

파이썬 웹 프로그래밍

01. 웹 프로그래밍의 이해

02. 웹 표준 라이브러리

03. Django 웹 프레임워크

04. Django 활용

1. 웹 프로그래밍의 이해

1.1 인터넷의 이해

1) 인터넷 != WWW(World Wide Web)
인터넷 기반의 대표 서비스 중 하나

이름	프로토콜	포트	기능
WWW	HTTP	80	웹서비스
Email	SMTP/POP3/IMAP	25/110/114	이메일 서비스
FTP	FTP	21	파일 전송 서비스
DNS	DNS	23	네임서비스
NEWS	NNTP	119	인터넷 뉴스 서비스

2) 인터넷 (Internet)
TCP/IP 기반의 네트워크가 전세계적으로 확대되어 하나로 연결된 네트워크들의
네트워크 (네트워크의 결합체)

1. 웹 프로그래밍의 이해

1.1 인터넷의 이해

3) TCP / IP

- 하드웨어, 운영체제, 접속 매체와 관계없이 동작할 수 있는 개방형 구조
- OSI 7 계층에서 4계층으로 단순화.

OSI 7계층	TCP/IP 4계층	
응용 계층	응용 계층	• 네트워크를 사용하는 WWW, FTP, 텔넷, SMTP 등의 응용 프로그램으로 구성.
표현 계층		
세션 계층	전송 계층	• 도착지까지 데이터를 전송 • 각각의 시스템을 연결 • TCP 프로토콜을 이용하여 데이터를 전송
전송 계층		
네트워크 계층	인터넷 계층	• 데이터를 정의 및 경로 지정 • 정확한 라우팅을 위해 IP 프로토콜을 사용 • IP 주소가 위치하는 계층
데이터 링크 계층		
물리 계층	물리 계층	• 물리적 계층 즉 이더넷 카드와 같은 하드웨어

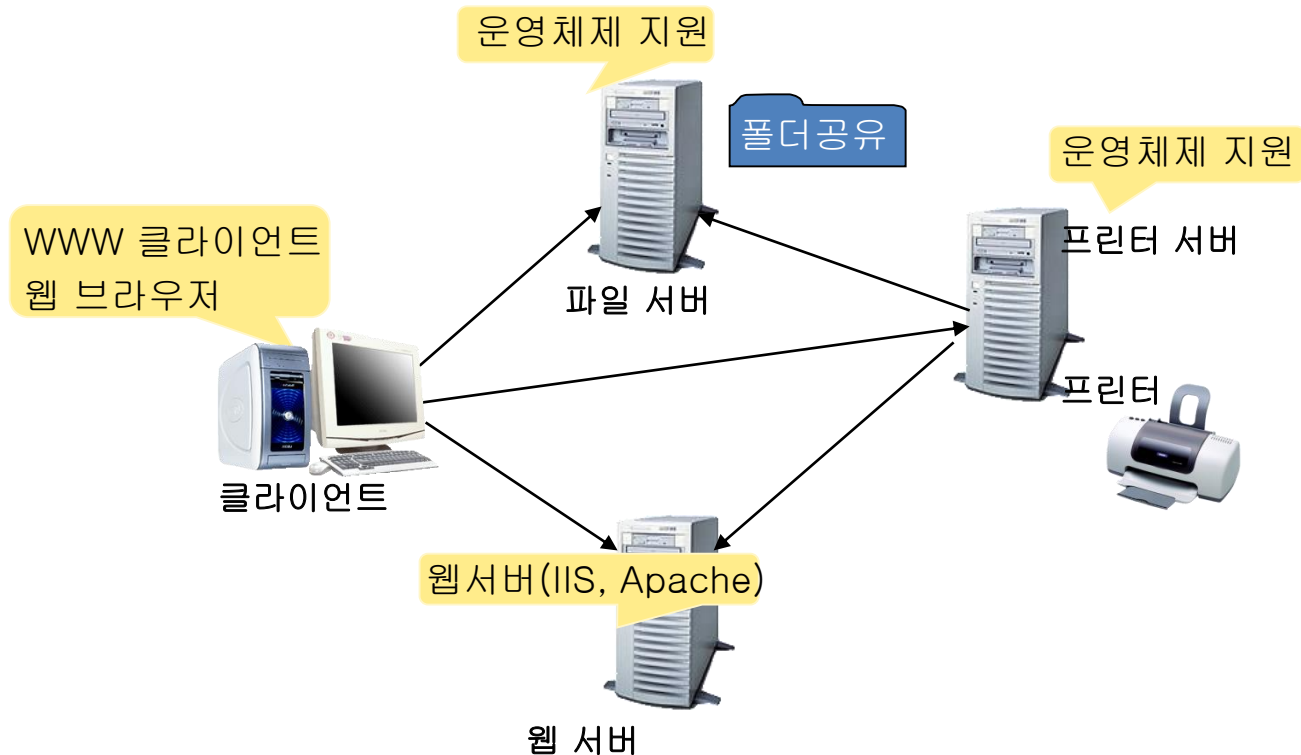
1. 웹 프로그래밍의 이해

1.1 인터넷의 이해

4) 서비스 (클라이언트 / 서버 통신)

- 서버 : 네트워크에서 서비스를 제공하는 컴퓨터
- 클라이언트 : 네트워크에서 서비스를 제공받는 컴퓨터

하드웨어적 구분은 사실상 없음



1. 웹 프로그래밍의 이해

1.2 HTTP(Hyper Text Transfer Protocol)

- 1) WWW 서비스를 위한 TCP/IP 응용계층 프로토콜 중 하나
- 2) 웹 서버 와 클라이언트는 HTTP를 이용한 통신
- 3) 무 상태 연결(Stateless Connection)

