

Django

- 파이썬으로 만들어진 무료 오픈소스 웹 애플리케이션 프레임워크
- 설치

```
$ pip install django
```

```
Collecting django
  Downloading https://files.pythonhosted.org/packages/32/ab/22530cc1b2114e6067eece94a333d6c749fa1c56a009f0721e51c181ea53/Django-2.1.2-py3-none-any.whl (7.3MB)
    100% |#####| 7.3MB 1.2MB/s
Collecting pytz (from django)
  Downloading https://files.pythonhosted.org/packages/30/4e/27c34b62430286c6d59177a0842ed90dc789ce5d1ed740887653b898779a/pytz-2018.5-py2.py3-none-any.whl (510kB)
    100% |#####| 512kB 1.4MB/s
Installing collected packages: pytz, django
Successfully installed django-2.1.2 pytz-2018.5
```

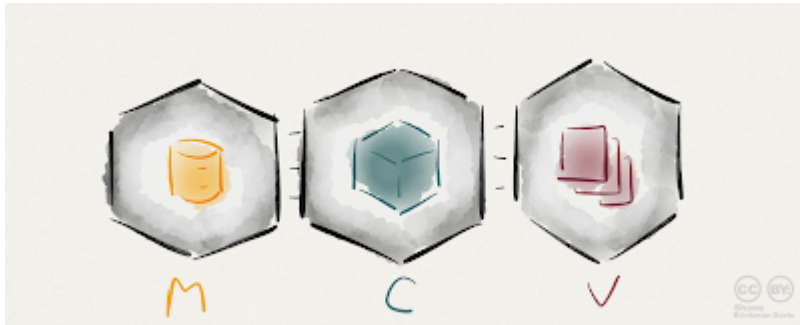
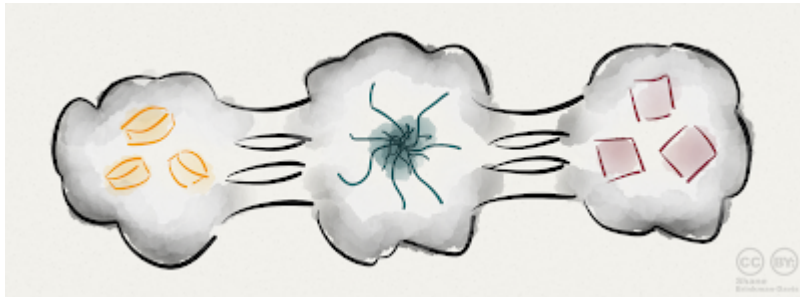
- **Urlresolver**

- 웹 서버에 요청이 오면 장고로 전달되고, 장고가 웹 페이지의 주소를 가져와 무엇을 할지 확인
- 각 URL 에 대해 일일이 확인하기는 비효율적이므로, 패턴으로 일치여부를 판단
 - 일치하는 패턴의 경우 요청을 연관된 함수(*view*)에 넘긴다.



