

Python Django

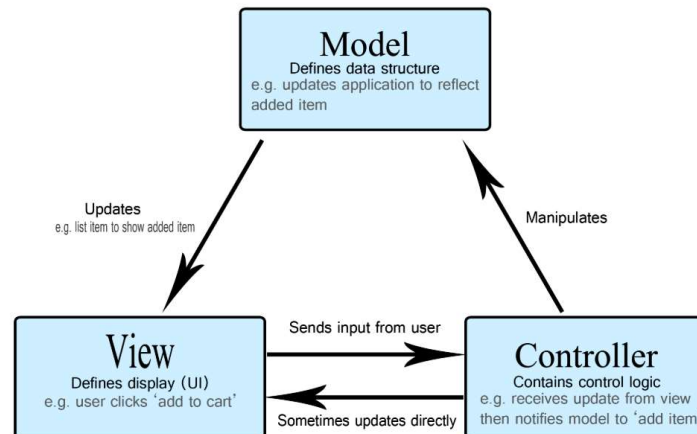
➤ Django

- 파이썬으로 만들어진 무료 오픈소스 웹 애플리케이션 프레임워크(web application framework)
- 쉽고 빠르게 웹사이트를 개발할 수 있도록 돕는 구성요소로 이루어진 웹 프레임워크
- 1. 웹 서버에 요청이 오면 장고로 전달
- 2. 장고 urlresolver는 웹 페이지의 주소를 가져와 무엇을 할지 확인
- 3. 일치하는 패턴이 있으면, 장고는 해당 요청을 관련된 함수(view)에 넘겨줍니다

Python Django

➤ MVC(Model-View-Controller) 패턴

- 소프트웨어 공학에서 사용되는 소프트웨어 디자인 패턴
- 사용자 인터페이스로부터 비즈니스 로직을 분리하여 애플리케이션의 시각적 요소나 그 이면에서 실행되는 비즈니스 로직을 서로 영향 없이 쉽게 고칠 수 있는 애플리케이션을 만들 수 있다.
- **모델(Model)**은 애플리케이션의 정보(데이터)를 나타내며, 데이터베이스나 파일 등의 데이터 소스를 제어한다
- **뷰(View)**는 텍스트, 체크박스 항목 등과 같은 사용자 인터페이스 요소를 나타내고, 모델로부터 제공된 데이터를 반영하여 사용자에게 보여주게 되는 부분이다.
- **컨트롤러(Controller)**는 데이터와 비즈니스 로직 사이의 상호동작을 관리(사용자의 요청을 파악하여 그에 맞는 데이터를 모델에 의뢰하고, 그것을 뷰에 반영하여 사용자에게 제공) 한다



Python Django

➤ MTV 패턴

- Django는 Model, Template, View라는 MTV 패턴을 따르고 있는데, MTV은 MVC (Model View Controller)와 유사한 점이 많다.
- Django는 Controller의 역할을 Django Framework 자체에서 한다
- MTV에서의 Model은 데이터를 표현하는데 사용되며, 하나의 모델 클래스는 DB에서 하나의 테이블로 표현된다. MTV의 View는 HTTP Request를 받아 그 결과인 HTTP Response를 리턴하는 컴포넌트로서, Model로부터 데이터를 읽거나 저장할 수 있으며, Template을 호출하여 데이터를 UI 상에 표현하도록 할 수 있다.
- MTV의 Template은 Presentation Logic 만을 갖는데 HTML을 생성하는 것을 목적으로 하는 컴포넌트이다.
- 모델(Model)은 데이터를 표현하는데 사용되며, Python 클래스 형식으로 정의된다. 하나의 모델 클래스는 데이터베이스에서 하나의 테이블로 표현된다.
- 템플릿(Template)은 사용자에게 보여지는 부분만을 담당한다.
- 뷰(View)는 HTTP Request를 받아 HTTP Response를 리턴하는 컴포넌트로, 모델로부터 데이터를 읽거나 저장할 수 있다

Python Django

➤ Django MTV 개발 방식

- 모델 - 테이블 정의

Python 클래스이며, `django.db.models.Model` 클래스를 상속받는 서브클래스이다.