1. 웹 프로그래밍의 이해

1.2 HTTP(Hyper Text Transfer Protocol)



1. 웹 프로그래밍의 이해

1.3 URL(Uniform Resource Locator)

- 1) 인터넷 상의 자원의 위치
- 2) 특정 웹 서버의 특정파일에 접근하기 위한 경로 혹은 주소

접근 프로토콜://IP 주소 또는 도메인 이름/문서의 경로/문서이름

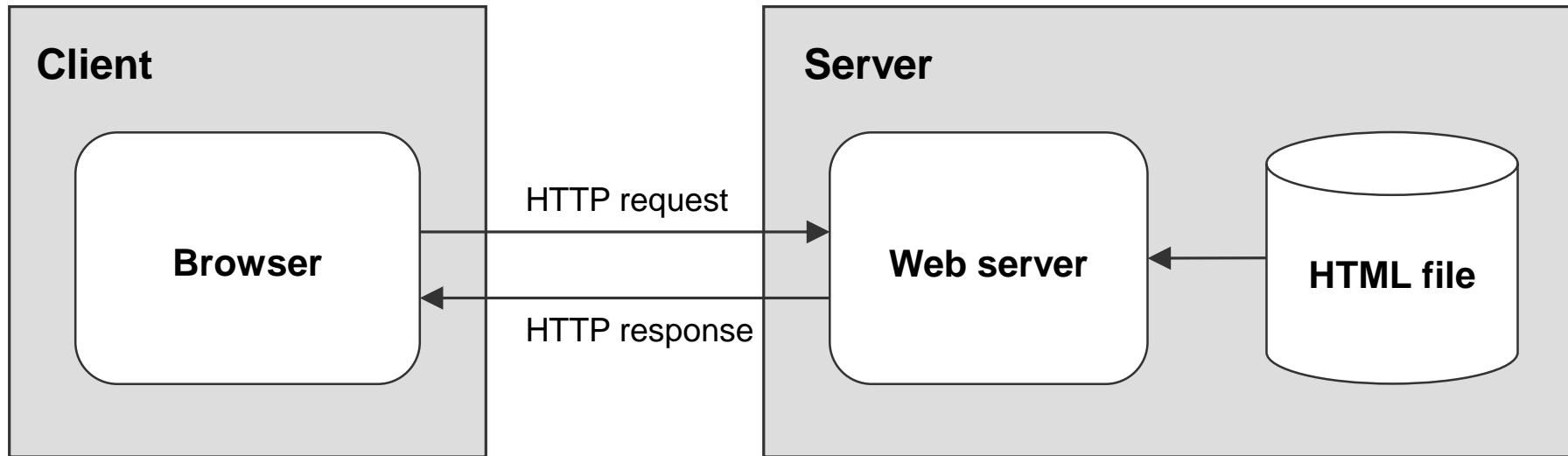


<http://www.sunnyvale.co.kr/docs/index.html>

1. 웹 프로그래밍의 이해

1.4 웹 프로그래밍

1) 정적인(static) 웹 페이지 (~~웹프로그래밍~~, 퍼블리싱)



웹 페이지는 HTML 이라는 표준 마크업랭귀지로 작성.

1. 웹 프로그래밍의 이해

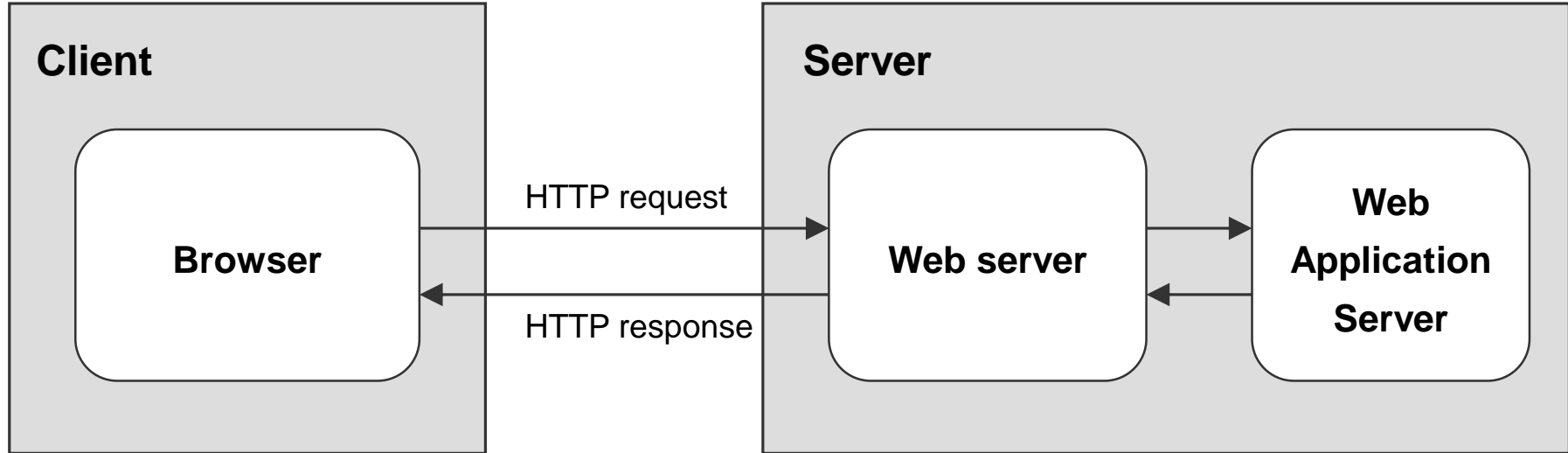
1.4 웹 프로그래밍

2) 정적인(static) 웹 페이지 처리 방식

- HTML(Hypertext Markup Language)은 브라우저가 웹 페이지로 변환하는 언어이다.
- 정적인 웹 페이지는 파일 형태로 저장되어 있으면서 사용자의 입력에 따라 변하지 않는 HTML 문서이다.
- HTTP(Hypertext Transfer Protocol)는 웹 브라우저와 웹 서버가 통신하는 프로토콜이다.
- 웹 브라우저는 HTTP 요청(HTTP request) 메시지를 서버에 전달함으로써 웹 서버의 페이지를 요청한다.
- 웹 서버는 HTTP 응답(HTTP response) 메시지를 전달함으로써 HTTP 요청에 응답한다. 정적인 웹 페이지에서는 HTTP 응답이 HTML 문서를 포함한다.

1. 웹 프로그래밍의 이해

3) 동적인(Dynamic) 웹 페이지를 만들어 내는 모든 기술



1) 자바 – Tomcat, Nginx

2) 루비 – Unicorn

3) 파이썬 – uWSGI

1. 웹 프로그래밍의 이해

4) 동적인(Dynamic) 웹 페이지 처리 방식

- 동적인 웹 페이지는 웹 어플리케이션에 의해 생성되는 HTML 문서이다. 웹 브라우저가 웹 어플리케이션에 전달한 파라미터 값에 따라 웹 페이지가 변한다.
- 웹 서버가 동적인 웹 페이지에 대한 요청을 받으면 서버는 웹 어플리케이션으로 요청을 넘긴다. 그러면 어플리케이션이 HTML 문서를 생성하여 웹 서버로 결과를 전달한다.
- 웹 서버는 HTML 문서를 HTTP 응답(HTTP response)으로 감싼 후 브라우저로 결과를 전달한다.
- 전달 받은 HTML 문서가 정적인 HTML 파일에서 왔는지 아니면 웹 어플리케이션에 의해 동적으로 생성된 문서인지 브라우저는 알지 못한다. 어느 쪽이든 브라우저는 전달받은 HTML 문서를 화면에 표시한다.

