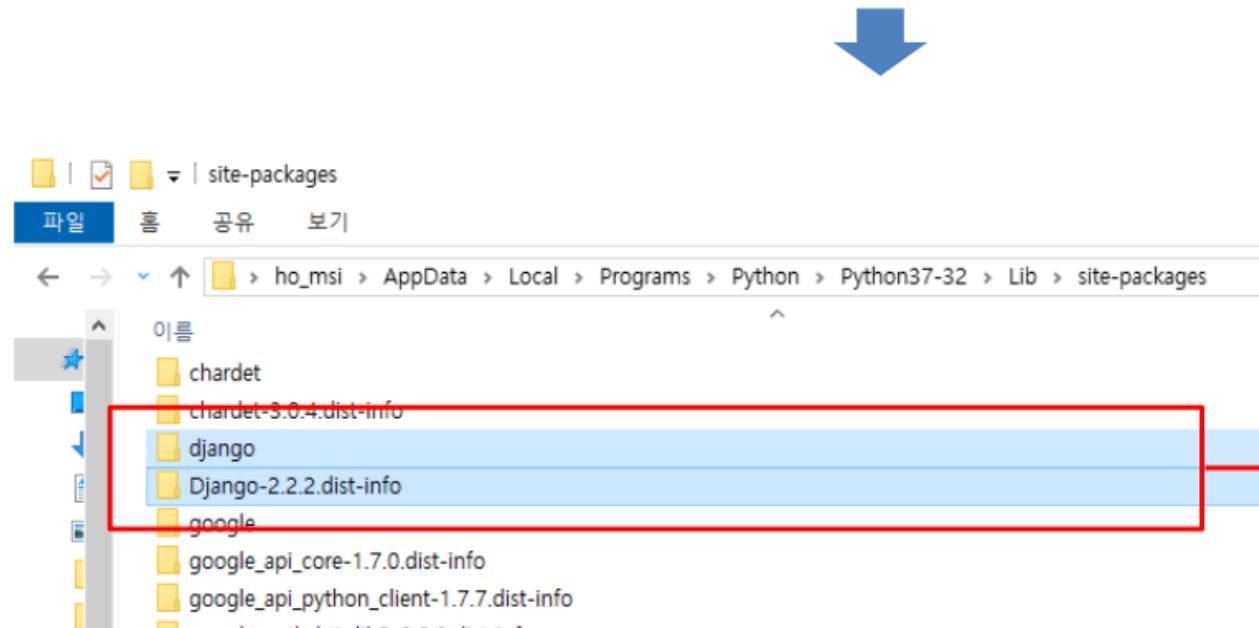
Django 업그레이드

장고 설치 경로 이동 후 디렉토리 삭제



```
C:\WINDOWS\system32\cmd.exe
thon#python37-32\lib\site-packages (from Django) (2018.9)
C:\Users\ho_msi>python -c "import django; print(django.__path__)"
['C:\Users\ho_msi\AppData\Local\Programs\Python\Python37-32\lib\site-packages\django']
```

Django 설치 경로 확인



Django 디렉토리 삭제

[참조] Django 가상환경 세팅 - Virtual environment

가상환경 – Virtualenv

한 서버에서 여러개의 프로젝트 진행 시,
여러 프로젝트의 파이썬 라이브러리 버전간 충돌을 방지하기 위해
프로젝트마다 독립적인 환경을 제공

해당 폴더 생성 후 > 폴더 위치 이동

```
pip install virtualenv          # 가상환경 설치  
python -m venv my_env          # 가상환경 생성 – 현재 폴더에 my_env 폴더 생성됨.
```

Django 를 설치, upgrade 설치를 해야 함.

가상환경 활성화 / 가상환경 해제

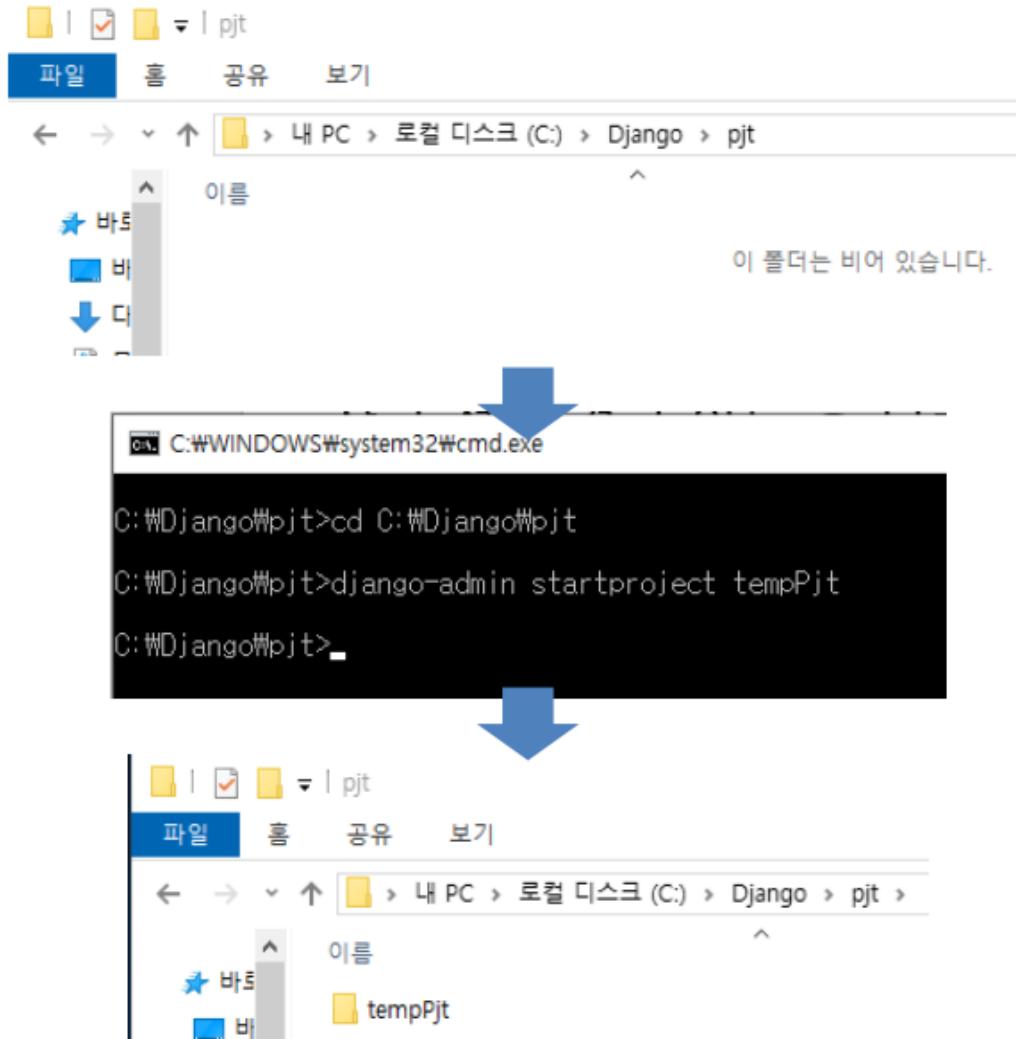
```
venv\Scripts\activate.bat      # my_env\Scripts\ 폴더에서 activate.bat 실행  
                                #(my_env) 표시가 나타남.  
(my_env) venv\Scripts\deactivate # deactivate 빠져나옴.
```



같은 환경에서
진행된다면
설정할 필요 없음

2

Django 프로젝트 생성



→ 경로 확인

1. 제작할 위치 폴더에서 프로젝트 생성

→ 프로젝트 생성

→ 프로젝트 확인

2

애플리케이션 생성

```

C:\Windows\system32\cmd.exe
C:\Django\pjt>dir tempPjt
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 685D-86B2

C:\Django\pjt\tempPjt 디렉터리

2019-06-27 오전 06:44 <DIR> .
2019-06-27 오전 06:44 <DIR> ..
2019-06-27 오전 06:44 648 manage.py
2019-06-27 오전 06:44 <DIR> tempPjt
    1개 파일 648 바이트
    3개 디렉터리 36,237,549,568 바이트 남음

C:\Django\pjt>cd tempPjt

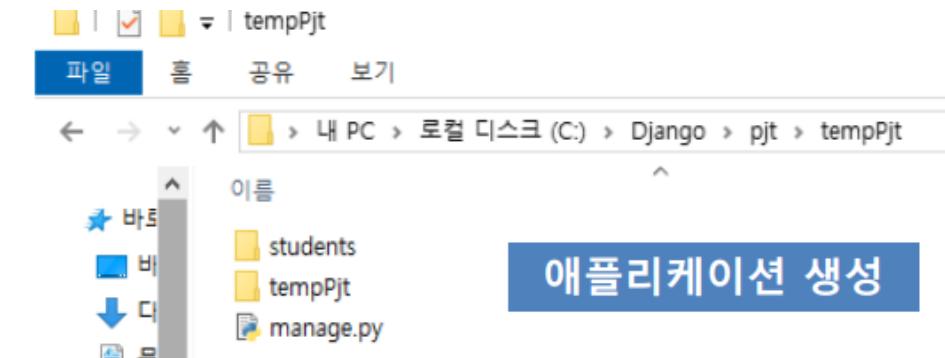
C:\Django\pjt\tempPjt>dir
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 685D-86B2

C:\Django\pjt\tempPjt 디렉터리

2019-06-27 오전 06:44 <DIR> .
2019-06-27 오전 06:44 <DIR> ..
2019-06-27 오전 06:44 648 manage.py
2019-06-27 오전 06:44 <DIR> tempPjt
    1개 파일 648 바이트
    3개 디렉터리 36,239,015,936 바이트 남음

C:\Django\pjt\tempPjt>python manage.py startapp students
C:\Django\pjt\tempPjt>

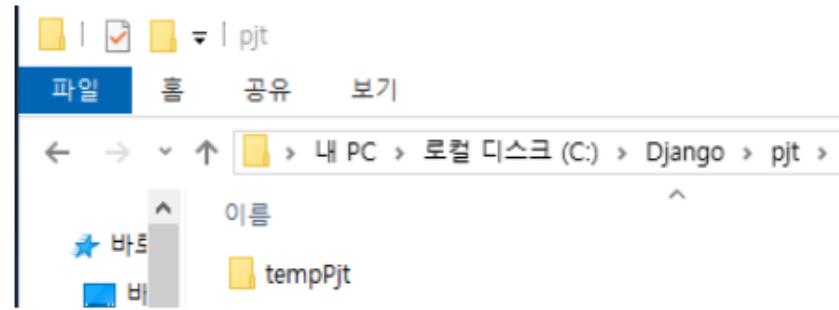
```