모델 클래스의 속성들은 데이터베이스의 필드가 된다.

Django 는 모델 클래스를 통해 데이터베이스에 접근할 수 있는 API 를 자동적으로 생성해준다.

- 템플릿 - 사용자가 보게 될 화면의 모습 정의

- 뷰 - 애플리케이션의 제어 흐름 및 처리 로직을 정의

- 모듈 간에 독립성을 유지할 수 있고, 소프트웨어 개발 원칙 느슨한 결합 설계의 원칙에 부합

- 장고에서 프로젝트를 생성하기 위해 `startproject` 및 `startapp` 명령을 실행하면, 자동으로 프로젝트 뼈대에 해당하는 디렉토리와 파일을 만들어 줍니다.

- MTV 방식으로 개발할 수 있도록 골격이 만들어지고 파일 이름도 Django에서 생성해줌

- `models.py` 파일에 모델을 정의

- 템플릿은 `templates` 디렉토리의 하위에 `*.html` 파일로 저장

- `views.py` 파일에 뷰 정의

Python Django

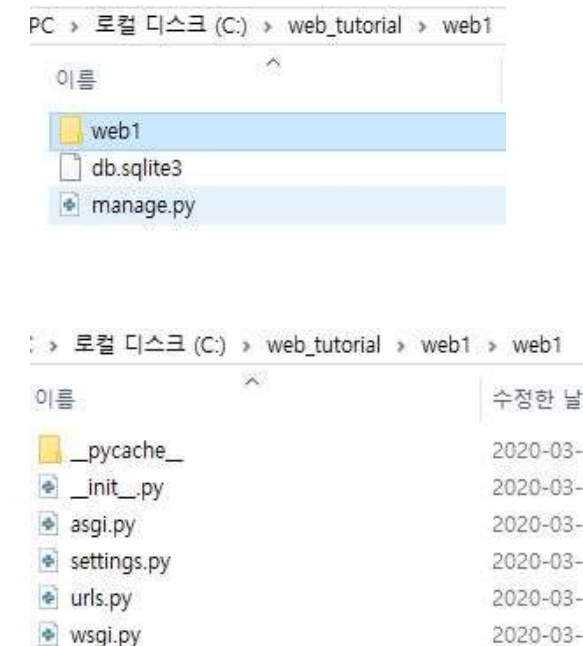
➤ Django 설치

```
# 장고설치
$ pip install django
$ python -m django --version

#장고 프로젝트 시작하기
$ django-admin startproject 웹프로젝트명

#웹 프로젝트 폴더 아래 테스트 서버를 구동 실행
$ python manage.py runserver

# 웹 브라우저의 주소창에
127.0.0.1:8000 또는 localhost:8000
```



- Django에서 새로운 웹 프로젝트를 만들기 위해서는 django-admin.py라는 Django 관리자 모듈을 사용한다.
- 프로젝트를 만들 디렉토리로 이동한 후, 아래와 같이 "django-admin startproject 프로젝트명" 를 실행하여 새 프로젝트를 생성한다.

Python Django

➤ Creating a project

```
$ django-admin startproject mysite
```

- 외부 mysite/ : 루트 디렉토리는 프로젝트의 컨테이너
- manage.py: Django 프로젝트와 다양한 방식으로 상호 작용할 수 있는 명령 줄 유틸리티
- 내부 mysite/ : 디렉토리는 프로젝트의 실제 파이썬 패키지
- mysite/settings.py: Django 프로젝트의 설정 / 구성
- mysite/urls.py : Django 프로젝트의 URL 선언
- mysite/asgi.py : ASGI 호환 웹 서버가 프로젝트를 제공하기위한 진입 점
- mysite/wsgi.py: WSGI 호환 웹 서버가 프로젝트를 제공하기위한 진입 점

```
$ python manage.py runserver  
# default port 8000 아닌 다른 포트 사용 명령  
$ python manage.py runserver 8080
```

```
mysite/  
    manage.py  
    mysite/  
        __init__.py  
        settings.py  
        urls.py  
        asgi.py  
        wsgi.py
```