파이썬 웹 프로그래밍

01. 웹 프로그래밍의 이해

02. 웹 표준 라이브러리

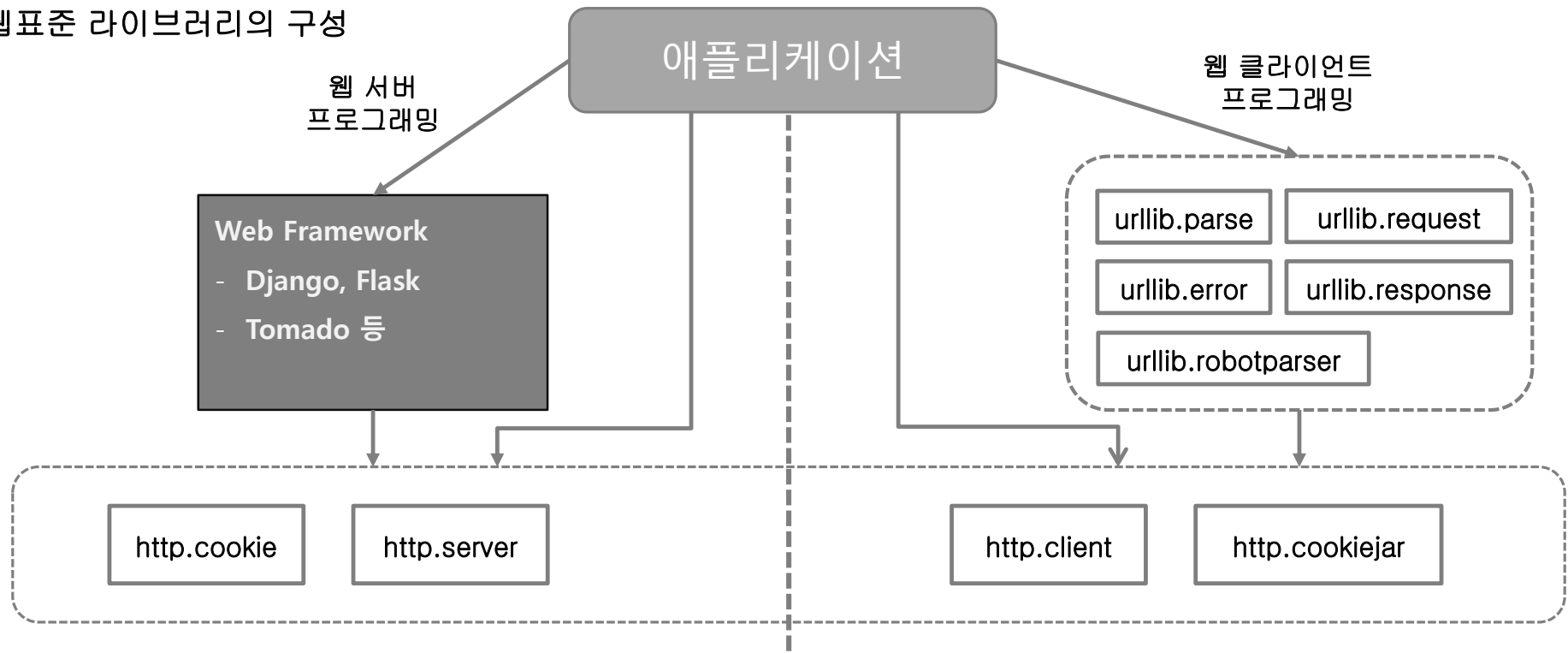
03. Django 웹 프레임워크

04. Django 활용

2. 웹 표준 라이브러리

2.1 파이썬 웹 표준 라이브러리 구성

- 1) 파이썬 설치 시, 기본적으로 함께 설치되는 표준 라이브러리
- 2) 웹 클라이언트 프로그래밍 또는 웹 서버 프로그래밍에 따라 사용 모듈이 달라진다.
- 3) 2.x 와 3.x 에 따라 패키지명, 모듈명이 재구성되어 있는 차이가 있고 함수와 클래스등의 내용 자체는 거의 동일
- 4) 웹표준 라이브러리의 구성



2. 웹 표준 라이브러리

2.2 웹 클라이언트 라이브러리

- 1) 웹 서버에 요청을 보내는 모든 애플리케이션은 웹 클라이언트라 할 수 있다.
- 2) 웹브라우저, Open API를 사용하는 프로그램들
- 3) 웹 클라이언트와 웹 서버는 HTTP/HTTPS 프로토콜을 사용한다.

2.2.1 urllib.parse 모듈

- 1) URL 분해, 조립, 변경 등을 처리하는 함수를 제공한다.

```
from urllib.parse import urlparse

result = urlparse("http://www.python.org:80/guido/python.html;philosophy?overall=3#here")
print(result)
```

- 2) 그 외에 urlsplit(), urljoin(), parse_qs() 등의 함수를 제공한다.

2. 웹 표준 라이브러리

2.2.2 urllib.request 모듈의 urlopen() 함수

1) URL에서 데이터를 가져오는 기본 기능을 제공한다.

2) GET 방식으로 웹 서버에 요청 보내기

```
from urllib.request import urlopen

f = urlopen('http://www.naver.com')
print(f.read())
```

3) POST 방식으로 웹 서버에 요청 보내기

```
from urllib.request import urlopen, Request
from urllib.parse import urlencode

data = urlencode({'a': 'Whatdoesthefoxsay', 'b': 'lingdingdingdingdiding' })
data = data.encode('UTF-8')

request = Request('http://www.example.com', data)
request.add_header('Content-Type', 'text/html')

f = urlopen(request)
```

2. 웹 표준 라이브러리

4) Request 객체를 사용한 요청에 헤더 지정하기

```
from urllib.request import Request, urlopen

url = "http://www.naver.com"

request = Request(url)
request.add_header('Content-Type', 'text/html')

response = urlopen(request)
print(response.read())
```

[과제]

parse_image.py 작성하기

2. 웹 표준 라이브러리

2.2.3 http.client 모듈의 HttpConnection

- 1) HTTP 프로토콜의 저수준의 요청을 보낼 수 있다.
- 2) urllib.request 요청과 동일한 요청을 보낼 수 있다.
- 3) GET 방식 요청

```
from http.client import HTTPConnection

conn = HTTPConnection('www.example.com')

conn.request('GET', '/')
r = conn.getresponse()
print(r.status, r.reason)

data = r.read()

conn.request('GET', '/hello.html')
r = conn.getresponse()
print(r.status, r.reason)
```

2. 웹 표준 라이브러리

3) HEAD 메소드 요청

```
from http.client import HTTPConnection

conn = HTTPConnection('www.example.com')

conn.request('GET', '/')
r = conn.getresponse()
print(r.status, r.reason)

data = r.read()

conn.request('GET', '/hello.html')
r = conn.getresponse()
print(r.status, r.reason)
```

4) PUT, POST 방식의 요청도 가능하다.