Urlresolver is a POSTMAN

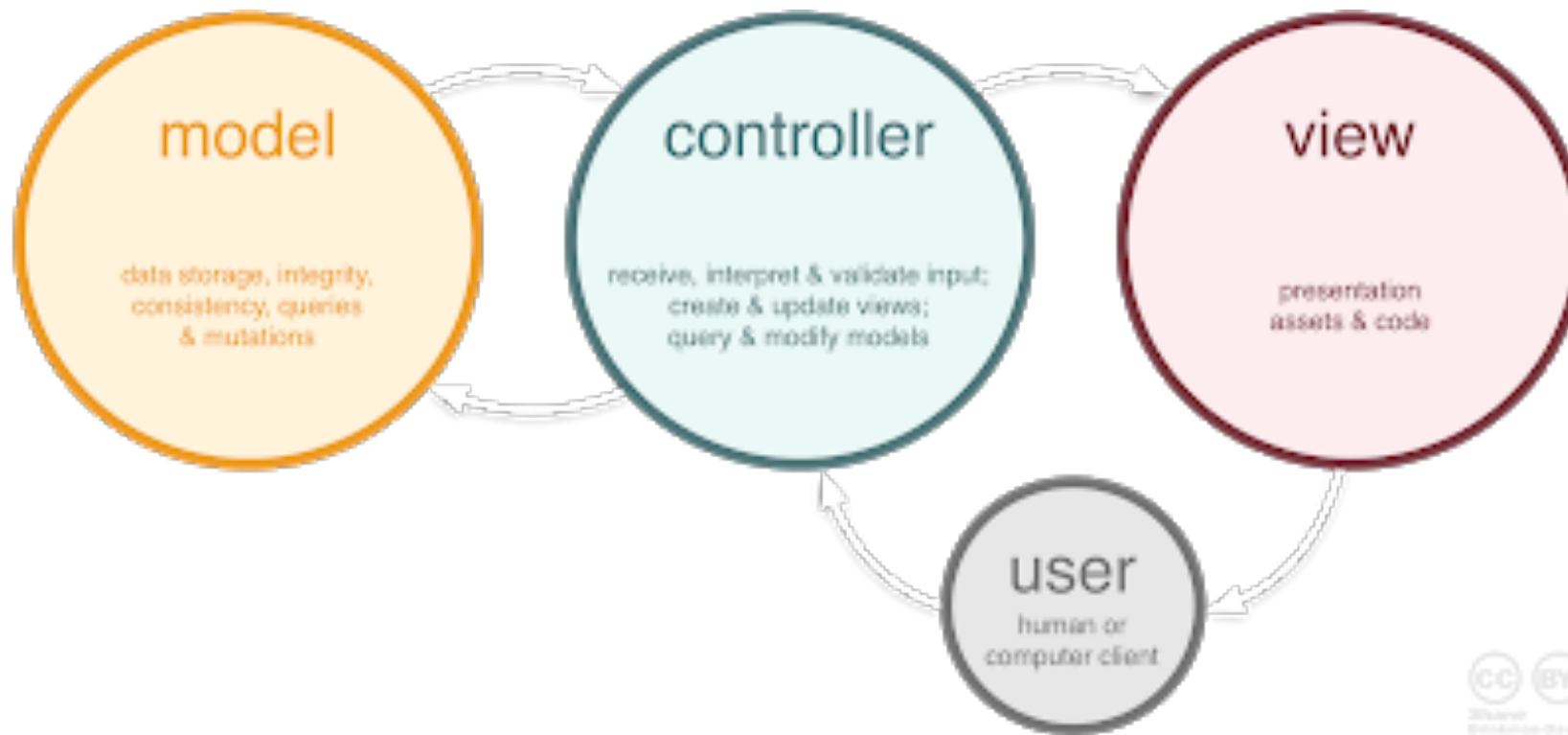
MVC & MTV



- *Model – Template – View*



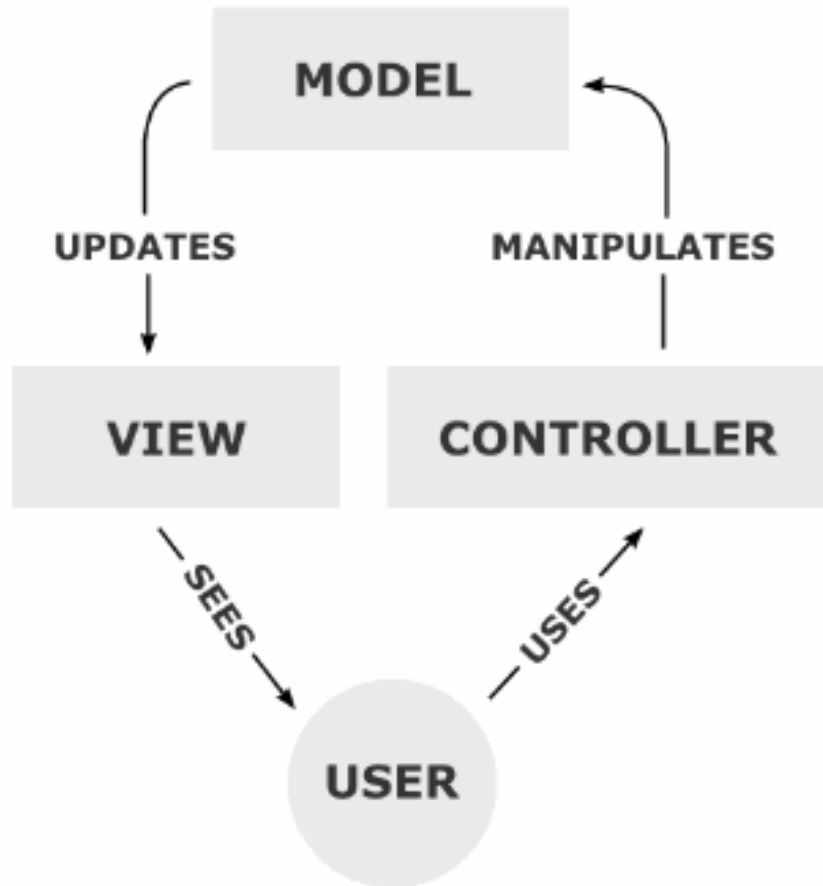
MVC & MTV