프로젝트(tempPjt) 디렉토리 이동

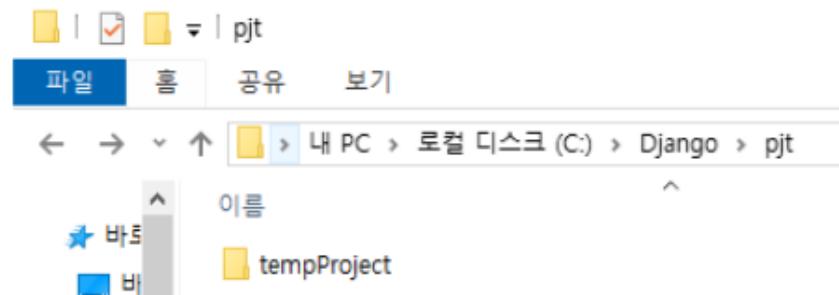
애플리케이션 생성

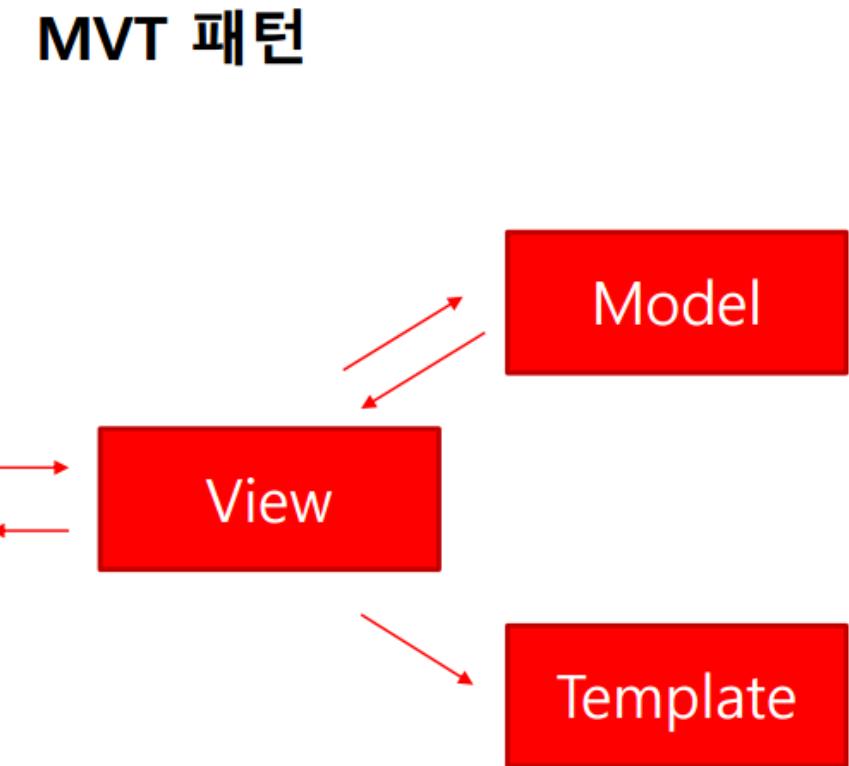
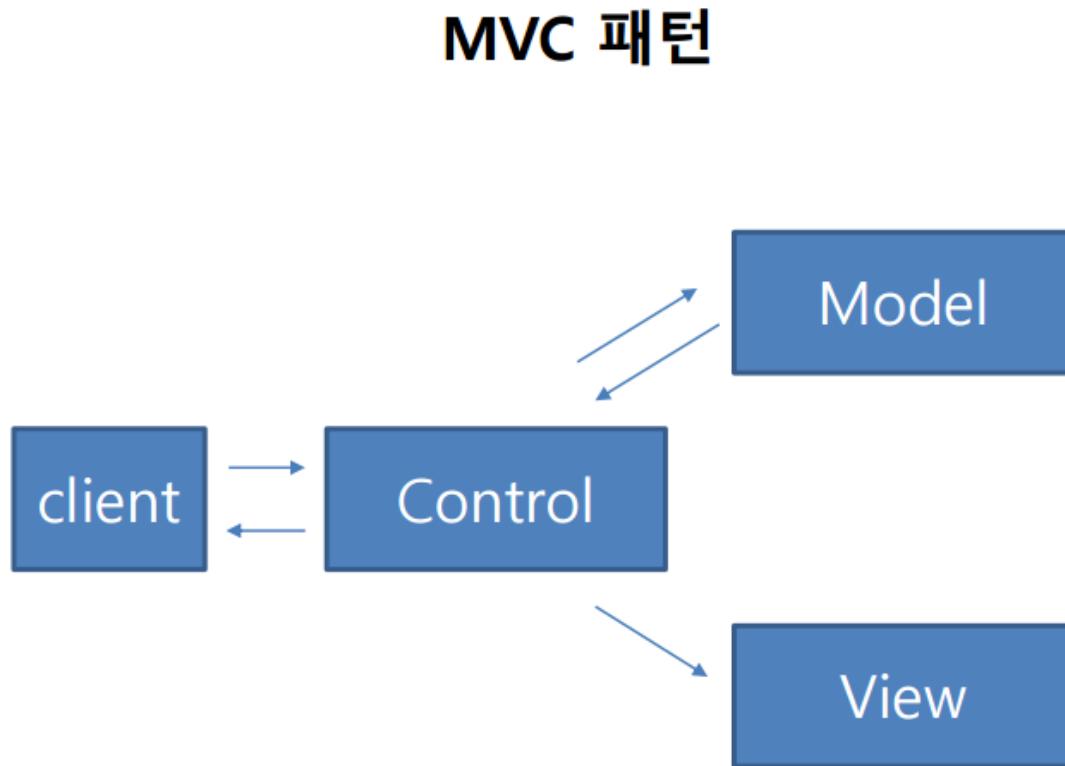
2 프로젝트 디렉토리 이름 변경



```
C:\#Django#\pjt#\tempPjt>cd ../  
C:\#Django#\pjt>move tempPjt tempProject  
1개의 디렉터리를 이동했습니다.  
C:\#Django#\pjt>
```

→ 디렉토리 이름 변경





2

프로젝트 디렉토리 구조

```
C:\WINDOWS\system32\cmd.exe
C:\Django\pjt>dir tempProject /s
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 685D-86B2

C:\Django\pjt\tempProject 디렉터리

2019-06-27 오전 06:50 <DIR> .
2019-06-27 오전 06:50 <DIR> ..
2019-06-27 오전 06:44 648 manage.py
2019-06-27 오전 06:50 <DIR> students
2019-06-27 오전 06:50 <DIR> tempPjt
1개 파일 648 바이트
```

```
C:\Django\pjt\tempProject\students 디렉터리

2019-06-27 오전 06:50 <DIR> .
2019-06-27 오전 06:50 <DIR> ..
2019-06-27 오전 06:50 66 admin.py
2019-06-27 오전 06:50 96 apps.py
2019-06-27 오전 06:50 <DIR> migrations
2019-06-27 오전 06:50 60 models.py
2019-06-27 오전 06:50 63 tests.py
2019-06-27 오전 06:50 66 views.py
2019-06-27 오전 06:50 0 __init__.py
6개 파일 351 바이트
```

C:\Django\pjt\tempProject\students 디렉터리

2019-06-27	오전 06:50	<DIR>
2019-06-27	오전 06:50	<DIR>
2019-06-27	오전 06:50	..
		0 __init__.py
		0 바이트

C:\Django\pjt\tempProject\tempPjt 디렉터리

2019-06-27	오전 06:50	<DIR>
2019-06-27	오전 06:50	<DIR>
2019-06-27	오전 06:44	..
2019-06-27	오전 06:44	3,211 settings.py
2019-06-27	오전 06:44	770 urls.py
2019-06-27	오전 06:44	407 wsgi.py
2019-06-27	오전 06:44	0 __init__.py
2019-06-27	오전 06:44	__pycache__
		4,388 바이트

C:\Django\pjt\tempProject\tempPjt__pycache__ 디렉터리

2019-06-27	오전 06:50	<DIR>
2019-06-27	오전 06:50	<DIR>
2019-06-27	오전 06:50	..
2019-06-27	오전 06:50	2,233 settings.cpython-37.pyc
2019-06-27	오전 06:50	128 __init__.cpython-37.pyc
		2,361 바이트

전체 파일:
 14개 파일 7,748 바이트
 14개 디렉터리 36,229,095,424 바이트 남음

URLconf – tempProject/tempPjt/urls.py

```
urlpatterns = [
    path('student/register/', views.student_register),
    path('student/search/', views.student_search),
    path('student/modify/', views.student_modify),
    path('student/delete/', views.student_delete)
]
```

path('student/register/', views.student_register),

URL(클라이언트 요청 URL)

view(함수 또는 메서드)

View(`views.py`)

View – `tempProject/students/views.py`

```
def registerStudent(request):
    # 데이터 베이스 등을 이용한 프로그램 실행 결과
    return HttpResponseRedirect()
```



클라이언트의 요청에 따른 애플리케이션 실행 결과를
Template(html), 에러메시지 등을 이용해서 클라이언트한테 response 한다.

Model(models.py)

Model - tempProject/students/models.py

ORM(Object Relational Mapping) 사용



class를 이용한 맵핑



```
class Student(models.Model):  
    s_name = models.CharField(max_length=30)  
    s_major = models.CharField(max_length=30)
```



students_student 테이블 생성

애플리케이션 이름 + _ + 클래스명(소문자)

Template(*.html)

1. **template** 파일은 직접 생성을 해야 함.

tempProject/tempPjt/settings.py에서 **Template** 파일 검색 디렉토리 정의

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'students.app.StudentConfig'  
    'board'
```

- * 넣지 않으면 구동이 안됨.
- * **fullName**을 다 적거나
app_name을 적어도 됨.

```
TEMPLATES = [  
    {  
        'BACKEND': 'django.template.backends.django.DjangoTemplates',  
        'DIRS': [],  
        'APP_DIRS': True,  
        'OPTIONS': {  
            'context_processors': [  
                'django.template.context_processors.debug',  
                'django.template.context_processors.request',  
                'django.contrib.auth.context_processors.auth',  
                'django.contrib.messages.context_processors.messages',  
            ],  
        },  
    },  
]
```

3 프로젝트 설정(setting.py, urls.py)

프로젝트 > setting.py, urls.py

```
## setting.py
```

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'students.apps.StudentsConfig',  
    'students.app.StudentConfig'  
    'students'  
]
```

```
## urls.py
```

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

프로젝트 설정(`views.py`)

프로젝트 > app > urls.py 설정 (app안 urls.py 생성후 세팅)
프로젝트 > views.py

students app > urls.py

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

app_name = 'students'
urlpatterns = [
    path("",views.index,name='index'),
    path('reg/',views.regStudent,name='reg'),
]
```

views.py

```
def index(request):
    return render(request,'students/index.html')

def regStudent(request):
    return
    render(request,'students/registerStudent.html')
```

프로젝트 설정(/tempPjt/settings.py)

settings.py – 프로젝트의 전체적인 설정을 담당한다.

- 모든 애플리케이션을 설정파일에 등록

1. students app 추가한 폴더안에

애플리케이션의 설정클래스를 이용한 등록

tempProject/students/apps.py

```
from django.apps import AppConfig
```

```
class StudentsConfig(AppConfig):
    name = 'students'
```

```
INSTALLED_APPS = [
```

```
    ...
    'django.contrib.staticfiles',
```

```
    'students.apps.StudentConfig',
```

```
]
```

1. app명.apps

2. apps.py의 클래스 명을 넣어야 함.

* app_name='board' 적어도 됨.

- 개발모드와 운영모드 설정

```
DEBUG = True
```

```
DEBUG = True #개발모드  
DEBUG = False #운영모드
```

```
ALLOWED_HOSTS = ['localhost', '127.0.0.1']
```

#개발모드

- 입력하지 않아도 자동으로 localhost로 정의

#운영모드

- 서버 IP주소 입력

* 실서버 운영시 : **False**, 서버 ip를 등록해야 서비스 됨.

3 기본 사용자 및 그룹 테이블 생성

1. Python 기본 db를 사용시 등록을 먼저 함

```
C:\Django\pjtt\tempProject>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 sessions.0001_initial... OK
```

사용자 및 그룹 테이블 생성



2. Admin 로그인시 보여지는 화면

AUTHENTICATION AND AUTHORIZATION

Groups

Users

+ Add Change

+ Add Change

3. db반영시 반드시 아래 명령어가 입력되어야 함.

```
C:\Django\pjtt\tempProject>python manage.py makemigrations
Migrations for 'students':
  students\migrations\0001_initial.py
    - Create model Student
```

```
C:\Django\pjtt\tempProject>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, students
Running migrations:
  Applying students.0001_initial... OK
```

데이터베이스 변경사항 반영

3 관리자 계정 생성 및 서버 구동

```
C:\Django\pjtt\tempProject>python manage.py createsuperuser
Username (leave blank to use 'ho-msi'): blogstudy
Email address: blogstudy@naver.com
Password:
Password (again):
The password is too similar to the username.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

관리자 계정 생성

1. email,password 등록 후
- admin로그인 할수 있음.