2. 웹 표준 라이브러리

[과제]

http.client 모듈의 `HttpConnection`를 사용하여 `parse_image2.py` 작성하기

2. 웹 표준 라이브러리

2.3 웹 서버 라이브러리

1) 파이썬 웹 서버 라이브러리를 사용해 작성하긴 보다는 웹 프레임워크를 사용하여 개발하는 경우가 많다

2) 간단한 웹 서버

```
from http.server import BaseHTTPRequestHandler, HTTPServer

port = 9999

class MyHTTPRequestHandler(BaseHTTPRequestHandler):
    def do_GET(self):
        self.send_response(200)
        self.send_header('Content-type', 'text/html; charset=utf-8')
        self.end_headers()
        self.wfile.write("<h1>안녕하세요</h1>".encode('utf-8'))

httpd = HTTPServer(("", port), MyHTTPRequestHandler)
print('Server running on port', port)
httpd.serve_forever()
```

2. 웹 표준 라이브러리

2.3.1 BaseHttpServer 모듈

- 1) 기반 클래스로 HTTP 프로토콜 처리가 가능하다.
- 2) BaseHttpServer 모듈에 정의되어 있는 BaseHttpRequestHandler 클래스를 상속받아 핸들러를 정의한다.
- 3) 기반 클래스 를 상속받으면 따로 HTTP 프로토콜 관련된 로직을 코딩하지 않아도 된다.

[과제]

matplotlib와 연동되는 myhttpserver2 작성해 보기

2. 웹 표준 라이브러리

2.3.2 SimpleHttpServer 모듈

- 1) 간단한 핸들러가 미리 구현되어 있기 때문에 필요할 때 즉시 웹서버를 구동할 수 있다.
- 2) 이 모듈에는 SimpleHttpRequestHandler가 정의되어 있다.
- 3) do_GET() 및 do_HEAD() 메소드가 정의되어 있어 GET과 HEAD 방식의 처리만 가능하다.

파이썬 웹 프로그래밍

01. 웹 프로그래밍의 이해

02. 웹 표준 라이브러리

03. Django 웹 프레임워크

04. Django 활용

4. Django 웹 프레임워크

4.3 설치

- 1) 다음 명령을 입력하여 간단하게 장고를 설치할 수 있다

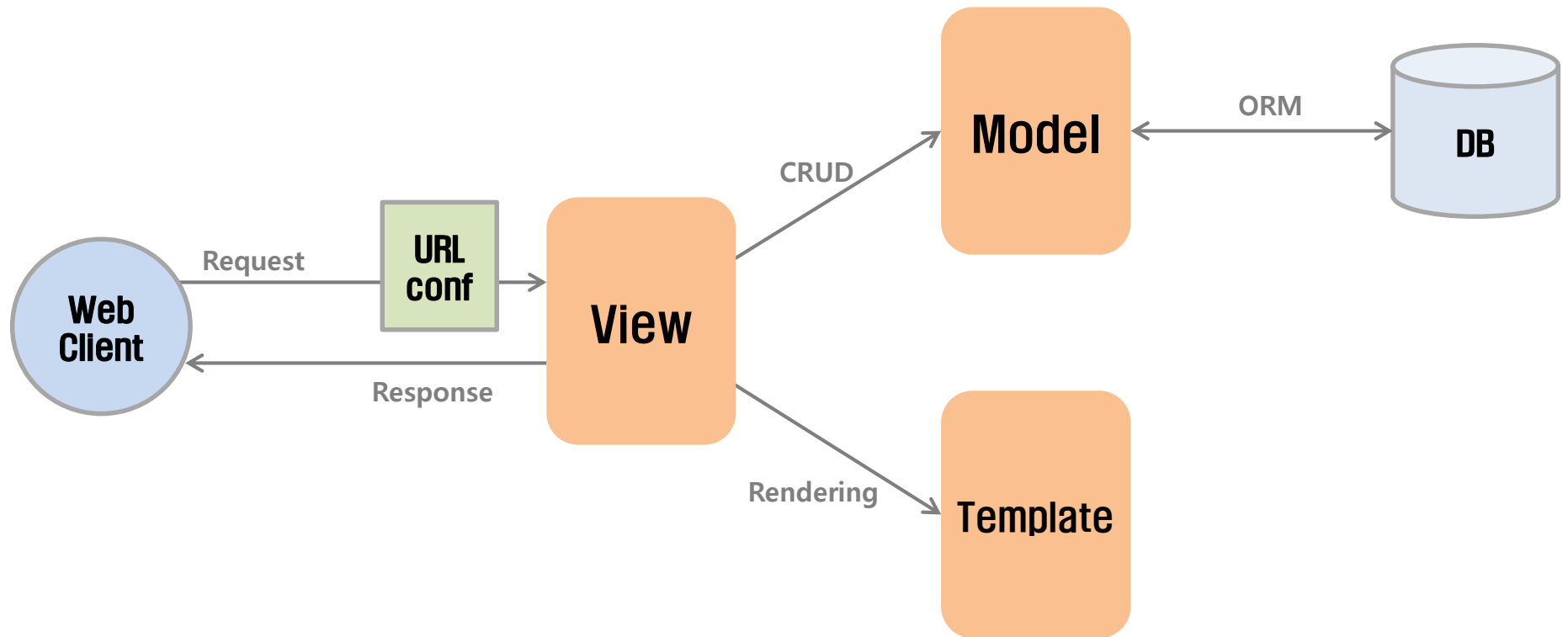
```
# pip install Django
Collecting Django
  Using cached
https://files.pythonhosted.org/packages/56/0e/afdacb47503b805f3ed213fe732bff05254c8befaa034bbada580be8a0ac/Django-2.0.6-py3-none-any.whl
Requirement already satisfied: pytz in c:\wpython\python36\lib\site-packages (from Django)
Installing collected packages: Django
Successfully installed Django-2.0.6
```

- 2) 설치 확인하기

```
# python
Python 3.6.2 (v3.6.2:5fd33b5, Jul 8 2017, 04:57:36) [MSC v.1900 64 bit (AMD64)] on win32
>>> import django
>>> print(django.get_version())
2.0.6
>>>
```


4. Django 웹 프레임워크

4.4 장고 웹 애플리케이션 디자인 패턴(MTV 패턴)



4. Django 웹 프레임워크

1) Model – 데이터베이스 CRUD

- 사용될 데이터의 정의를 담고 있는 장고의 클래스
- ORM를 사용하여 테이블과 클래스를 매핑
- 보통 `models.py` 파일에 정의한다.
- 장고는 테이블 및 컬럼을 자동으로 생성하기 위해 많은 규칙을 가지고 있다.