The central idea behind MVC is code reusability and separation of concerns.

MVC & MTV

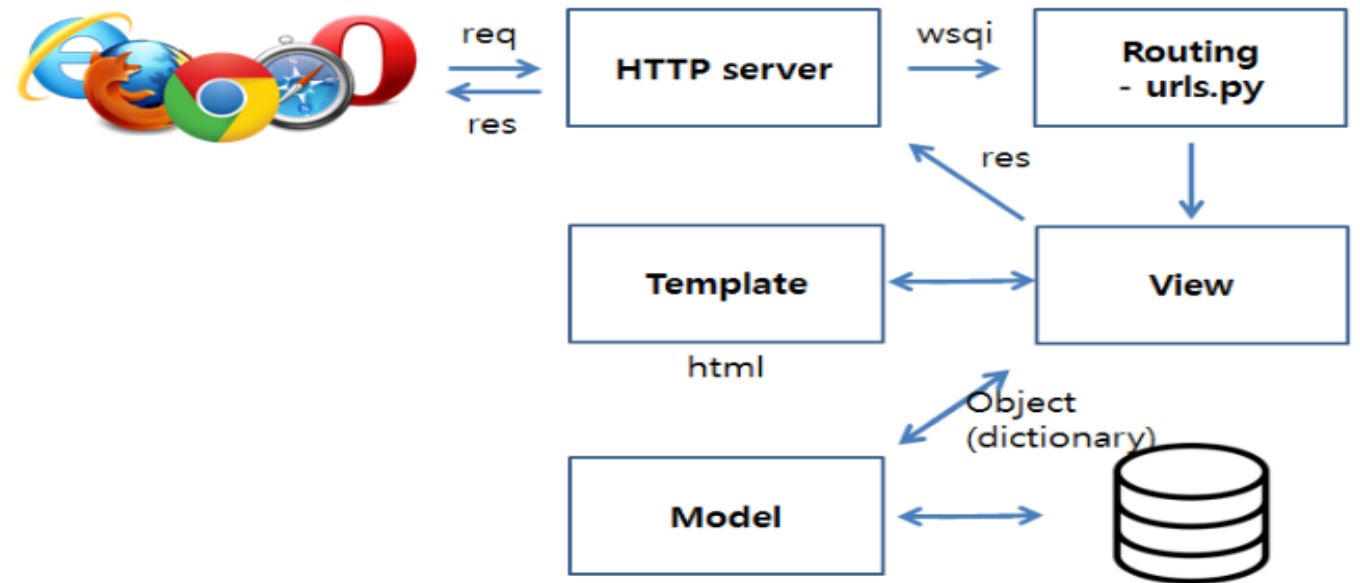
- MVC
 - **M**odel **V**iew **C**ontroller
 - 애플리케이션을 세가지의 역할로 구분한 개발 방법론