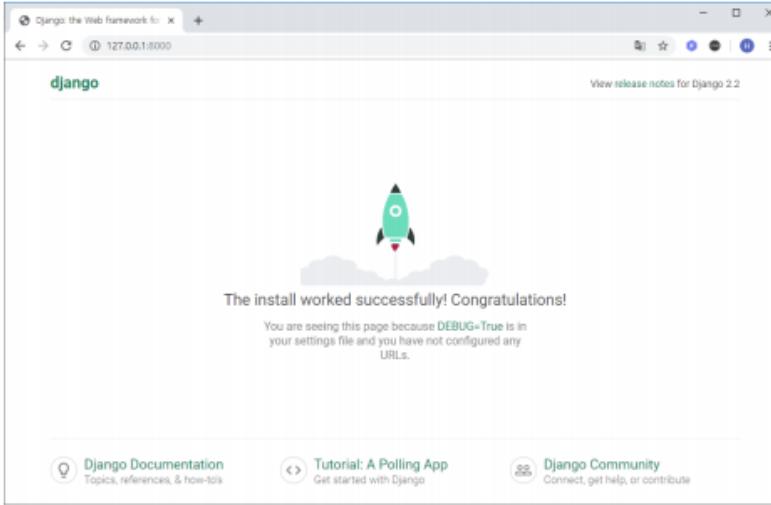
* 서버 화면창에서 print 출력 할때
python -u manage.py runserver 0.0.0.0:8000

서버 구동

```
C:\Django\pjtt\tempProject>python manage.py runserver 0.0.0.0:8000
Watching for file changes with StatReloader
Performing system checks...

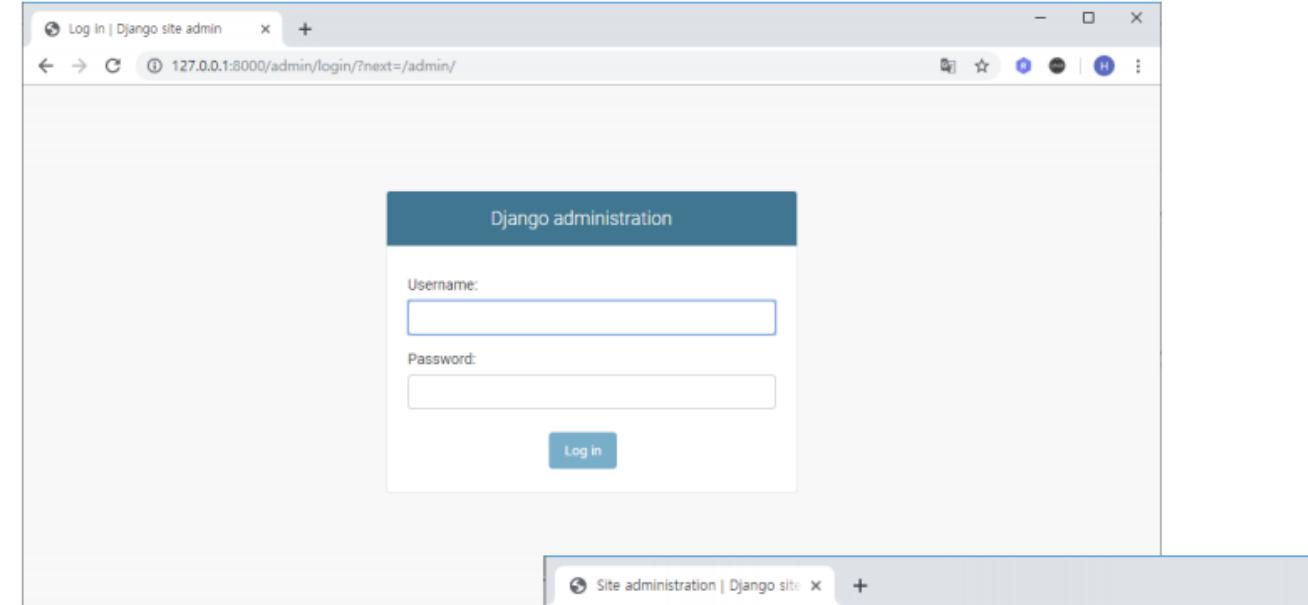
System check identified no issues (0 silenced).
June 27, 2019 - 10:07:22
Django version 2.2.2, using settings 'tempPjt.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CTRL-BREAK.
[27/Jun/2019 10:07:43] "GET / HTTP/1.1" 200 16348
[27/Jun/2019 10:07:43] "GET /static/admin/css/fonts.css HTTP/1.1" 200 423
[27/Jun/2019 10:07:43] "GET /static/admin/fonts/Roboto-Light-webfont.woff HTTP/1.1" 200 85692
[27/Jun/2019 10:07:43] "GET /static/admin/fonts/Roboto-Bold-webfont.woff HTTP/1.1" 200 86184
[27/Jun/2019 10:07:43] "GET /static/admin/fonts/Roboto-Regular-webfont.woff HTTP/1.1" 200 85876
Not Found: /favicon.ico
[27/Jun/2019 10:07:43] "GET /favicon.ico HTTP/1.1" 404 1973
[27/Jun/2019 10:07:58] "GET /admin/ HTTP/1.1" 302 0
[27/Jun/2019 10:08:03] "GET /admin/login/?next=/admin/ HTTP/1.1" 200 1823
[27/Jun/2019 10:08:03] "GET /static/admin/css/login.css HTTP/1.1" 200 1233
[27/Jun/2019 10:08:03] "GET /static/admin/css/responsive.css HTTP/1.1" 200 17944
[27/Jun/2019 10:08:03] "GET /static/admin/css/base.css HTTP/1.1" 200 16378
[27/Jun/2019 10:09:29] "POST /admin/login/?next=/admin/ HTTP/1.1" 302 0
[27/Jun/2019 10:09:29] "GET /admin/ HTTP/1.1" 200 3050
[27/Jun/2019 10:09:29] "GET /static/admin/css/dashboard.css HTTP/1.1" 200 412
[27/Jun/2019 10:09:29] "GET /static/admin/img/icon-changelink.svg HTTP/1.1" 200 380
```

3 관리자 계정 생성 및 서버 구동



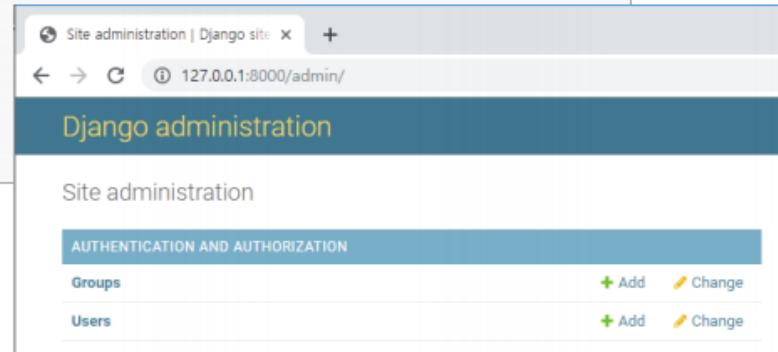
client 화면

[웹사이트 주소]
http://127.0.0.1:8000/



admin 화면

[웹사이트 주소]
http://127.0.0.1:8000/admin



models.py에 테이블 클래스 정의 후 admin.py에 등록

```
from django.db import models

class Student(models.Model):
    s_name = models.CharField(max_length=100)
    s_major = models.CharField(max_length=100)
    s_age = models.IntegerField(default=0)
    s_grade = models.IntegerField(default=0)
    s_gender = models.CharField(max_length=30)

    def __str__(self):
        return self.s_name
```

관리자페이지에서
보여지는 컬럼

student 테이블 생성을 위한 Student 클래스 정의

student table					
id(PK)	s_name	s_major	s_age	s_grade	s_gender



```
from django.contrib import admin
from students.models import Student

admin.site.register(Student)
```

admin.py에 Student 등록



```
C:\#Django#\pj#\tempProject>python manage.py makemigrations
Migrations for 'students':
  students\migrations\0001_initial.py
    - Create model Student

C:\#Django#\pj#\tempProject>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, students
Running migrations:
  Applying students.0001_initial... OK
C:\#Django#\pj#\tempProject>
```

DB변경사항 반영

4 models.py – Field Type

AutoField

```
#auto increment 되는 IntegerField  
id = models.AutoField(primary_key=True)
```

BooleanField

```
# 기본으로 null=False라서 true/false 가짐  
# default값을 설정하지 않으면 None값이 나오기 때문에 주의
```

CharField

```
# 문자열을 저장하는 필드로 max_length로 최대 길이를 제한할수 있음.
```

DateField

```
# datetime.date를 저장하는 필드  
# auto_now=True를 인자로 추가하면  
# Model.save로 저장할 때마다 현재 시간으로 갱신
```

DateTimeField : DateField 와 동일

4 models.py – Field Type

IntegerField

#32bit integer를 표현

TextField

CharField보다 큰 텍스트 설정

max_length 인자로 widget까지는 제어할 수 있음

model, database까지 제어되지는 않음.

Relationship Field – ForeignKey

외래 키를 지정하는 것을 통해 다대일 관계를 정의.

1명의 user는 여러 개의 상품을 가질수 있음 (다대일 관계)

다대일 관계에서는 2개 위치인수 필요

```
class Product(models.Model):
```

```
    owner = models.ForeignKey(User, on_delete=models.CASCADE)
```

ON DELETE CASCADE는 ForeignKey를 포함한 객체를 삭제.

예) 1개의 게시글에는 여러 개의 댓글 가능

- 1개의 게시글을 삭제할때
해당되는 댓글을 모두 삭제

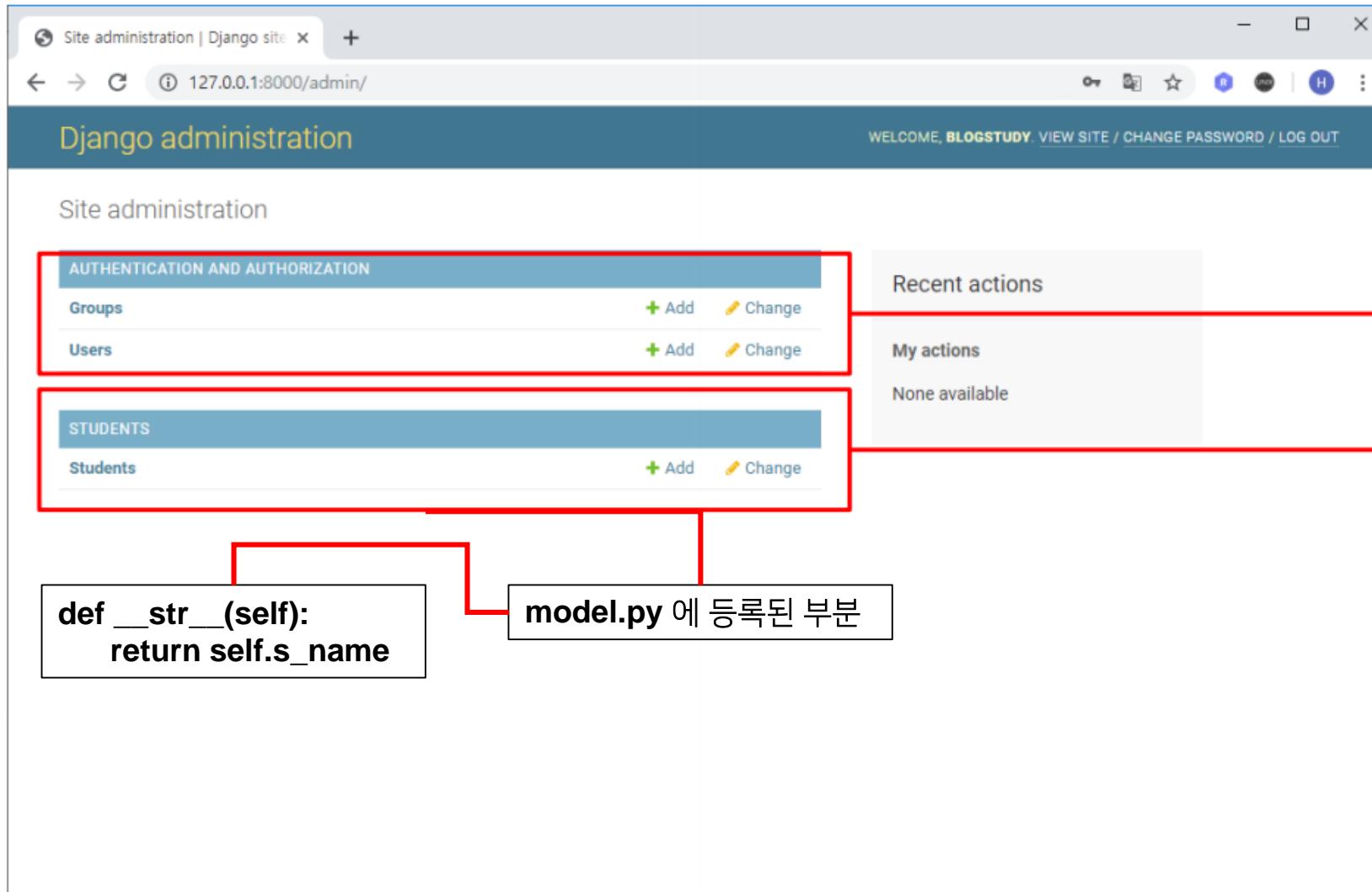
<https://ohgym.tistory.com/80?category=731677>

<https://brunch.co.kr/@ddangdol/5>

<https://076923.github.io/posts/Python-Django-11/>

4

테이블 생성



기본 사용자 및 그룹 테이블

Student 테이블

레코드 다루기

장고 shell모드 실행

```
C:\Django\pj1\tempProject>python manage.py shell
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>>
```

레코드 추가(create) – 데이터 생성

```
>>> from students.models import Student
>>> qs = Student(s_name='HongGilDong', s_major='computer', s_age=21, s_grade=2, s_gender='M')
>>> qs.save()
>>> -
```

The diagram illustrates the process of creating a new record. On the left, a red box highlights the Python code used to create a student object with name 'HongGilDong', major 'computer', age 21, grade 2, and gender 'M'. This code is connected by a blue arrow to the 'STUDENT' list in the Django admin interface. Another blue arrow connects the corresponding form fields (name, major, age, grade, gender) in the admin interface to the 'HongGilDong' entry in the list.