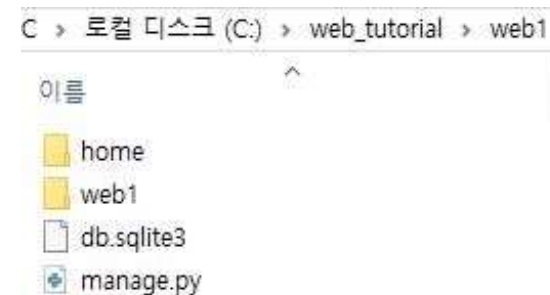
- 'mysite.settings'설정을 사용하는 Django 버전 3.0<http://127.0.0.1:8000/> 에서 개발 서버 시작
- CONTROL-C로 서버 종료

Python Django

➤ Django App

- Django App은 Django 프로젝트 내에서 사용하는 "Python 패키지"이다
- Django App 패키지 안에 독자적인 모델(model), 뷰(view), 템플릿(template), URL 매핑 등을 가지고 있으며, 하나의 Django 프로젝트는 모듈화된 여러 개의 앱들로 구성된다.
- 규모가 큰 Django 프로젝트는 보통 여러 개의 Django App들을 모듈화하여 구성
- 모듈화된 App들로 구성하면 개발 및 유지 보수가 효율적
- 잘 모듈화된 App은 여러 웹 프로젝트에서 쉽게 재사용할 수도 있다.
- 하나의 Django App을 생성하기 위해서는 "manage.py startapp App명" 를 실행

```
python manage.py startapp home
```



Python Django

➤ MTV 코딩 순서

- **프로젝트 뼈대 만들기** : 프로젝트 및 앱 개발에 필요한 디렉토리와 파일 생성
- **모델 코딩하기** : 테이블 관련 사항을 개발(models.py, admin.py 파일)
- **URLconf 코딩하기** : URL 및 뷰 매핑 관계를 정의 (urls.py 파일)
- **뷰 코딩하기** : 애플리케이션 로직 개발(views.py 파일)
- **템플릿 코딩하기** : 화면 UI 개발 (templates/디렉토리 하위의 *.html 파일들)