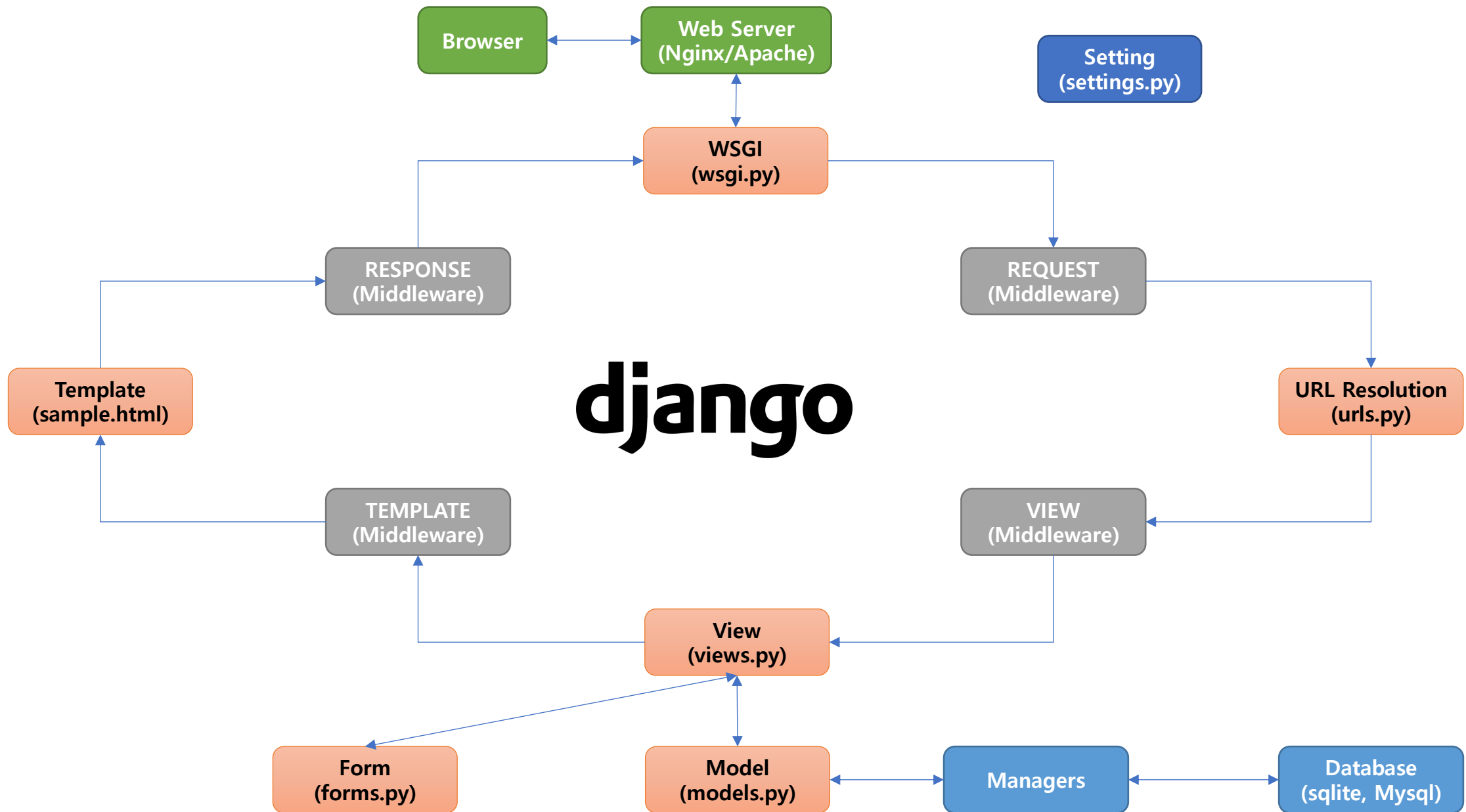
1. 사용자가 웹사이트에 접속한다. (Uses)
2. Controller는 사용자가 요청한 웹페이지를 서비스 하기 위해서 모델을 호출한다. (Manipulates)
3. 모델은 데이터베이스나 파일과 같은 데이터 소스를 제어한 후에 그 결과를 리턴한다.
4. Controller는 Model이 리턴한 결과를 View에 반영한다. (Updates)
5. 데이터가 반영된 View는 사용자에게 보여진다. (Sees)

MVC & MTV

- MTV
 - **M**odel **T**emplate **V**iew
 - Model: 안전하게 데이터를 저장
 - View: 데이터를 유저에게 보여줌 (MVC와는 다름), URL을 Parsing
 - Template: 사용자 입력과 같은 이벤트 처리를 하고, Model, View 업데이트