STUDENT

HongGilDong

1 student

S name: HongGilDong

S major: computer

S age: 21

S grade: 2

S gender: M

Delete

레코드 읽기(read) – 데이터 검색

```
>>> Student.objects.all()
<QuerySet [<Student: HongGiDong>, <Student: HongGiJa>, <Student: HongGiJaSoon>]>
>>> qs = Student.objects.all()
>>> print(qs)
<QuerySet [<Student: HongGiDong>, <Student: HongGiJa>, <Student: HongGiJaSoon>]>
>>> type(qs)
<class 'django.db.models.query.QuerySet'>
>>>
```

데이터 전체 : QuerySet 타입으로 반환

```
>>> qs = Student.objects.get(s_name='HongGiDong')
>>> qs
<Student: HongGiDong>
>>> type(qs)
<class 'students.models.Student'>
>>>
```

데이터 한개 : Student 타입으로 반환

레코드 읽기(read) – 필드 데이터 검색

데이터 다수 : 첨자([])를 이용한 접근

```
>>> qs = Student.objects.all()
>>> qs
<QuerySet [<Student: HongGiDong>, <Student: HongGiJa>, <Student: HongGiJaSoon>]>
>>> qs[1]
<Student: HongGiJa>
>>> qs[1].s_name
'HongGiJa'
>>> qs[1].s_age
21
>>> qs[1].s_major
'computer'
```

데이터 한개 : '.'를 이용한 속성 접근

```
>>> qs = Student.objects.get(s_name='HongGiDong')
>>> qs.s_name
'HongGiDong'
>>> qs.s_age
21
>>> qs.s_major
'computer'
>>>
```

레코드 읽기(read) – 데이터 필터(filter)

```
>>> qs = Student.objects.filter(s_age__lt=22)
>>> qs
<QuerySet [<Student: HongGi | Dong>]>
>>> qs = Student.objects.filter(s_age__gt=22)
>>> qs
<QuerySet [<Student: HongGi | JaSoon>]>
>>> qs = Student.objects.filter(s_age__lte=22)
>>> qs
<QuerySet [<Student: HongGi | Dong>, <Student: HongGi | Ja>]>
>>> qs = Student.objects.filter(s_age__gte=22)
>>> qs
<QuerySet [<Student: HongGi | Ja>, <Student: HongGi | JaSoon>]>
>>>
```

_lt	~보다 작다
_lte	~보다 작거나 같다
_gt	~보다 크다
_gte	~보다 크거나 같다
_isnull	~ null인 자료 검색
_contains	특정 문자열을 포함하는 자료 검색
_startswith	특정 문자열로 시작하는 자료 검색
_endswith	특정 문자열로 끝나는 자료 검색

레코드 읽기(read) – 데이터 정렬

```
<QuerySet [<Student: HongGi|Dong>, <Student: HongGi|Ja>, <Student: HongGi|JaSoon>]>
>>> qs = Student.objects.order_by('s_age')
>>> qs
<QuerySet [<Student: HongGi|Dong>, <Student: HongGi|Ja>, <Student: HongGi|JaSoon>]>
>>> qs = Student.objects.order_by('-s_age')
>>> qs
<QuerySet [<Student: HongGi|JaSoon>, <Student: HongGi|Ja>, <Student: HongGi|Dong>]>
>>>
```

order_by(' ')	오름차순
order_by('-')	내림차순

레코드 업데이트(update) – 데이터 수정

```
>>> qs = Student.objects.get(s_name='HongGi|Dong')
>>> qs
<Student: HongGi|Dong>
>>> qs.s_major = 'mathematics'
>>> qs.save()
>>> qs.s_major
'mathematics'
```

S major:

computer



S major:

mathematics

레코드 삭제(delete) – 데이터 삭제

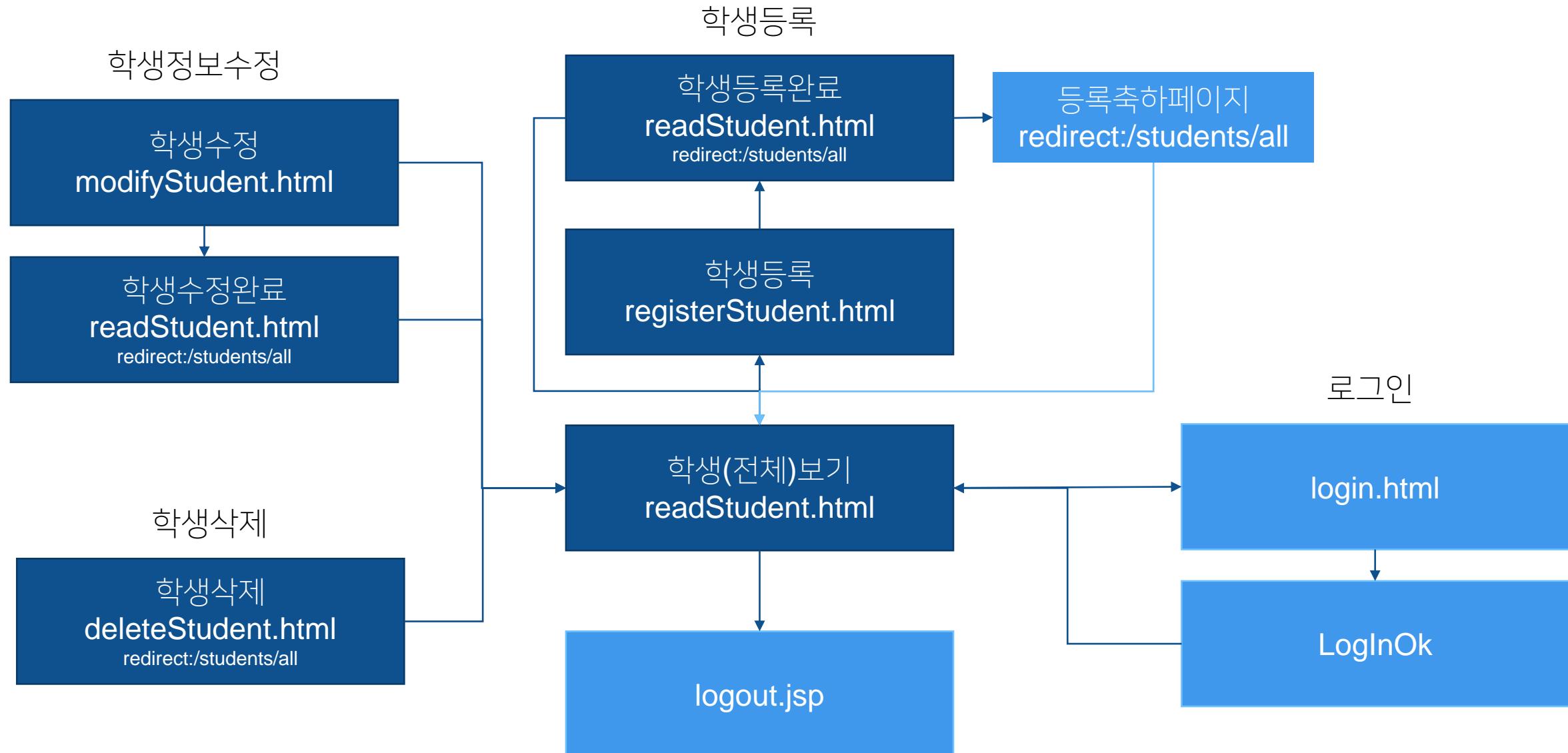
```
>>> qs = Student.objects.filter(s_age__gte=22)
>>> qs
<QuerySet [, <Student: HongGiJaSoon>]>
>>> qs.delete()
(2, {'students.Student': 2})
```

Action: Go 0 of 1 selected

<input type="checkbox"/> STUDENT
<input type="checkbox"/> HongGilDong

1 student

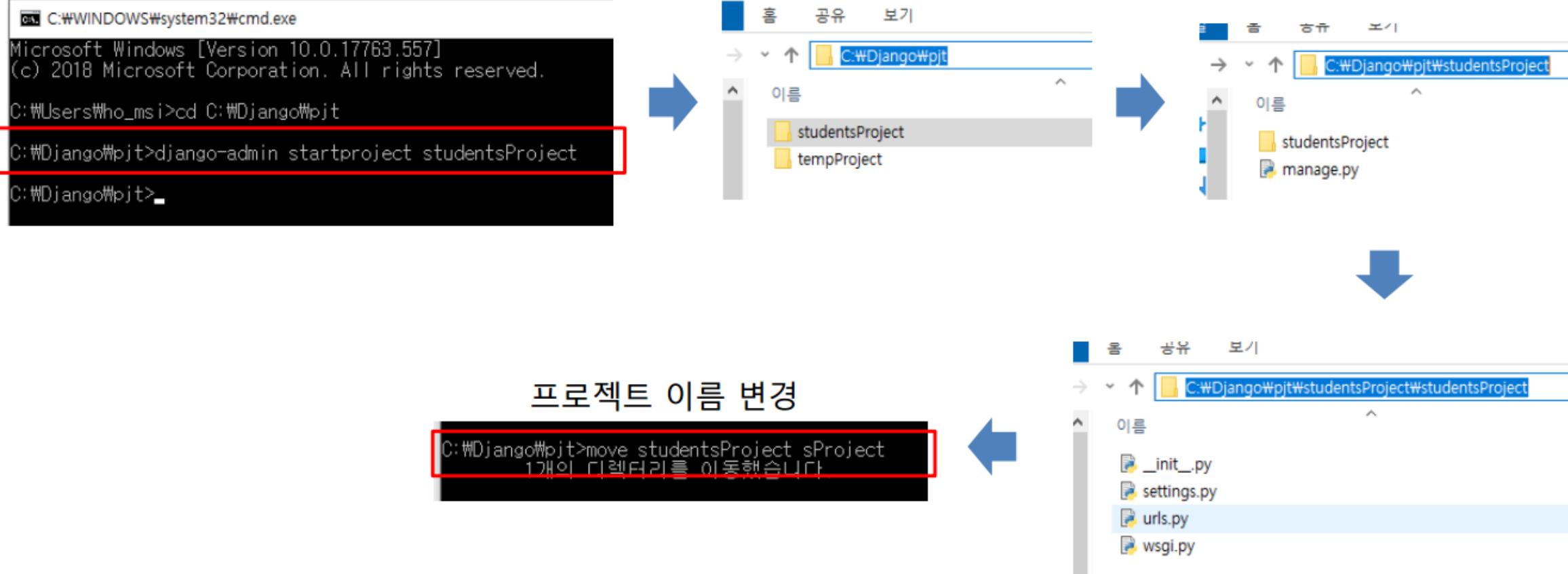
	URL	view	template	redirection
학생등록	/students/reg	regStudent()	registerStudent.html	
학생(전체)보기		regConStudent()		/students/all
학생(전체)보기	/students/all	reaStudentAll()	readStudents.html	
학생수정	/students/<str:s_name>/mod	reaStudentOne()	modifyStudent.html	
		modConStudent()		reaStudentAll()
학생삭제	/students/<str:s_name>/del	delStudent()	deleteStudent.html	



실습페이지

5

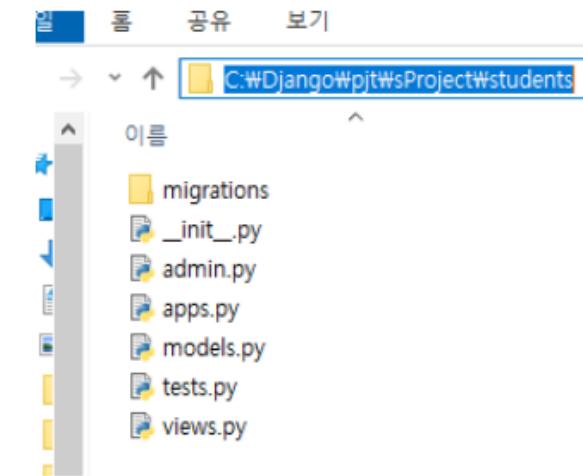
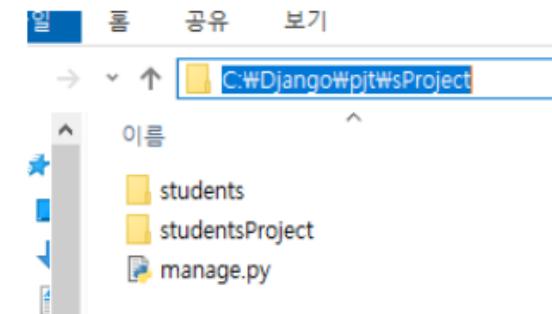
프로젝트 생성



5

애플리케이션 생성

```
C:\Django\pjt>cd sProject  
C:\Django\pjt\sProject>python manage.py startapp students  
C:\Django\pjt\sProject>
```