예)

테이블 자동 생성에 대한 자세한 설명

<https://docs.djangoproject.com/en/2.0/topics/db/models/>

- CRUD를 위한 여러 기능을 지원한다.

4. Django 웹 프레임워크

2) Template – 화면 UI 설계

- 자체 템플릿 시스템을 가지고 있다.
- 디자이너와 개발간에 업무가 완전 분리
- 템플릿에서도 파이썬 코드를 직접 사용할 수 있다. (템플릿 자체 문법에 맞게 작성)
- 템플릿 파일은 *.html 확장자를 가지며 적절한 디렉토리에 위치해 있어야 한다.

장고는 템플릿 파일을 찾을 때, settings.py에 지정되어 있는 `TEMPLATE_DIRS` 및 `INSTALLED_APPS`의 디렉토리를 검색한다.

4. Django 웹 프레임워크

3) View – 로직 설계

- 웹 요청을 받아 데이터베이스 CRUD등비지니스 로직을 수행한다.
- 그리고 그 결과 데이터를 HTML로 변환하기 위해 템플릿 처리를 한다.
- 렌더링 된 HTML을 웹 클라이언트, 브라우저로 응답하게 된다.
- 장고에서는 뷰는 함수 또는 클래스 메소드로 작성된다.
- 응답은 HTML, 404 에러, 리다이렉트 등 다양한 응답이 가능하다.
- 보통, `views.py`에 작성되지 만 다른 파일에 작성하는 것도 가능하다.

4. Django 웹 프레임워크

4) URLconf - URL 설계

- URL 지정 방식은 직관적이고 이해하기 쉽다. (Elegant URL)
- 보통, urls.py 파일에 URL과 처리함수(view)를 매핑하는 파이썬 코드를 작성한다.

4. Django 웹 프레임워크

4.5 장고 웹 애플리케이션 구조

장고 프레임워크를 사용하는 웹 애플리케이션은 **프로젝트** 와 **애플리케이션** 두 용어의 개념을 명확히 이해해야 한다.

1) 프로젝트

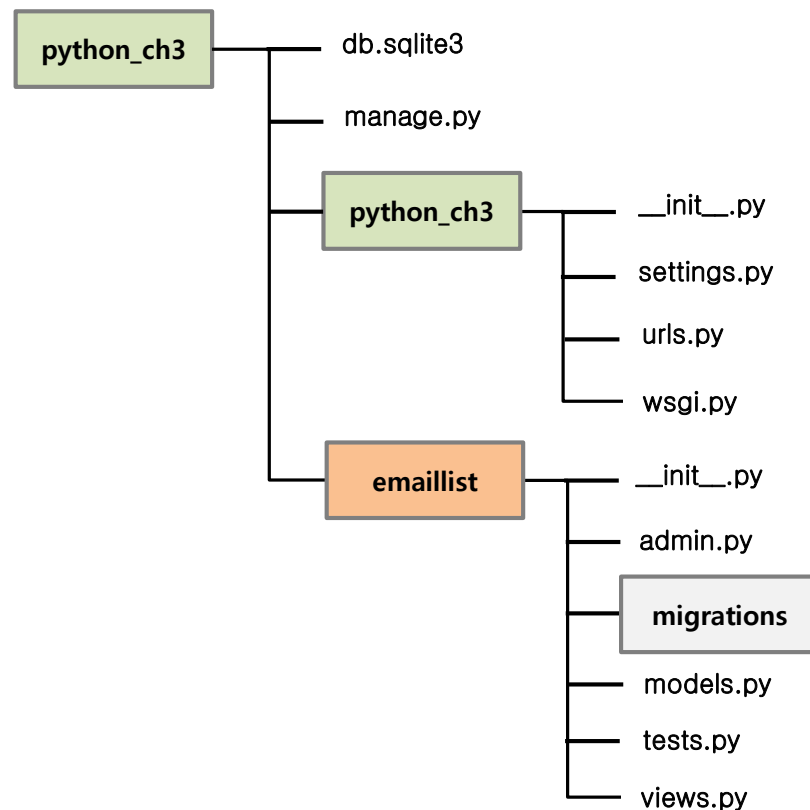
개발 대상이 되는 전체 프로그램을 의미

2) 애플리케이션

프로젝트를 여러 개의 기능으로 나누었을 때, 프로젝트 하위의 여러 서브 프로그램

3) 프로젝트 디렉토리와 애플리케이션 디렉토리를 구분해서 관리

4) 애플리케이션 단위로 프로젝트를 구성하고 프로젝트를 모아 더 큰 프로젝트를 구성하는 계층적 구조가 가능하다.



4. Django 웹 프레임워크

4.6 프로젝트 및 애플리케이션 생성 하기

1) 프로젝트 생성

```
[root@localhost ~]# django-admin startproject python_ch3  
[root@localhost ~]#
```

2) 프로젝트 디렉토리 확인하기

```
[root@localhost ~]# cd python_ch3/  
[root@localhost python_ch3]# ls -la  
합계 8  
drwxr-xr-x  3 root root  39  6월  6 13:19 .  
dr-xr-x---. 19 root root 4096  6월  6 13:19 ..  
-rwxr-xr-x  1 root root  808  6월  6 13:19 manage.py  
drwxr-xr-x  2 root root   70  6월  6 13:19 python_ch3  
[root@localhost python_ch3]#
```

4. Django 웹 프레임워크

3) emaillist 애플리케이션 생성

```
[root@localhost python_ch3]# ls -la
```

합계 4

```
drwxr-xr-x 3 root root 39 6월 6 13:19 .
```

```
drwxr-xr-x 3 root root 23 6월 6 13:55 ..
```

```
-rwxr-xr-x 1 root root 808 6월 6 13:19 manage.py
```

```
drwxr-xr-x 2 root root 108 6월 6 13:56 python_ch3
```

```
[root@localhost python_ch3]# python manage.py startapp emaillist
```

```
[root@localhost python_ch3]# ls -l
```

합계 4

```
drwxr-xr-x 3 root root 116 6월 6 13:56 emaillist
```

```
-rwxr-xr-x 1 root root 808 6월 6 13:19 manage.py
```

```
drwxr-xr-x 2 root root 108 6월 6 13:56 python_ch3
```

```
[root@localhost python_ch3]#
```


4. Django 웹 프레임워크

4) 데이터베이스 변경사항 반영

- Oarcle, MySQL, MongoDB 등 모두 사용 가능(default로 SQLite3 데이터베이스 엔진 사용)
- 장고는 모든 웹 애플리케이션이 사용자와 사용자 테이블이 필요하다고 가정하고 설계됨

```
[root@localhost python_ch3]# 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 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 sessions.0001_initial... OK
```

```
[root@localhost python_ch3]#
```