Django cycle



MTV 코딩 순서

- 화면설계
 - 뷰, 템플릿
 - 테이블 설계
 - 모델
- 1) 프로젝트 생성
 - 프로젝트 및 앱 개발에 필요한 디렉토리와 파일 생성
 - 2) 모델 생성
 - 테이블 관련 내용 개발 (models.py, admin.py)
 - 3) URL conf 생성
 - URL 및 뷰 매핑 관계를 정의 (urls.py)
 - 4) View 생성
 - 애플리케이션 로직 개발 (views.py)
 - 5) Template 생성
 - 화면 UI 개발 (templates/ 디렉토리 하위의 *.html 파일)

settings.py

- 데이터베이스 설정
 - Default로 Sqlite3 사용
- 템플릿 항목 설정
 - TEMPALTES 항목 지정
- 정적 파일 항목 설정
 - STATIC_URL 등 관련 항목 지정
- 애플리케이션 등록
 - 프로젝트에 포함되는 모든 애플리케이션 등록
- 타임존 지정
 - UTC 변경 (한국)

models.py

- 테이블을 정의하는 파일
- ORM (Object Relation Mapping) 사용
 - 테이블을 클래스로 매핑하여 CRUD 기능을 클래스 객체에 대해 수행 → DB 반영
- 테이블을 클래스, 테이블의 컬럼은 클래스 변수로 매핑
 - django.db.models.Model 클래스 상속
- models.py에서 DB 변경 사항 발생시, 실제 DB에서도 반영
 - django 1.7부터 마이그레이션 기능 사용
 - ex) makemigrations, migrate 사용

URLconf 주요사항

- URL과 View(함수 또는 메소드)를 매핑하는 urls.py 파일
- 프로젝트 URL과 앱 URL로 구성하는 것 추천
- {% url %} 템플릿 태그 사용

views.py

- 뷰 로직을 생성하는 파일
- 함수(Function-based view) 또는 클래스(Class-based view)로 생성 가능

templates

- 웹 화면(페이지) 별로 템플릿 파일(*.html)이 필요
- TEMPLATES 설정의 DIR 항목에 지정된 디렉토리에 앱 템플릿 파일 저장
 - ex) templates/ 디렉토리

Admin

- 테이블 내용을 열람하고 수정하는 기능을 제공하는 사이트
- User 및 Group 테이블 관리
 - settings.py에 django.contrib.auth 애플리케이션이 등록
- 사용자가 비즈니스 로직 개발에 필요한 테이블 관리(CRUD)

개발용 웹 서버

- 테스트용 runserver 제공
- 상용화를 위해서는 Apache or Nginx 등의 상용 서버로 변경

애플리케이션 설계

- 화면 설계
 - templates/ 디렉토리 하위의 *.html 파일

애플리케이션 설계

- 테이블 설계
 - models.py 작성

필드명	타입	제약 조건	설명
id	Integer	PK, Auto Increment	기본 키(Primary Key)
title	CharField(10)	Blank, Null	북마크 제목
url	URLField	Unique	북마크 URL

애플리케이션 설계

- 로직 설계
 - 처리 흐름을 설계
 - URL을 받아서 최종 HTML 템플릿 파일을 만드는 과정



애플리케이션 설계

- URL 설계
 - URLconf 코딩에 반영
 - urls.py에 작성

URL 패턴	뷰 이름	템플릿 파일 이름
/bookmark/	BookmarkLV(ListView)	bookmark_list.html
/bookmark/99/*	BookmarkDV(DetailView)	bookmark_detail.html
/admin/	(Django 제공)	

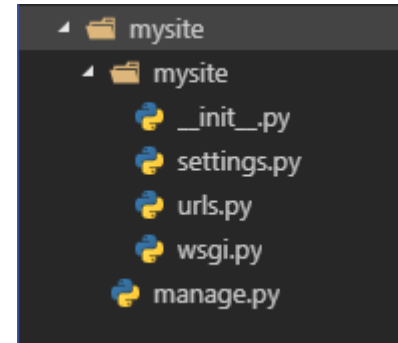
애플리케이션 설계

- 작업/코딩 순서
 - 1) 테이블 정의에 관련된 Model 코딩
 - 2) URLconf
 - 3) 뷰 작성
 - 4) 템플릿 작성

작업 순서	관련 명령/파일	필요한 작업 내용
구조 생성	startproject settings.py migrate createsuperuser startapp settings.py	mysite 프로젝트 생성 프로젝트 설정 항목 변경 User/Group 테이블 생성 프로젝트 관리자인 슈퍼유저 생성 북마크 앱 생성 북마크 앱 등록
모델 코딩	models.py admin.py makemigrations migrate	모델(테이블) 정의 Admin 사이트에 모델 등록 모델을 데이터베이스에 반영
URLconf 코딩	urls.py	URL 정의
뷰 코딩	views.py	뷰 로직 작성
템플릿 코딩	templates/ 디렉토리	템플릿 파일 작성
그 외	-	