5 프로젝트 설정 변경(settings.py)

students/apps.py

```
apps.py
1 from django.apps import AppConfig
2
3
4 class StudentsConfig(AppConfig):
5     name = 'students'
```

애플리케이션 등록

studentsProject/settings.py

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

studentsProject/settings.py

```
#TIME_ZONE = 'UTC'
TIME_ZONE = 'Asia/Seoul'
```

타임존 변경

```
## 한국어 변경시
# LANGUAGE_CODE = 'ko-kr'
TIME_ZONE = 'Asia/Seoul'
```

5 데이터베이스 만들기(models.py)

students/models.py

```
models.py
1 from django.db import models
2
3 # Create your models here.
4 class Student(models.Model):
5     s_name = models.CharField(max_length=100)
6     s_major = models.CharField(max_length=100)
7     s_age = models.IntegerField(default=0)
8     s_grade = models.IntegerField(default=0)
9     s_gender = models.CharField(max_length=30)
10
11    def __str__(self):
12        return self.s_name
```

s_name : 이름(char)
s_major : 전공(char)
s_age : 나이(int)
s_grade : 학년(int)
s_gender : 성구분(char)

날짜 자동입력
from datetime import datetime
models.DateTimeField(default=datetime.now(),
blank=True)

students/admin.py

```
admin.py
1 from django.contrib import admin
2 from students.models import Student
3
4 # Register your models here.
5 admin.site.register(Student)
6
```

Admin에 table 적용

5 데이터베이스 만들기(models.py)

```
C:\Django\pjts\Project>python manage.py makemigrations
Migrations for 'students':
  students\migrations\0001_initial.py
    - Create model Student

C:\Django\pjts\Project>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, students
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 sessions.0001_initial... OK
  Applying students.0001_initial... OK

C:\Django\pjts\Project>
```

데이터베이스 반영

5 관리자 계정 생성 및 웹서버 실행

```
C:\Django\pjtsProject>python manage.py createsuperuser
Username (Leave blank to use 'ho_ms1'): blogstudy
Email address: blogstudy@naver.com
Password:
Password (again):
The password is too similar to the username.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.

C:\Django\pjtsProject>
```

관리자 계정
계정 : blogstudy
메일 : blogstudy@naver.com
암호 : blogstudy1234



```
C:\Users\ho_ms1>cd C:\Django\pjtsProject
C:\Django\pjtsProject>python manage.py runserver 0.0.0.0:8000
Watching for file changes with StatReloader
Performing system checks...

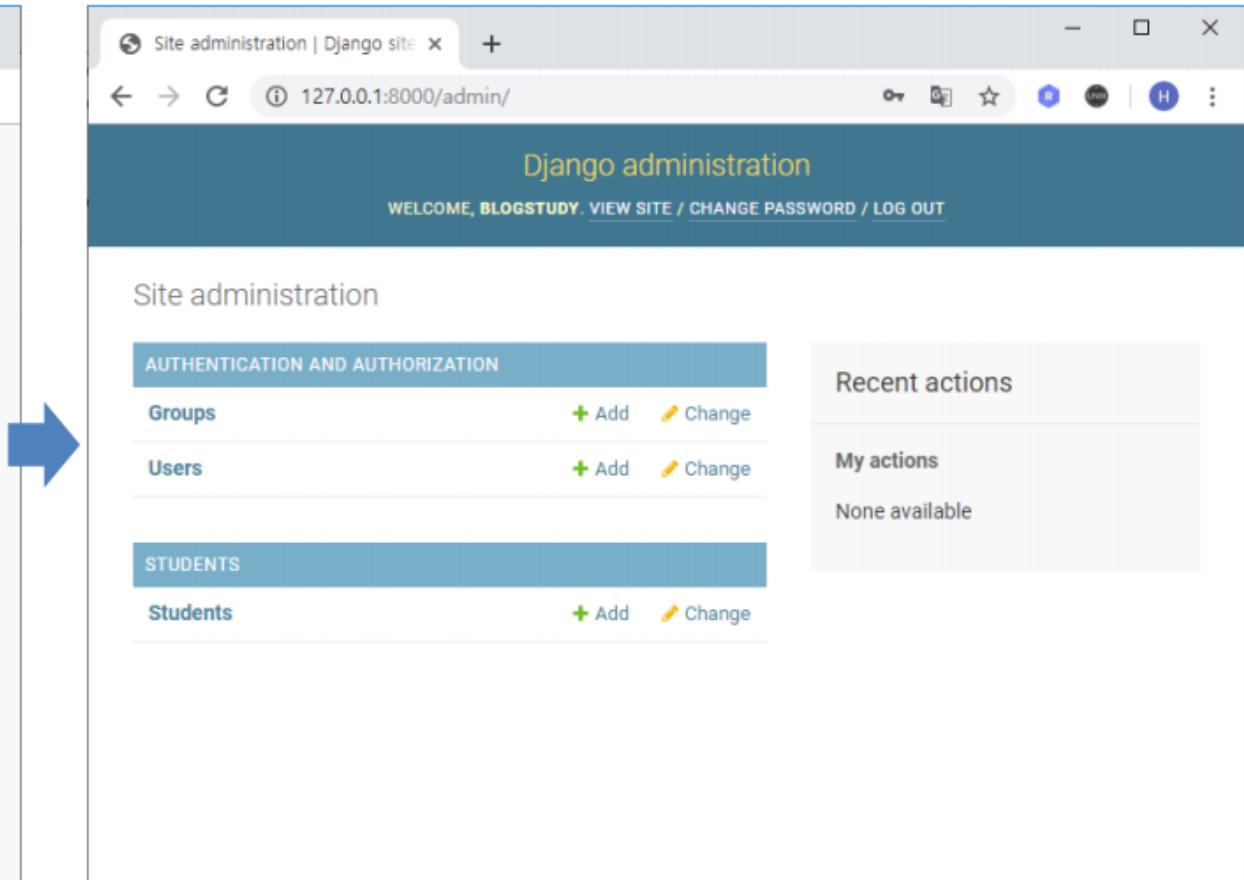
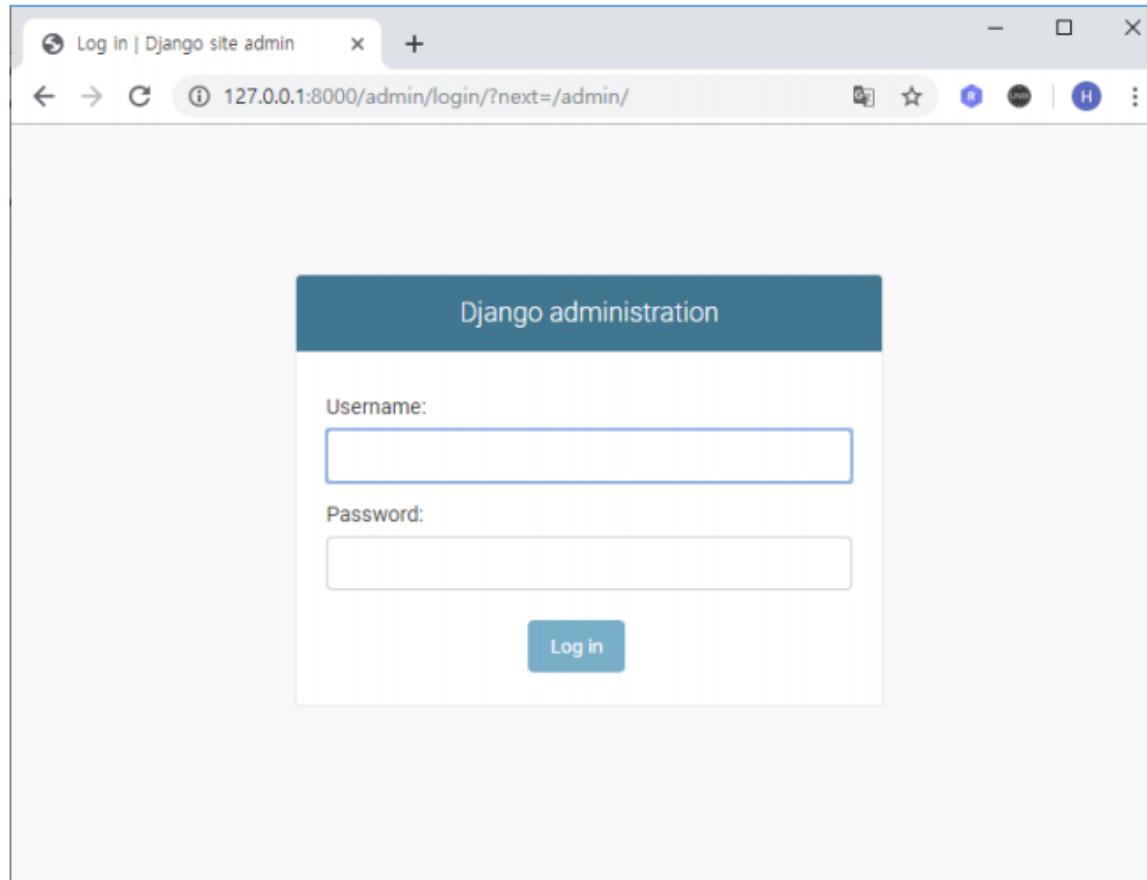
System check identified no issues (0 silenced).
June 28, 2019 - 10:57:02
Django version 2.2.2, using settings 'studentsProject.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CTRL-BREAK.
```

웹서버 실행

5

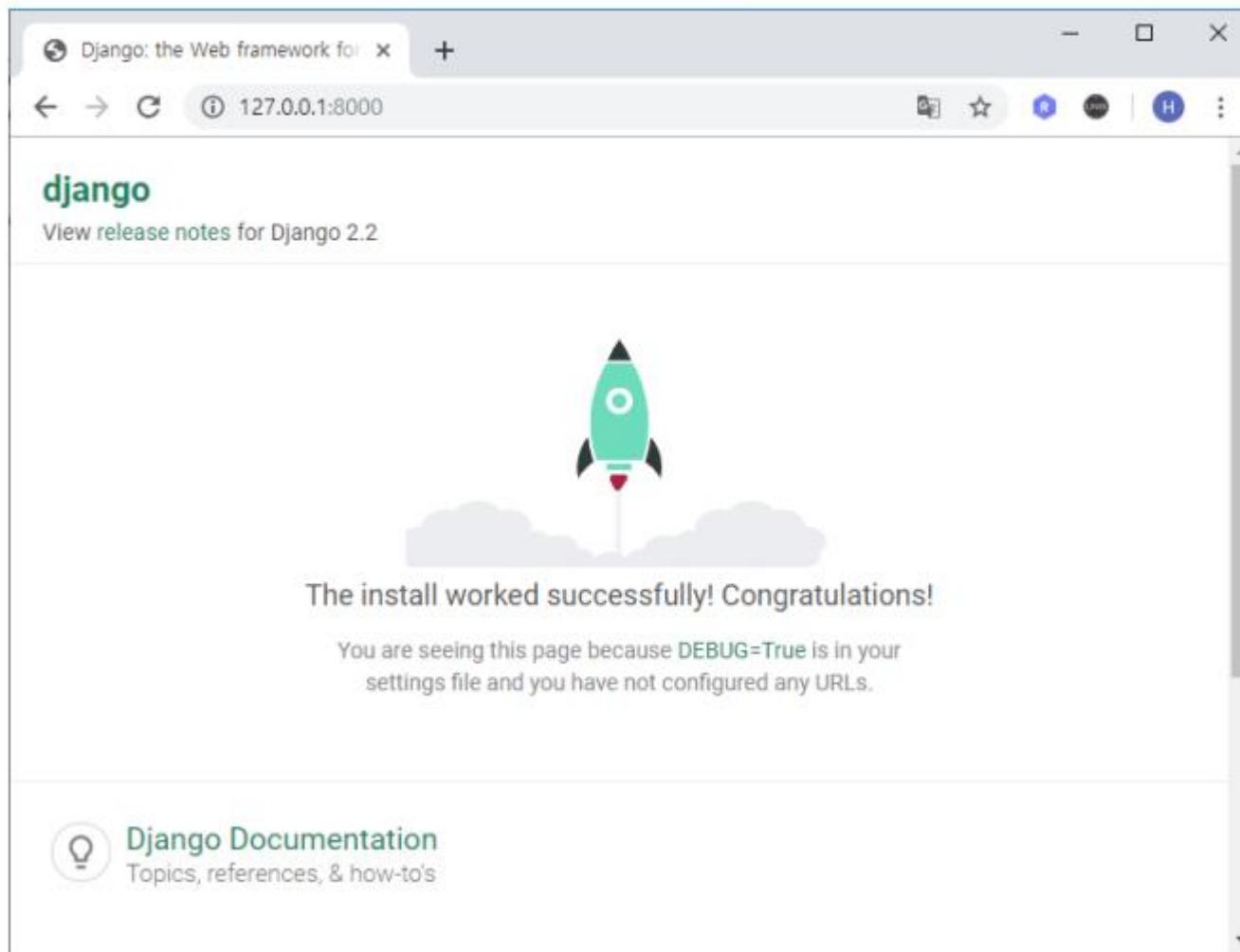
관리자페이지 접속

http://127.0.0.1:8000/admin/



5 Django 인트로(시작) 페이지 접속

http://127.0.0.1:8000



6 학생 등록 및 리스트 페이지 만들기

학생 등록 페이지

1

URLconf

- studentsProject/urls.py
- students/urls.py

프로젝트, app urls.py 부분 추가

2

views

- students/views.py



3

template

- students/templates/students/registerStudent.html

The screenshot shows a browser window with a title bar labeled 'Title'. The address bar contains the URL '127.0.0.1:8000/students/reg/'. Below the address bar, there is a navigation bar with icons for back, forward, and search. The main content area is titled 'Register Student'. It contains five input fields with labels: 'name :', 'major :', 'age :', 'grade :', and 'gender :'. Each label is followed by a horizontal input field. At the bottom right of the form is a 'Register' button.