4. Django 웹 프레임워크

5) 서버실행

- 개발과정 중에 현재 상태를 확인해 볼 수 있도록 runserver 라는 간단한 테스트용 웹서버를 제공

```
[root@localhost python_ch3]# python manage.py runserver 0.0.0.0:8000
```

```
Performing system checks...
```

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

```
June 06, 2018 - 07:37:09
```

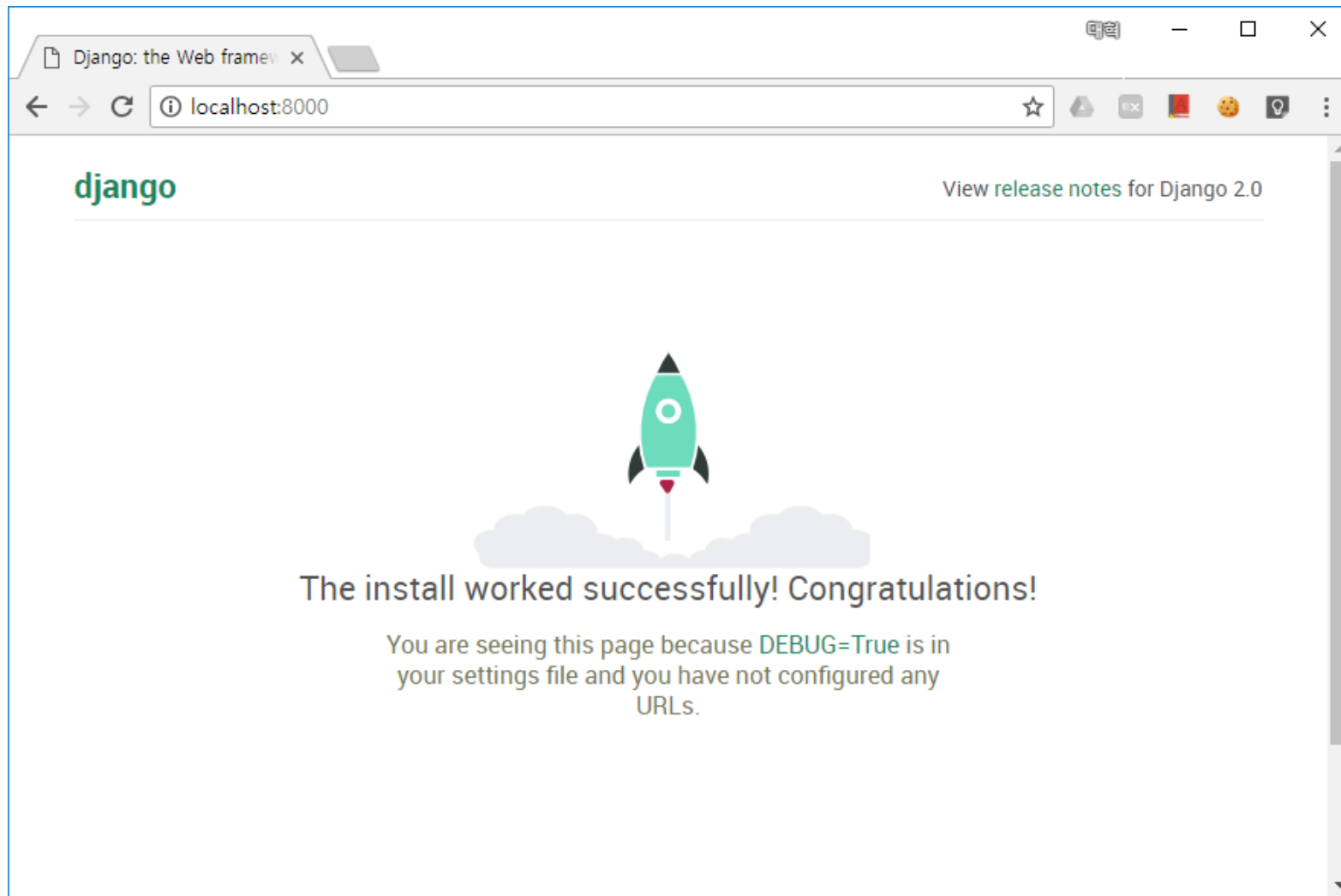
```
Django version 1.11.13, using settings 'python_ch3.settings'
```

```
Starting development server at http://0.0.0.0:8000/
```

```
Quit the server with CONTROL-C.
```