- 프로젝트 생성

\$ django-admin.py startproject mysite



- 프로젝트 설정 파일 변경
 - settings.py
 - DATABASES
 - TEMPLATES
 - STATIC_URL
 - TIME_ZONE
 - MEDIA_URL
 - INSTALLED_APPS

- 프로젝트 설정 파일 변경
 - settings.py

```
# Database
# https://docs.djangoproject.com/en/2.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

- 프로젝트 설정 파일 변경
 - settings.py

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

- 프로젝트 설정 파일 변경
 - settings.py

```
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.1/howto/static-files/

STATIC_URL = '/static/'
```


- 프로젝트 설정 파일 변경
 - settings.py

```
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.1/howto/static-files/

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

- 프로젝트 설정 파일 변경
 - settings.py

```
# Internationalization
# https://docs.djangoproject.com/en/2.1/topics/i18n/

LANGUAGE_CODE = 'ko-kr'

TIME_ZONE = 'Asia-Seoul'
```

- 프로젝트 설정 파일 변경
 - settings.py

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

- 애플리케이션 생성

\$ ***django-admin startproject mysite***

- 테이블 생성

- 사용자와 사용자의 권한 그룹 테이블이 반드시 필요

```
$ 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
```

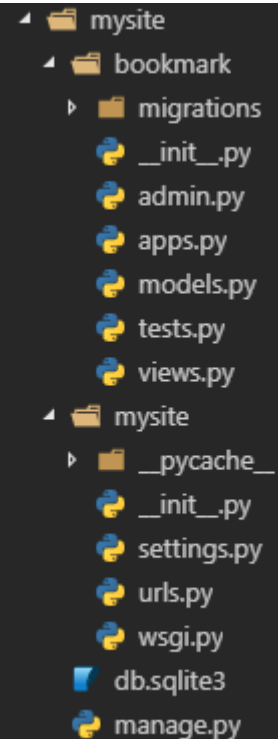
```
  Applying sessions.0001_initial... OK
```

- 슈퍼유저 생성

```
$ python manage.py createsuperuser
사용자 이름 (leave blank to use 'edowo'): down
이메일 주소: edowon@test.com
Password:
Password (again):
Superuser created successfully.
```

- 애플리케이션 생성

```
$ python manage.py startapp bookmark
```



```
mysite
├── bookmark
│   ├── migrations
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── models.py
│   ├── tests.py
│   └── views.py
├── mysite
│   ├── __pycache__
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   ├── wsgi.py
│   ├── db.sqlite3
│   └── manage.py
```

- 애플리케이션 등록
 - settings.py

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

- 테이블 정의
 - models.py

```
from django.db import models

# Create your models here.
class Bookmark(models.Model):
    title = models.CharField(max_length=100, blank=True, null=True)
    url = models.URLField('url', unique=True)

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

- Admin 사이트에 테이블 반영
 - admin.py