학생 등록 페이지 – urls.py

app 폴더 > urls.py 추가

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

app_name = 'students'
urlpatterns = [
    path('reg/', views.regStudent, name='reg'),
    path('regCon/', views.regConStudent, name='regCon'),
    path('modCon/', views.modConStudent, name='modCon'),
    path('all/', views.reaStudentAll, name='stuAll'),
    path('<str:name>/det/', views.detStudent, name='stuDet'),
    path('<str:name>/mod/', views.reaStudentOne, name='stuMod'),
    path('modCon', views.modConStudent, name='modCon'),
    path('<str:name>/stuDel', views.delConStudent, name='stuDel'),
]
```

1. 어플리케이션 urls.py 추가시켜줌

urls로 들어오는 모든 링크를 views로 연결

하드코딩 url 보다
url의 name 값을 사용하는 것이 유지보수에 유리

```
<a href="/students/reg/{{question.id}}">
<a href="{% url 'students:reg' question.id%}">
```

6

학생 등록 페이지 – students:regCon

프로젝트 파일 > students > urls.py

프로젝트 파일 > students > views.py

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

urls.py

```
app_name = 'students'
urlpatterns = [
    path('reg/',views.regStudent,name='reg'),
    path('regCon/',views.regConStudent,name='regCon'),
```

```
path('all/',views.reaStudentAll,name='stuAll'),
path('modCon/',views.modConStudent,name='modCon'),
path('<str:name>/det/',views.detStudent,name='stuDet'),
path('<str:name>/mod/',views.reaStudentOne,name='stuMod'),
path('modCon/', views.modConStudent, name='modCon'),
path('<str:name>/stuDel/', views.delConStudent,
name='stuDel'),
]
```

등록 버튼 클릭시 이동페이지 연결

```
2 from django.http import HttpResponseRedirect
from django.shortcuts import redirect, render
from django.urls import reverse
from students.models import Student
def regConStudent(request):
```

views.py

```
name = request.POST['name']
major = request.POST['major']
age = request.POST['age']
grade = request.POST['grade']
gender = request.POST['gender']
```

form에서 들어온 데이터 연결

```
qs =
Student(s_name=name,s_major=major,s_age=age,s_grade=grade,
s_gender=gender)
qs.save()
```

db>stduent테이블 1행 추가

```
return HttpResponseRedirect(reverse('students:stuAll'))
```

6

학생 등록 페이지 – RegisterStudent.html

프로젝트 파일 > students > RegisterStudent.html

3

```
<!DOCTYPE html>
<html lang="ko">
<head>
<script>
    function cancel(){
        location.href="{% url 'students:stuAll' %}";
    }
</script>
```

취소 버튼 클릭시 **list**페이지로 이동

```
<h2>Register Student</h2>
<form action="{% url 'students:regCon' %}" method="post">
    {% csrf_token %}
    name:<input type="text" name="name"><br>
    major:<input type="text" name="major"><br>
    age:<input type="text" name="age"><br>
    grade:<input type="text" name="grade"><br>
    gender:<input type="text" name="gender"><br>
    <input type="submit" value="등록">
<input type="button" onclick="cancel()" value="취소"><br>
</form>
</body>
</html>
```

등록 버튼 클릭시 **action** 페이지 이동

form 데이터를 암호화해서 전송

6 [참조] checkbox

```
user_hobby = request.POST['hobby']
list_item = request.POST.getlist('hobby')
print("type : ", type(user_hobby))
print("hobby : ", user_hobby)
print("list_item type : ", type(list_item))
print("list_item : ", list_item)
```

checkbox

타입 : str (마지막 checkbox 1개만 출력)

타입 : list (선택된 모든 checkbox 값 출력)

```
## form.html
<input type="checkbox" name="hobby" value="game"
{%- if 'game' in uid %} checked {%- endif %}>
게임<br>
```

리스트를 문자열로 변경

join : list -> str

“.join(list_item)

‘,’.join(list_item)

문자열을 리스트로 변경

split : str -> list

list = str.split(',')
list = str.split(‘,’)

6

[참조] index페이지 연결 > http://127.0.0.1:8000

프로젝트 폴더 > urls.py 추가

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

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

어플리케이션 urls.py 연결 시켜줌.

students > urls.py 에서 설정

```
from django.urls import path  
from . import views  
  
app_name = 'students'  
urlpatterns = [  
    path("", views.index, name='index'),  
]
```

- * 127.0.0.1:8000 으로 들어올때
 - 프로젝트 파일 > urls.py에서 students 으로 이동
 - 어플케이션에서 '' 설정해 줌.

6 [참조] CSS 연결

css, js, images 파일 연결 방법 : 프로젝트 파일 > setting.py 설정
자동 import : vs-code **ctrl + .**

```
import os ##추가시켜줌.
```

```
# setting.py > static 파일 설정  
STATIC_URL = 'static/'
```

```
# 이 부분을 추가
```

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

```
## html에서 css, image 코드
```

```
<link href="{% static 'css/style.css' %}" rel="stylesheet">  

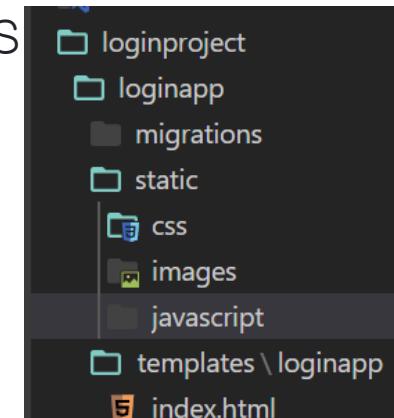
```

* 실제로 staticfiles 앱은
디버그 모드가 **True** 일 때만 실행
(개발모드일때 실행)

디버그 모드가 **false** 변경 시
웹 서버가 대신 처리, 웹서버는 위치를 알지 못함.
STATIC_ROOT 경로에 모아 관리해야 함.

```
STATIC_ROOT = os.path.join(BASE_DIR,  
'staticfiles/') STATIC_URL = '/static/'  
STATICFILES_DIRS = [ os.path.join(BAS  
'static'), ]
```

프로젝트 폴더에서
static 폴더 생성
- CSS
- images ## 폴더
생성



한줄 설명문 처리

[HTML]

```
<!-- 이곳에 작성... -->
```

[Javascript]

```
// 이곳에 작성...
```

[Django Template]

```
{# 이곳에 작성... #}
```

여러 줄 설명문 처리

[HTML]

```
<!--  
이곳에 작성  
-->
```

[Javascript]

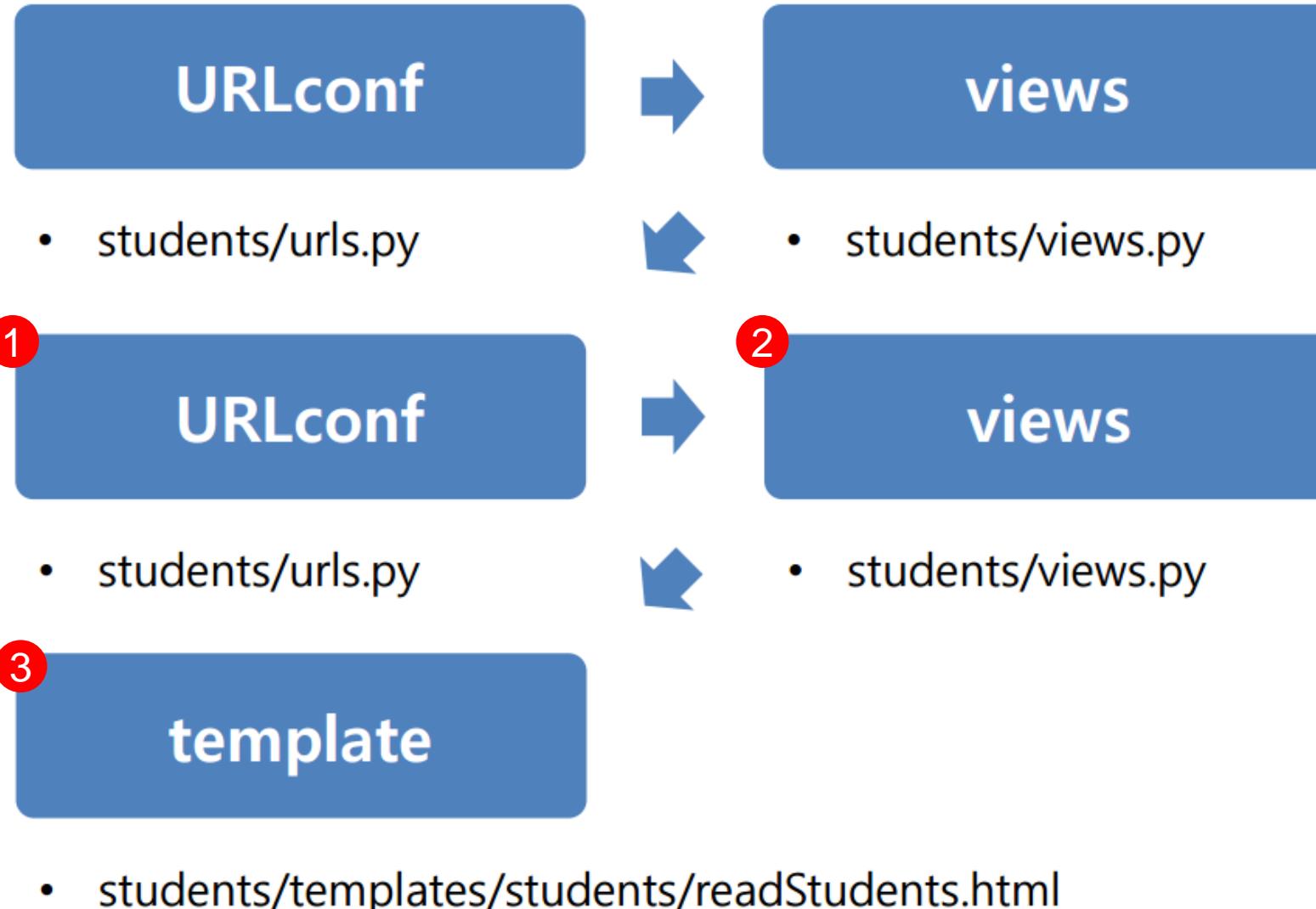
```
/*  
이곳에 작성...  
*/
```

[Django Template]

```
{% comment %}  
이곳에 작성...  
{% endcomment %}
```

6 학생리스트 페이지 만들기

학생 등록 확인 및 학생 리스트 페이지



The screenshot shows a browser window displaying the URL `127.0.0.1:8000/students/all/`. The page title is "Read Students". It lists two students:

- 1. hong_gil_dong 수정 삭제
- 2. hong_gil_ja 수정 삭제

A link labeled "신규학생등록" is visible. Below the list, there is an "Action" dropdown menu and a "Go" button. The footer indicates "0 of 2 selected". The student names "hong_gil_ja" and "hong_gil_dong" are also listed under their respective checkboxes.

6 학생리스트 페이지 – urls.py

1

app 폴더 > urls.py 추가

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

app_name = 'students'
urlpatterns = [
    path('reg/',views.regStudent,name='reg'),
    path('all/',views.reaStudentAll,name='stuAll'),  

    path('regCon/',views.regConStudent,name='regCon'),
    path('modCon/',views.modConStudent,name='modCon'),
    path('<str:name>/det/',views.detStudent,name='stuDet'),
    path('<str:name>/mod/',views.reaStudentOne,name='stuMod'),
    path('modCon/', views.modConStudent, name='modCon'),
    path('<str:name>/stuDel/', views.delConStudent, name='stuDel'),  

]
```