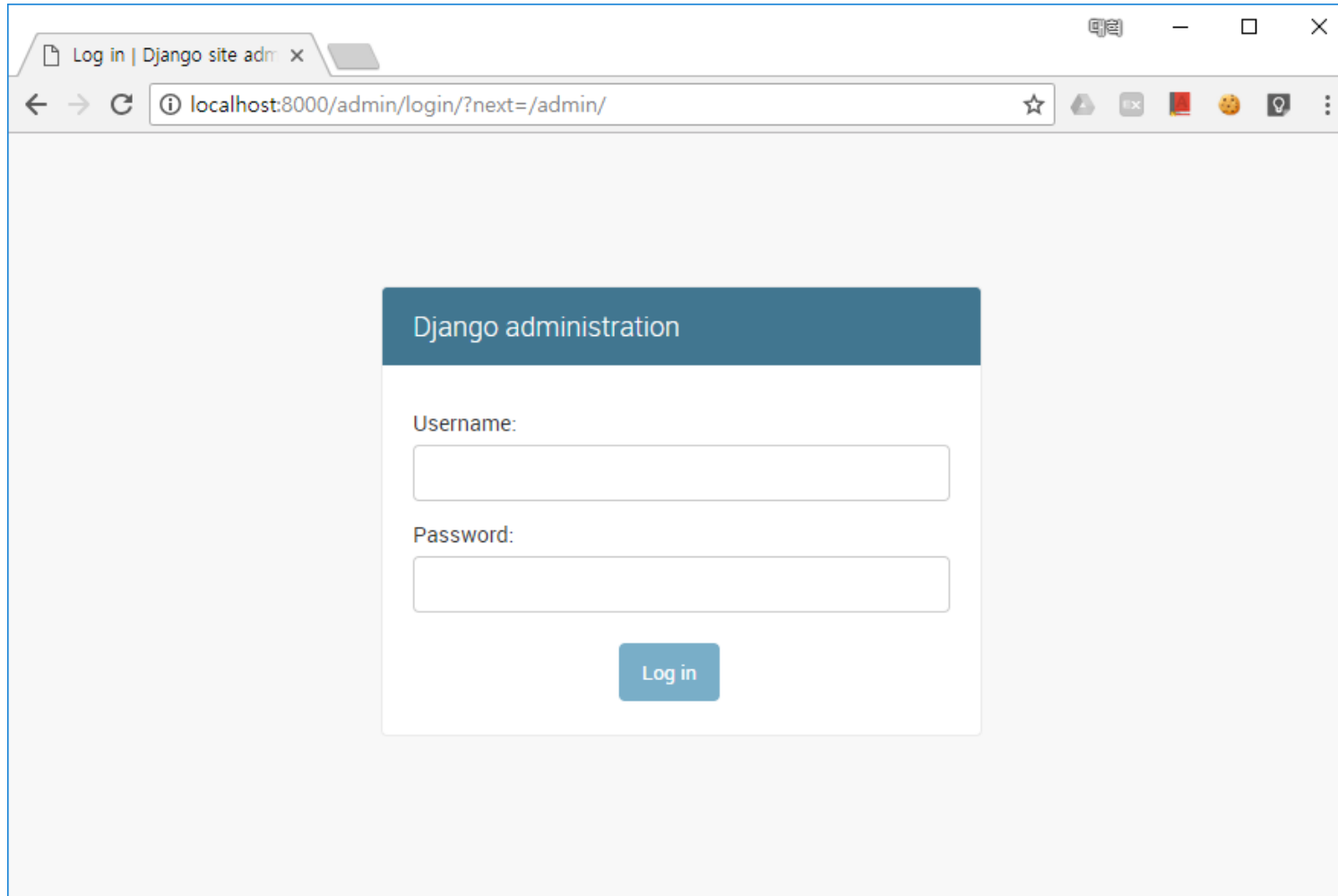
4. Django 웹 프레임워크

6) 테스트 하기



4. Django 웹 프레임워크

7) Admin 페이지 접속 확인



The screenshot shows a web browser window with the address bar displaying `localhost:8000/admin/login/?next=/admin/`. The page content is a login form for the Django administration interface. The form has a dark blue header with the text "Django administration". Below the header, there are two input fields: "Username:" and "Password:". At the bottom of the form is a blue button labeled "Log in".

Log in | Django site admin x

localhost:8000/admin/login/?next=/admin/

Django administration

Username:

Password:

Log in

4. Django 웹 프레임워크

8) 관리자 만들기

```
[root@localhost python_ch3]# python manage.py cratesuperuser
```

```
Unknown command: 'cratesuperuser'
```

```
Type 'manage.py help' for usage.
```

```
[root@localhost python_ch3]#
```

```
[root@localhost python_ch3]# python manage.py createsuperuser
```

```
Username (leave blank to use 'root'): admin
```

```
Email address: kickscar@gmail.com
```

```
Password:
```

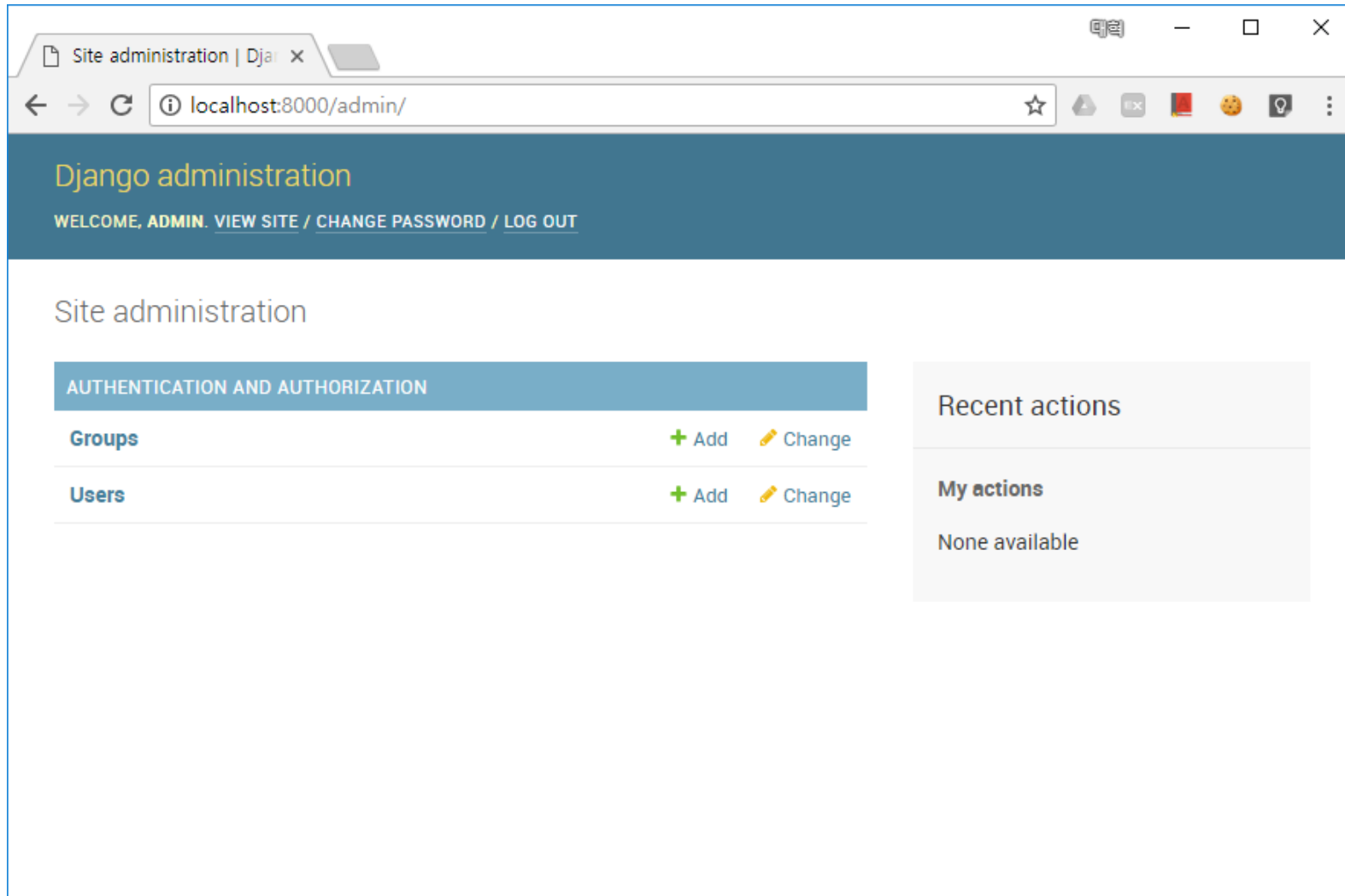
```
Password (again):
```

```
Superuser created successfully.
```

```
[root@localhost python_ch3]#
```