```
from django.contrib import admin
from bookmark.models import Bookmark

# Register your models here.
class BookmarkAdmin(admin.ModelAdmin):
    list_display = ('title', 'url')

admin.site.register(Bookmark, BookmarkAdmin)
```


- 데이터베이스 변경 사항 반영

```
$ python manage.py makemigrations  
Migrations for 'bookmark':  
  bookmark\migrations\0001_initial.py  
    - Create model Bookmark
```

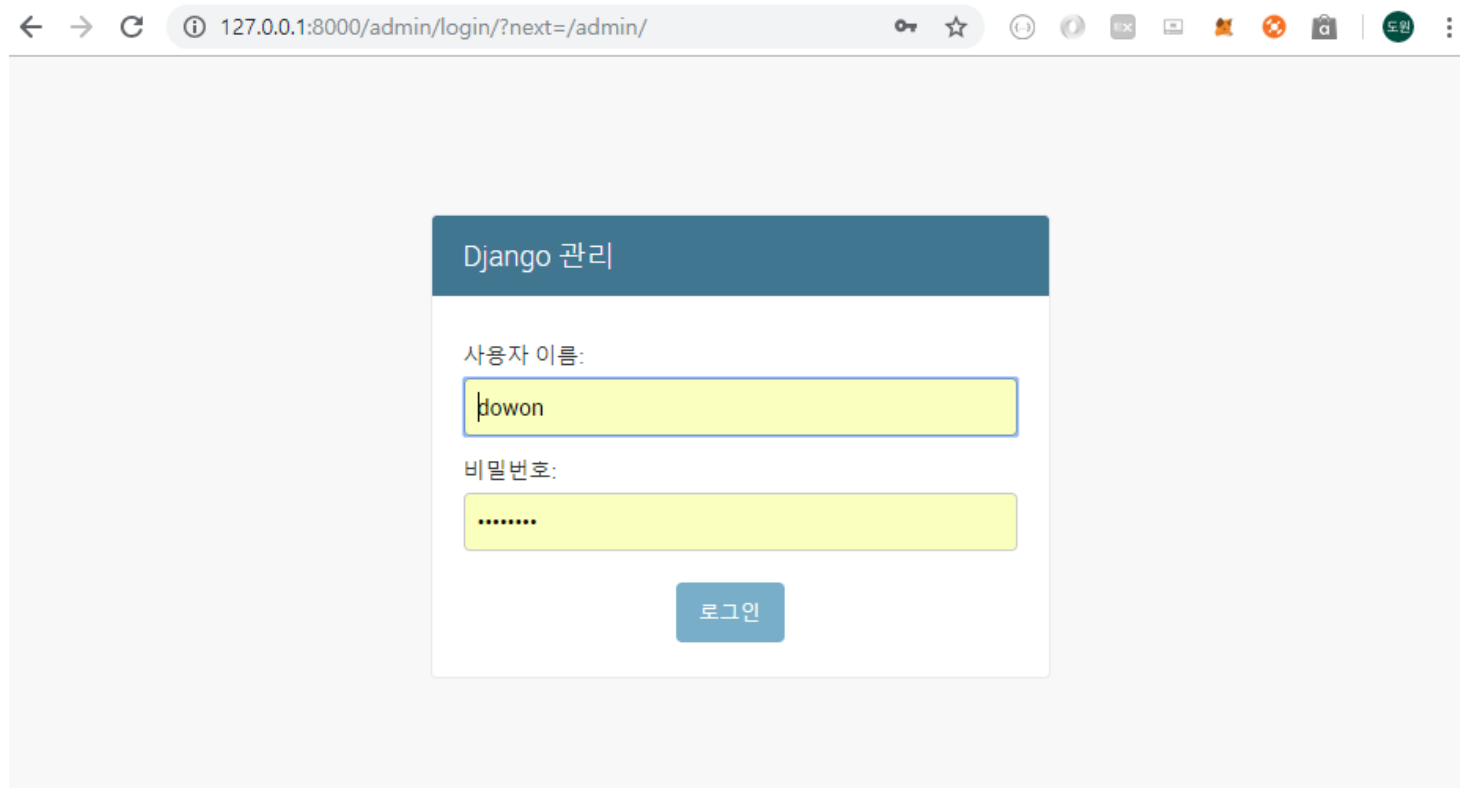
```
$ python manage.py migrate  
Operations to perform:  
  Apply all migrations: admin, auth, bookmark, contenttypes, sessions  
Running migrations:  
  Applying bookmark.0001_initial... OK
```

- 테이블 모습 확인

```
$ python manage.py runserver 0.0.0.0:8000
Performing system checks...

System check identified no issues (0 silenced).
October 23, 2018 - 01:26:31
Django version 2.1.2, using settings 'mysite.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CTRL-BREAK.
```

- 테이블 모습 확인



- 테이블 모습 확인

Django 관리

환영합니다, **DOWON**. [사이트 보기](#) / [비밀번호 변경](#) / [로그아웃](#)

사이트 관리

BOOKMARK

Bookmarks

+ 추가 ✎ 변경

인증 및 권한

그룹

+ 추가 ✎ 변경

사용자(들)

+ 추가 ✎ 변경

최근 활동

나의 활동

이용할 수 없습니다.