Python Django

➤ 공통 모듈 구조

웹 프로젝트 폴더/

`__init__.py`

`admin.py`

`apps.py`

`migrations/`

`models.py`

`tests.py`

`views.py`

`urls.py`

`forms.py`: 입력 폼 선언

`behaviors.py`: 모델 믹스인 위치에 대한 옵션

`constants.py`: 앱에 쓰이는 상수 선언

`decorators.py`: 데코레이터

`db/`: 여러 프로젝트에서 사용되는 커스텀 모델이나 컴포넌트

`fields.py`: 폼 필드

`factories.py`: 테스트 데이터 팩토리 파일

`helpers.py`: 뷰와 모델 파일을 가볍게 하기 위해 유틸리티 함수 선언

`managers.py`: `models.py`가 너무 커질 경우 커스텀 모델 매니저가 위치

`signals.py`: 커스텀 시그널

`viewmixins.py`: 뷰 모듈과 패키지를 더 가볍게하기 위해 뷰 믹스인을 이 모듈로 이전

`urls.py`: 앱의 URL 패턴 선언

Python Django

➤ 공통 모듈 구조

- Django 프로젝트 기본 설정 옵션은 settings.py 모듈 파일 하나로 설정한다.
- settings.py 파일의 내용
 - 데이터베이스 설정 : 디폴트로 SQLite3 데이터베이스 엔진을 사용하는 것을 지정
 - 템플릿 설정 : TEMPLATES 항목으로 지정
 - 지역 시각 및 다국어 설정 : 최초에는 세계표준(UTC)로 설정되어 있는데 , 한국 시간으로 변경해야 합니다.
 - 애플리케이션의 등록 : 개발하는 앱, 프로젝트에 포함되는 애플리케이션들을 모두 설정 파일에 등록
 - 정적 파일 설정 : STATIC_URL 등 관련 항목을 지정
 - 미디어 파일 설정
 - 루트 디렉토리를 포함한 각종 디렉토리의 위치, 로그의 형식, 디버그 모드, 보안 관련 사항 등 프로젝트 전반적인 사항을 설정

Python Django

➤ 데이터베이스 설정

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

- DATABASES ENGINE - 'django.db.backends.sqlite3', 'django.db.backends.postgresql', 'django.db.backends.mysql', or 'django.db.backends.oracle'

Python Django

➤ 템플릿 설정

```
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',  
            ],  
        },  
    },  
]
```

Python Django

➤ 지역시각 및 다국어 설정

- 기본 설정 값

```
LANGUAGE_CODE = 'en-us'  
TIME_ZONE = 'UTC'  
USE_I18N = True  
USE_L10N = True  
USE_TZ = True
```

- 한국어로 설정 변경

```
LANGUAGE_CODE = 'ko-kr'  
TIME_ZONE = 'Asia/Seoul'  
USE_I18N = True  
USE_L10N = True  
USE_TZ = True
```

Python Django

➤ 정적 파일 설정

- 기본 설정 값

```
STATIC_URL = '/static/'
```

- STATIC_URL 변수를 추가

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

Python Django

➤ 미디어 파일 설정

- 프로젝트 기본 설정값은 없으며 파일 업로드 기능 구현을 위해 다음 설정을 추가

```
MEDIA_URL = '/media/'  
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

- runserver 테스트 서버 구동 시에 /media 경로를 올바로 찾지 못하는 문제가 발생한다. 따라서 URL 패턴 매칭을 해주는 다음 코드가 필요하다.

```
from django.conf.urls.static import static  
  
urlpatterns = [  
    ....  
]  
  
if settings.DEBUG:  
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Python Django

➤ 애플리케이션 등록

- 앱을 생성한 경우 해당 앱 디렉토리 안에 기본 생성 파일

```
웹앱 이름 폴더/  
migrations/  
__init__.py  
admin.py  
apps.py : 애플리케이션의 설정 클래스를 정의  
models.py  
tests.py  
views.py
```

- 애플리케이션의 등록

```
INSTALLED_APPS = [  
    'django.contrib.admin',          ##관리 사이트  
    'django.contrib.auth',          #인증 시스템  
    'django.contrib.contenttypes',  #컨텐츠 유형을위한 프레임 워크.  
    'django.contrib.sessions',      #세션 프레임 워크  
    'django.contrib.messages',      #메시징 프레임 워크  
    'django.contrib.staticfiles',    #정적 파일 관리를위한 프레임 워크  
    '웹앱이름.apps.웹앱이름Config',  
]
```

- migrate 명령은 INSTALLED_APPS 설정을 확인하고 mysite/settings.py 파일의 데이터베이스 설정 및 앱과 함께 제공된 데이터베이스 마이그레이션에 따라 필요한 데이터베이스 테이블을 만듭니다