6 학생 리스트 페이지 – views.py : students:stuAll

2

```
## 학생리스트페이지
def reaStudentAll(request):
    qs = Student.objects.all()
    context = {'student_list':qs}
    return render(request,'students/readStudents.html',context)
```

views.py

```
qs = Student.objects.all()      ##모든 학생을 가져올때

b_count = qs.count()          ## 전체 개수

qs = Student.objects.filter(s_name__contains='h')
## 이름이 h로 시작되는 학생을 가져올때

qs = Student.objects.get(s_name = name)
## 변수를 받아 학생이름으로 가져올때

qs = FreeBoard.objects.all().order_by('-b_no')
## 역순정렬
```

_lt	~보다 작다
_lte	~보다 작거나 같다
_gt	~보다 크다
_gte	~보다 크거나 같다
_isnull	~ null인 자료 검색
_contains	특정 문자열을 포함하는 자료 검색
_startswith	특정 문자열로 시작하는 자료 검색
_endswith	특정 문자열로 끝나는 자료 검색
order_by(' ')	오름차순
order_by(' - ')	내림차순

```
## 학생리스트페이지
def reaStudentAll(request):
    qs = Student.objects.filter(b_userId__contains='h')

    b_count = qs.count()  ## 게시글 총 개수
    context ={'boardList':qs,'b_count':b_count,'master':'admin'}
    return render(request,'boardList.html',context)
```

6 [참조]게시판리스트

```
## 리스트페이지  
def boardList(request):  
    #qs = FreeBoard.objects.all()  
  
    qs = FreeBoard.objects.filter(b_userId__contains='h').order_by('-b_no')  
    b_count = qs.count()  
    context ={'boardList':qs,'b_count':b_count,'master':'admin'}  
  
    return render(request,'boardList.html',context)
```

게시판리스트

여러 개 데이터를 html로 넘길 때
딕셔너리 추가

학생 리스트 페이지 – readStudents.html : students:stuAll

3

```
<script>
    function studentDel(name){
        alert(name);
        if(confirm("삭제하시겠습니까?")){
            location.href=.." + name + "/stuDel/";
        }
    }
</script>
```

삭제시 javascript 확인

Read Students

```
{% if student_list %}
<ol>
    {% for s in student_list %}
        <li>
```

데이터가 있는지 확인

```
            <a href="#">{{s.s\_name}}
            <a href="#">수정
            <a id="btnCancel" onclick="studentDel('{{s.s_name}}')">삭제
            <a href="#">삭제
        </li>
    {% endfor %}
```

데이터를 for문으로 출력

{{ s.s_name }}

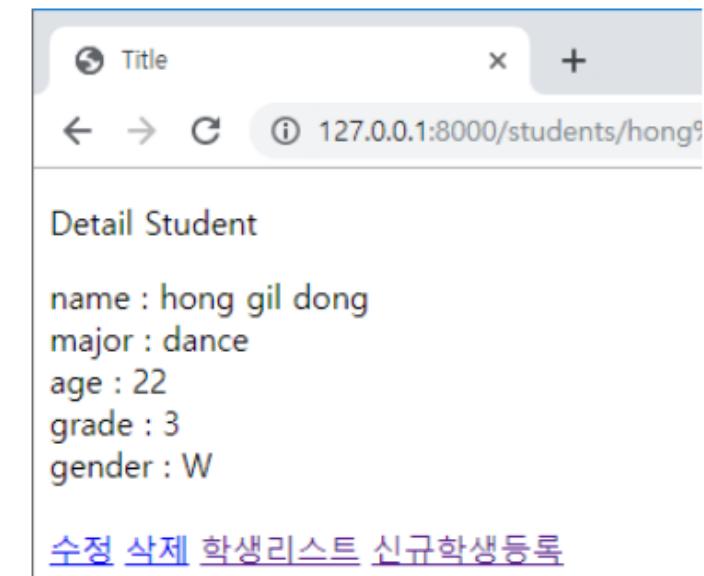
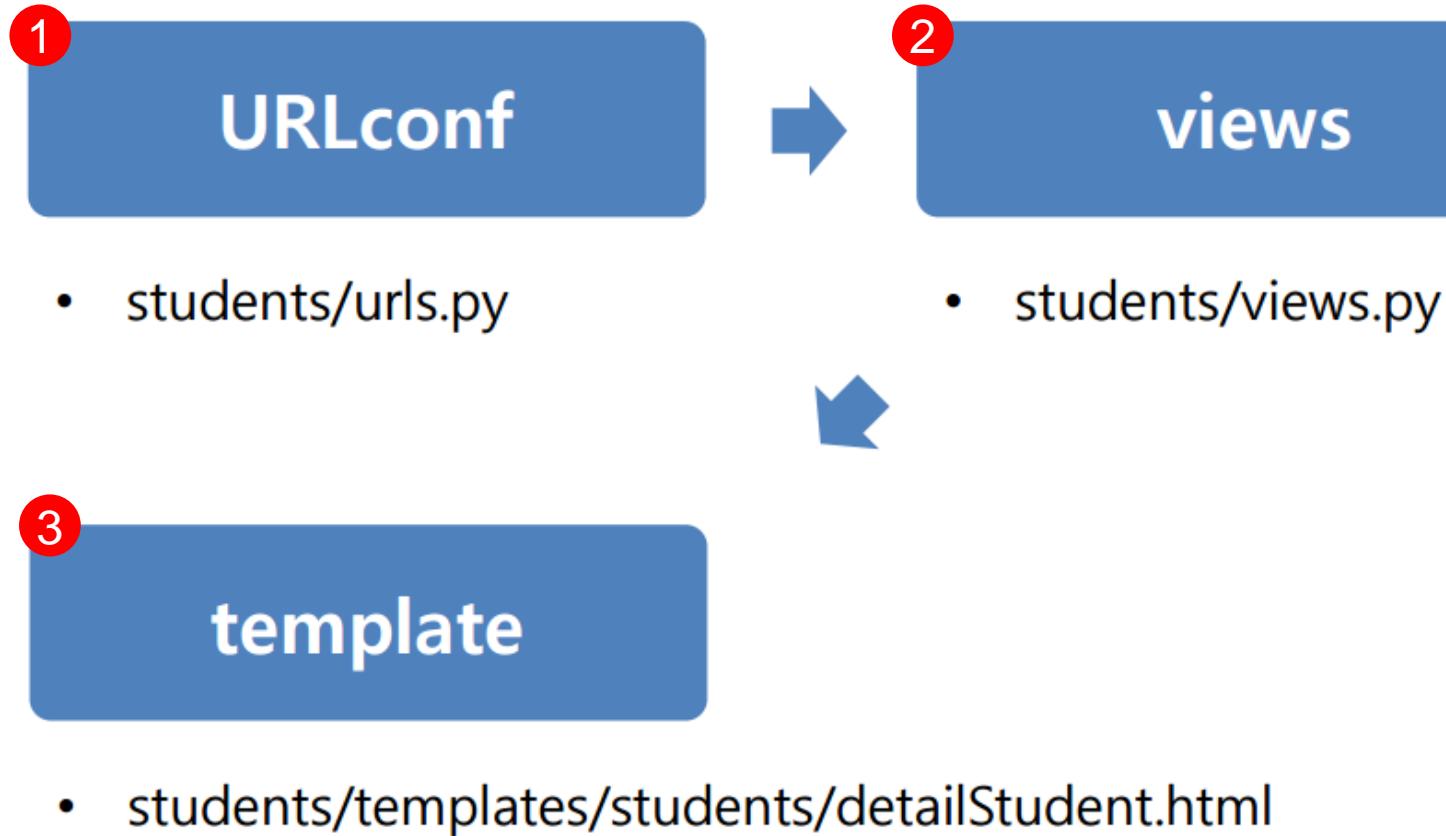
```
</ol>
{% else %}
    <p>가져올 데이터가 없습니다.</p>
{% endif %}
<a href="#">신규학생 등록
<br>
```

```
<h1>게시판 리스트</h1>
<h2>관리자 : {{ master }}, 테스트 변수 : {{ test }}</h2>
<table>
  <tr>
    <th>번호</th><th>제목</th><th>내용</th>
    <th>등록일</th><th>등록자</th><th>조회</th>
  </tr>
  {% if boardList %}
    {% for board in boardList %}
      <tr>
        <td>{{board.b_no }}</td>
        <td>{{board.b_title }}</td>
        <td>{{board.b_content }}</td>
        <td>{{board.b_date|date:'Y-m-d H:i' }}</td>
        <td>{{board.b_userId }}</td>
        <td>{{board.b_hit }}</td>
      </tr>
    {% endfor %}
  {% else %}
    <tr>
      <td cols="5">데이터가 없습니다.</td>
    </tr>
  {% endif %}
</table>
<br>
<button><a href="{% url 'board:boardWrite' %}">글쓰기</a></button>
```

dictionary 에 추가로 데이터를 보낼때
데이터 출력

7 학생 정보 보기 페이지 만들기

학생 상세 정보 페이지



7 학생 정보 보기 퍼[0|지] – urls.py

1

app 폴더 > urls.py 추가

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

app_name = 'students'
urlpatterns = [
    path('reg/',views.regStudent,name='reg'),
    path('all/',views.reaStudentAll,name='stuAll'),
    path('<str:name>/det/',views.detStudent,name='stuDet'),
    path('regCon/',views.regConStudent,name='regCon'),
    path('modCon/',views.modConStudent,name='modCon'),
    path('<str:name>/mod/',views.reaStudentOne,name='stuMod'),
    path('modCon/', views.modConStudent, name='modCon'),
    path('<str:name>/stuDel/', views.delConStudent, name='stuDel'),
]
```

html 파일에서 데이터를 전달 받음

<str : name > # 문자일때

<int : b_no > # 숫자일때

학생 정보 보기 페이지 – views.py : students:stuDet

2

```
def detStudent(request,name):
    qs = Student.objects.get(s_name = name)
    context ={'student_info':qs}
    return render(request,'students/detailStudent.html',context)
```

views.py

_lt	~보다 작다
_lte	~보다 작거나 같다
_gt	~보다 크다
_gte	~보다 크거나 같다
_isnull	~ null인 자료 검색
_contains	특정 문자열을 포함하는 자료 검색
_startswith	특정 문자열로 시작하는 자료 검색
_endwith	특정 문자열로 끝나는 자료 검색
order_by(' ')	오름차순
order_by('- ')	내림차순

학생 정보 보기 페이지 – readStudents.html : students:stuDet

3