4. Django 웹 프레임워크

9) 관리자 로그인 하기



4. Django 웹 프레임워크

4.7 emailist 애플리케이션 개발하기

4.7.1 애플리케이션 등록 및 타임존 설정

1) 애플리케이션 등록(python_ch2/settings.py)

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

2) 타임존 변경(python_ch2/settings.py)

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

4. Django 웹 프레임워크

4.7.2 Model 작성하기

1) MySQL 드라이버 설치

- MySQLdb: 오래 사용된 드라이버로 안정적이거나 Python3를 지원하지 않는다.
- Mysqlclient: MySQLdb를 개선한 드라이버로 Python3.3 이상을 지원하며 Django 추천 드라이버
- MySQL Connector/Python: Oracle 공식 파이썬 드라이버

2) Database 변경 (python_ch2/settings.py)

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.mysql',  
        'NAME': 'djdb',  
        'USER': 'djdb',  
        'PASSWORD': 'djdb',  
        'HOST': '127.0.0.1',  
        'PORT': '3306'  
    }  
}
```


4. Django 웹 프레임워크

2) 장고에 반영하기

데이터베이스와 관련된 변경 사항은 장고에 반드시 반영해 주어야 한다.

```
[root@localhost python_ch3]# python manage.py migrate
```

-
-
-

```
[root@localhost python_ch3]# python manage.py runserver 0000:8000
```

Performing system checks...

System check identified no issues (0 silenced).

-
-

4. Django 웹 프레임워크

2) 테이블 정의 (emaillist/modes.py)

emaillist 애플리케이션은 Emaillist 하나의 테이블이 필요하다.

테이블 정의는 models.py 파일에 정의한다.

```
class Emaillist(models.Model):  
    first_name = models.CharField(max_length=100)  
    last_name = models.CharField(max_length=50)  
    email = models.CharField(max_length=200)  
  
    def __str__(self):  
        return "Emaillist(%s, %s, %s)" % (self.first_name, self.last_name, self.email)
```

Emaillist 테이블 컬럼과 클래스 변수 간의 매핑

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
first_name	varchar(100)	NO		NULL	
last_name	varchar(50)	NO		NULL	
email	varchar(200)	NO		NULL	

4. Django 웹 프레임워크

3) Admin 페이지에 테이블 반영 (emaillist/admin.py)

```
from django.contrib import admin
from emaillist.models import Emaillist

admin.site.register(Emaillist)
```

4) 데이터베이스에 변경사항 반영

```
(venv) D:\Work\PycharmProjects\python_ch3>python manage.py makemigrations
```

Migrations for 'emaillist':

emaillist\migrations\0001_initial.py

– Create model Emaillist

```
(venv) D:\Work\PycharmProjects\python_ch3>python manage.py migrate
```

Operations to perform:

Apply all migrations: admin, auth, contenttypes, emaillist, sessions

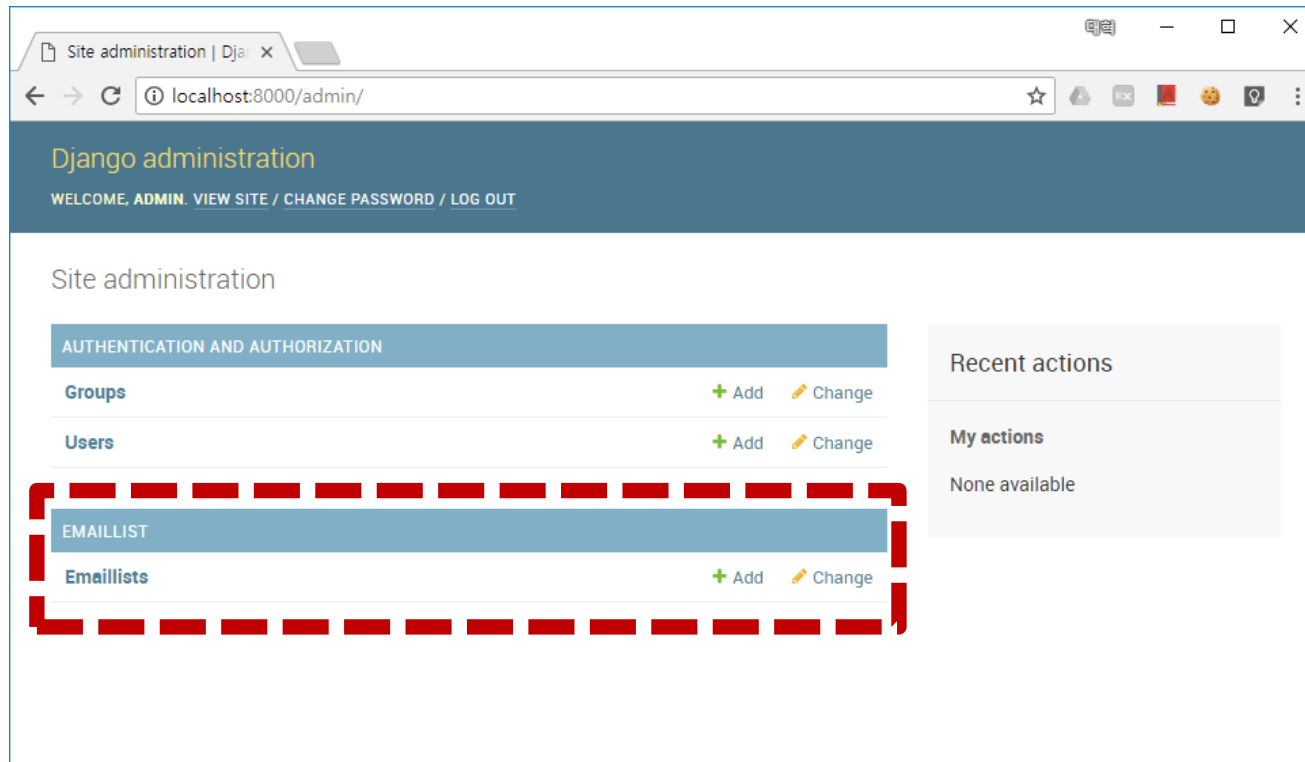
Running migrations:

Applying emaillist.0001_initial... OK

4. Django 웹 프레임워크

5) 테스트 및 확인

```
D:\Work\PycharmProjects\python_ch3\venv\Scripts\python.exe  
D:/DoWork/PycharmProjects/python_ch3/manage.py runserver 0000:8000  
Performing system checks...
```



4. Django 웹 프레임워크

[실습]

View 및 Template 작성하기

파이썬 웹 프로그래밍

01. 웹 프로그래밍의 이해

02. 웹 표준 라이브러리

03. Django 웹 프레임워크

04. Django 활용