```
$ python manage.py runserver 0.0.0.0:8000
Performing system checks...
```

```
System check identified no issues (0 silenced).
October 23, 2018 - 01:26:31
Django version 2.1.2, using settings 'mysite.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CTRL-BREAK.
[23/Oct/2018 01:27:02] "GET /admin/logout/ HTTP/1.1" 200 1209
[23/Oct/2018 01:27:04] "GET /admin/ HTTP/1.1" 302 0
[23/Oct/2018 01:27:04] "GET /admin/login/?next=/admin/ HTTP/1.1" 200 1839
[23/Oct/2018 01:27:39] "POST /admin/login/?next=/admin/ HTTP/1.1" 302 0
[23/Oct/2018 01:27:39] "GET /admin/ HTTP/1.1" 200 3771
```

- URLconf

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

from bookmark.views import BookmarkLV, BookmarkDV

urlpatterns = [
    path('admin/', admin.site.urls),

    path('bookmark/', BookmarkLV.as_view(), name='index'),
    path('bookmark/<int:pk/>', BookmarkDV.as_view(), name='detail'),
]
```

- View
 - views.py

```
from django.shortcuts import render

from django.views.generic import ListView, DetailView
from bookmark.models import Bookmark

# Create your views here.

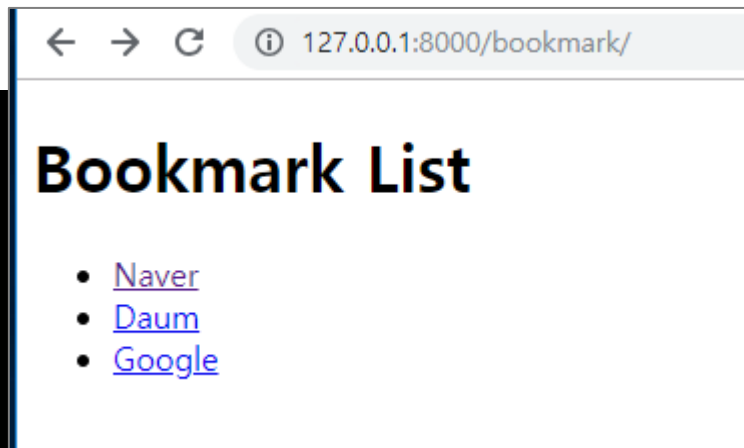
#--- ListView
class BookmarkLV(ListView):
    model = Bookmark

#--- DetailView
class BookmarkDV(DetailView):
    model = Bookmark
```

- bookmark_list.html

```
<!DOCTYPE html>
<html>
<head>
<title>Django Bookmark List</title>
</head>
<body>
<div id="content">
    <h1>Bookmark List</h1>

    <ul>
        {% for bookmark in object_list %}
            <li><a href="{% url 'detail' bookmark.id %}">{{ bookmark }}</a></li>
        {% endfor %}
    </ul>
</div>
</body>
</html>
```



- bookmark_detail.html

```
<!DOCTYPE html>
<html>
<head>
<title>Django Bookmark Detail</title>
</head>
<body>
<div id="content">
    <h1>{{ object.title }}</h1>

    <ul>
        <li>URL: <a href="{{ object.url }}">{{ object.url }}</a></li>
    </ul>
</div>
</body>
</html>
```

Naver

- URL: <http://www.naver.com>

- Admin 사이트에서 데이터 입력

Django 관리

환영합니다, **DOWON**. [사이트 보기](#) / [비밀번호 변경](#) / [로그아웃](#)

[홈](#) > [Bookmark](#) > [Bookmarks](#) > [bookmark 추가](#)

✔ Bookmark "Daum"가 성공적으로 추가되었습니다. 아래에서 다른 bookmark을 추가할 수 있습니다.

bookmark 추가

Title:

Google

Url:

https://www.google.co.kr|

저장 및 다른 이름으로 추가

저장 및 편집 계속

저장

- Admin 사이트에서 데이터 입력

Django 관리

환영합니다, **DOWON**. [사이트 보기](#) / [비밀번호 변경](#) / [로그아웃](#)

홈 > Bookmark > Bookmarks

✔ Bookmark "Google"가 성공적으로 추가되었습니다.

변경할 bookmark 선택

BOOKMARK 추가 +

액션:

----- ▼

실행

 3 중 아무것도 선택되지 않았습니다.

<input type="checkbox"/>	TITLE	URL
<input type="checkbox"/>	Google	https://www.google.co.kr
<input type="checkbox"/>	Daum	http://www.daum.net
<input type="checkbox"/>	Naver	http://www.naver.com

3 bookmarks