```
<h2>Detail Students</h2>  
{% if student_info %}  
    name : {{student_info.s_name}}<br>  
    major : {{student_info.s_major}}<br>  
    age : {{student_info.s_age}}<br>  
    grade : {{student_info.s_grade}}<br>  
    gender : {{student_info.s_gender}}<br>  
{% else %}  
    <p>가져올 데이터가 없습니다.</p>  
{% endif %}  
<br>  
<a href="{% url 'students:stuMod' student_info.s_name %}">수정</a>  
<a href="{% url 'students:stuDel' student_info.s_name %}">삭제</a>  
<a href="{% url 'students:stuAll' %}">학생리스트</a>  
<a href="{% url 'students:reg' %}">신규학생 등록</a>
```

* **student_info** 정보가 있는지 확인

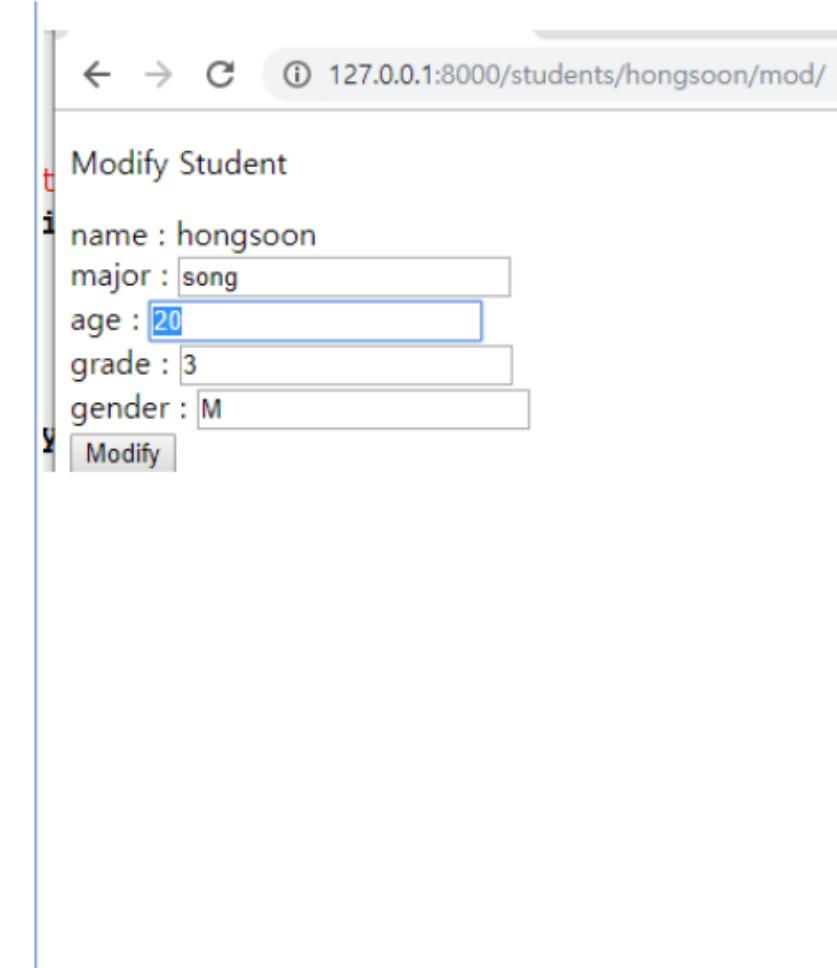
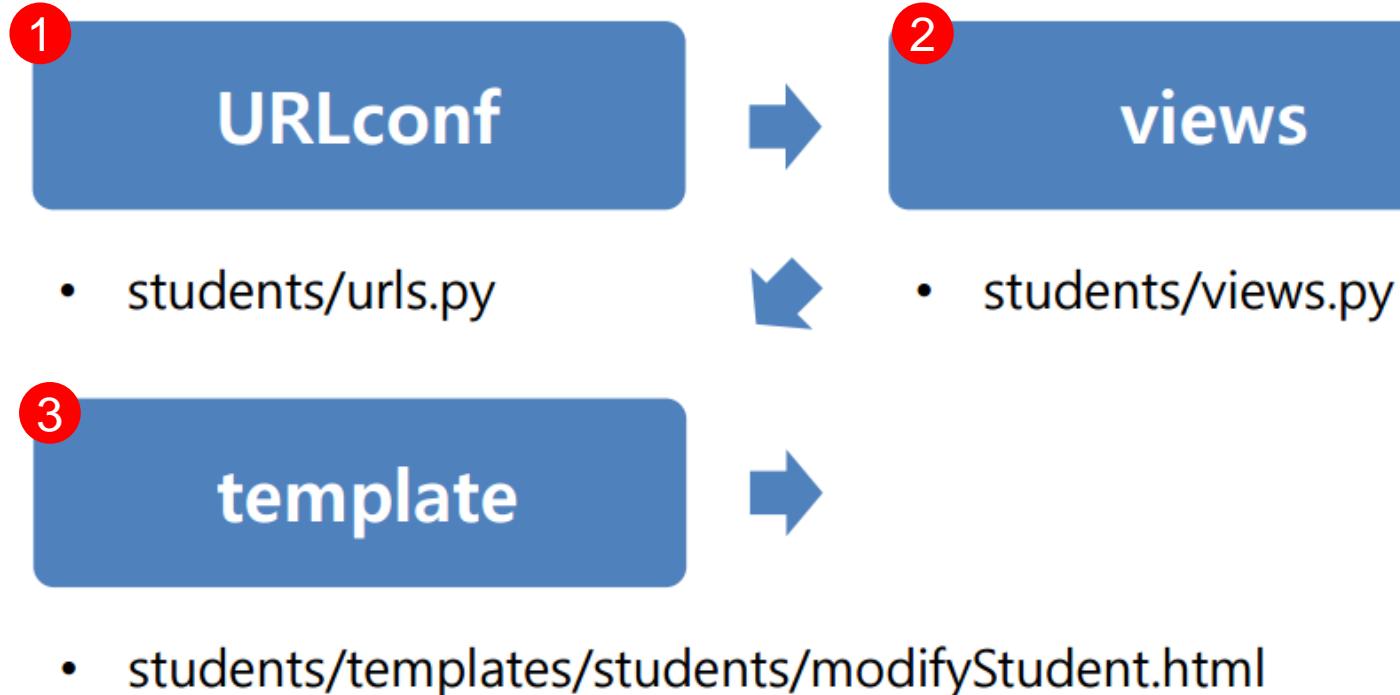
views.py에서 데이터를 전달받아
context ={'student_info':qs}
student_info 파일의 정보 출력

데이터를 **for**문으로 출력

{{ s.s_name }}

학생 정보 수정 폼 만들기 : students:stuMod

학생 정보 수정 폼



학생 정보 수정페이지 – urls.py : students:stuMod

1

app 폴더 > urls.py 추가

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

app_name = 'students'
urlpatterns = [
    path('reg/',views.regStudent,name='reg'),
    path('regCon/',views.regConStudent,name='regCon'),
    path('all/',views.reaStudentAll,name='stuAll'),
    path('<str:name>/det/',views.detStudent,name='stuDet'),
    path('<str:name>/mod/',views.reaStudentOne,name='stuMod'),
    path('modCon/',views.modConStudent,name='modCon'),
    path('<str:name>/stuDel/', views.delConStudent, name='stuDel'),
]
```

학생 정보 수정 페이지 – views.py : students:stuMod

2

```
def reaStudentOne(request,name):
    qs = Student.objects.get(s_name = name)
    context = {'student_info':qs}
    return render(request,'students/modifyStudent.html',context)
```

views.py

_lt	~보다 작다
_lte	~보다 작거나 같다
_gt	~보다 크다
_gte	~보다 크거나 같다
_isnull	~ null인 자료 검색
_contains	특정 문자열을 포함하는 자료 검색
_startswith	특정 문자열로 시작하는 자료 검색
_endwith	특정 문자열로 끝나는 자료 검색
order_by(' ')	오름차순
order_by('- ')	내림차순

학생 정보 수정 페이지 – modifyStudents.html : students:stuMod

3

```
<h2>Modify Student</h2>
```

```
{% if student_info %}  
    <form action="{% url 'students:modCon' %}" method="post">  
        {% csrf_token %}  
        name:{{student_info.s_name}}<br>  
        <input type="hidden" name="name" value={{student_info.s_name }}>  
        major:<input type="text" name="major"  
        value={{student_info.s_major }}><br>  
        age:<input type="text" name="age"  
        value={{student_info.s_age }}><br>  
        grade:<input type="text" name="grade"  
        value={{student_info.s_grade }}><br>  
        gender:<input type="text" name="gender"  
        value={{student_info.s_gender }}><br>  
        <input type="submit" value="수정"><br>  
    </form>  
    {% else %}  
        <p>데이터가 없습니다.</p>  
    {% endif %}
```

* **student_info** 정보가 있는지 확인

views.py에서 데이터를 전달받아
context ={'student_info':qs}
student_info 파일의 정보 출력

value="{{student_info.s_major }}"
- " " : 넣어야 글자 끊김이 없음.

학생 정보 수정 완료페이지 만들기 : students:modCon

학생 정보 수정 확인 페이지

1

URLconf

- students/urls.py

2

views

- students/views.py

← → C ⓘ 127.0.0.1:8000/student

Modify Student

name : hongsoon
major : song
age : 25
grade : 3
gender : M

Modify

학생 정보 수정 완료페이지 – urls.py students:modCon

1

app 폴더 > urls.py 추가

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

app_name = 'students'
urlpatterns = [
    path('reg/',views.regStudent,name='reg'),
    path('regCon/',views.regConStudent,name='regCon'),
    path('all/',views.reaStudentAll,name='stuAll'),
    path('<str:name>/det/',views.detStudent,name='stuDet'),
    path('<str:name>/mod/',views.reaStudentOne,name='stuMod'),  

    path('modCon/',views.modConStudent,name='modCon'),  

    path('<str:name>/stuDel/', views.delConStudent, name='stuDel'),
]
```

학생 정보 수정 완료페이지 – views.py : students:modCon

2

```
def modConStudent(request):
    name = request.POST['name']
    major = request.POST['major']
    age = request.POST['age']
    grade = request.POST['grade']
    gender = request.POST['gender']

    s_qs = Student.objects.get(s_name=name)

    s_qs.s_name = name
    s_qs.s_major = major
    s_qs.s_age = age
    s_qs.s_grade = grade
    s_qs.s_gender = gender
    s_qs.save()

    return HttpResponseRedirect(reverse('students:stuAll'))
```

views.py

* form에서 정보를 넘김

* 해당 데이터 이름으로(**name**) 검색

* 해당데이터에 데이터값 수정

7 학생 정보 삭제 페이지 만들기

학생 정보 삭제 페이지

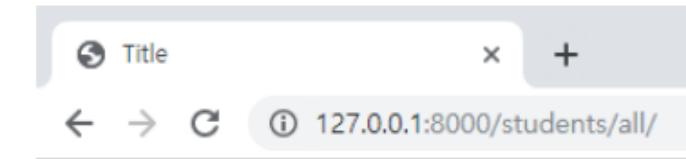
1

URLconf

2

views

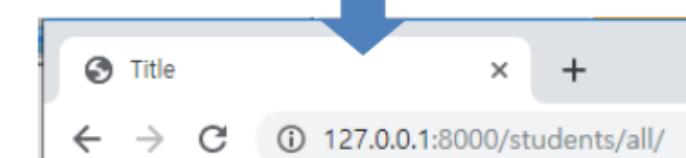
- students/urls.py
- students/views.py



Read Students

1. [honggildong](#) 수정 [삭제](#)
2. [honggilsoon](#) 수정 [삭제](#)

[신규학생등록](#)



Read Students

1. [honggilsoon](#) 수정 [삭제](#)

[신규학생등록](#)

학생 정보 삭제 페이지 – urls.py students:stuDel

1

app 폴더 > urls.py 추가

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

app_name = 'students'
urlpatterns = [
    path('reg/',views.regStudent,name='reg'),
    path('regCon/',views.regConStudent,name='regCon'),
    path('all/',views.reaStudentAll,name='stuAll'),
    path('<str:name>/det/',views.detStudent,name='stuDet'),
    path('<str:name>/mod/',views.reaStudentOne,name='stuMod'),
    path('modCon/',views.modConStudent,name='modCon'),  
    path('<str:name>/stuDel/', views.delConStudent, name='stuDel'),  
]
```

학생 정보 삭제 페이지 – views.py : students:stuDel

2

```
def delConStudent(request,name):
    qs = Student.objects.get(s_name=name)
    qs.delete()

    return HttpResponseRedirect(reverse('students:stuAll'))
```

views.py

* name으로 데이터 검색후
1개의 데이터를 삭제

11 Login 설정

```
## 로그인 설정

def login(request):
    if request.method == 'GET':
        return render(request, 'login.html')

    elif request.method == 'POST':
        user_id = request.POST['id']
        user_pw = request.POST['pw']
        res_data = {}
        if not (user_id and user_pw):
            res_data['error'] = '모든 값을 입력해 주세요.'
        else:
            member = Member.objects.get(user_id=user_id)
            if member:
                user_id = member.user_id
                request.session['session_id'] = user_id
                # 세션에서 user를 key로하고 값을 할당 받은 user_id로 설정
                # session_id = cleaned_data.get('session_id')
                return redirect('/')
                # '/' 홈으로 이동
            else:
                res_data['error'] = '잘못된 아이디 또는 비밀번호입니다.'
        return render(request, 'login.html', res_data)
```

```
## 로그아웃 설정

def logout(request):
    if request.session.get('session_id'):
        request.session.clear()    # 모든 세션 삭제
        del request.session['session_id'] ## 일부 세션만 삭제
    return redirect('/')
```

```
## html

<h1>로그인</h1>
{%
    if request.session.session_id %}
    <h1>로그인 상태 : {{ request.session.session_id }} </h1>
{%
    else %}
    <h1>로그인 하셔야 합니다. </h1>
{%
    endif %}
```

11 ajax 댓글리스트

```
<!-- 댓글-->
<li class="btn"><a href="#" class="replyBtn"
onclick="commentWrite()">등록</a>
</li>
```

The screenshot shows a comment list page with the following details:

- Header:** 총 19 개의 댓글이 달려있습니다. (Total 19 comments)
비밀번호 등록
- Comment 1:** aaa [2021-04-13 11:14:02]
대박! 이벤트 참여해요~ 담첨되었으면 좋겠어요~
수정 삭제
- Comment 2:** aaa [2021-04-13 11:14:02]
대박! 이벤트 참여해요~ 담첨되었으면 좋겠어요~
저장 취소

메소드	설명
.append()	선택된 요소의 마지막에 새로운 요소나 콘텐츠를 추가한다.
.prepend()	선택된 요소의 첫번째에 새로운 요소나 콘텐츠를 추가한다.
.appendTo()	선택된 요소를 해당 요소의 마지막에 추가한다.
.prependTo()	선택된 요소를 해당 요소의 첫번째에 추가한다.

THANK YOU