Python Django

➤ Django View

- Django에서의 뷰(View)는 다른 일반 MVC Framework에서 말하는 Controller와 비슷한 역할을 한다.
- View는 필요한 데이터를 모델 (혹은 외부)에서 가져와서 적절히 가공하여 웹 페이지 결과를 만들도록 컨트롤하는 역할을 한다.
- View들은 Django App 안의 views.py 라는 파일에 정의
- 함수형 뷰(function-based view)와 클래스형 뷰(class-based view)로 정의 가능
- 각 함수가 하나의 View를 정의한다.
- 각 View는 HTTP Request를 입력 파라미터로 받아들이고, HTTP Response를 리턴 한다.

```
from django.http import HttpResponse

def index(request):
    return HttpResponse("<h1>Hello, World!</h1>")
```

```
from django.http import Http404, HttpResponseNotFound

def error(request):
    #return HttpResponseNotFound('<h1>not found</h1>')
    raise Http404("Not Found")
```

Python Django

➤ Django View

- `render()` 함수는 요청 객체를 첫 번째 인수로, 템플릿 이름을 두 번째 인수로, 문맥을 딕셔너리를 선택적 세 번째 인수로 사용합니다.
- `HttpResponse` 주어진 컨텍스트로 렌더링 된 주어진 템플릿 의 객체를 반환합니다 .

Python Django

➤ Django 템플릿 (Template)

- Django에서의 View가 다른 MVC Framework에서의 Controller와 유사한 역할을 한다면, Django에서의 템플릿 (Template)은 MVC Framework에서의 View와 비슷한 역할을 한다.
- 템플릿 (Template)은 View로부터 전달된 데이터를 템플릿에 적용하여 Dynamic 한 웹페이지를 만드는데 사용된다.
- Template은 HTML 파일로서 Django App 폴더 밑에 "templates" 라는 서브폴더를 만들고 그 안에 템플릿 파일(*.html)을 생성한다. (단일 App이거나 동일 템플릿명이 없는 경우 사용할 수 있다.)
- 프로젝트 템플릿 디렉토리 : TEMPLATES 설정의 DIRS 항목에 지정된 디렉토리
- 앱 템플릿 디렉토리 : 각 애플리케이션 디렉토리마다 존재하는 templates 디렉토리
- 템플릿 파일을 찾는 순서 : 프로젝트 템플릿 디렉토리를 먼저 검색 > 앱 템플릿 디렉토리를 검색 > 앱 템플릿 디렉토리내에서 INSTALLED_APPS 설정 항목에 등록된 순서대로 검색
- 템플릿은 물론 순수하게 HTML로만 쓰여진 Static HTML 파일일 수는 있지만, 거의 대부분의 경우 View로부터 어떤 데이터를 전달받아 HTML 템플릿 안에 그 데이터를 동적으로 치환해서 사용한다.

```
from django.shortcuts import render
```

```
def index(request):  
    msg = 'My Message'  
    return render(request, 'home/index.html', {'message': msg})
```

#body 태그 안에 message를 보면 {{ }} 으로 둘러싸인 것을 볼 수 있는데, Django의 템플릿에서 {{ 변수명 }} 은 해당 변수의 값을 그 자리에 치환하라는 의미

Python Django

➤ 템플릿 셋팅

- Django에서는 여러 템플릿 엔진을 선택하여 사용할 수 있다
- 템플릿 엔진 셋팅은 Django 프로젝트의 settings.py 에서 할 수 있다.
- 디폴트 Django 템플릿 엔진을 사용하기 위해서는 settings.py 파일의 TEMPLATES 섹션에서 BACKEND를 `django.template.backends.django.DjangoTemplates` 로 설정한다 (기본 설정)

Python Django

➤ Django 템플릿 언어 (Django Template Language)

- **템플릿 변수** : {{ 와 }} 으로 둘러 싸여 있는 변수
변수의 값이 해당 위치에 치환된다.
변수에는 Primitive 데이터를 갖는 변수 혹은 객체의 속성 등을 넣을 수 있다.

```
<h4>  
  Name : {{ name }}  
  Type : {{ vip.key }}  
</h4>
```

- **템플릿 태그** : {% 와 %} 으로 둘러 싸여 있고 태그 안에는 if, for 루프 같은 Flow Control 문장에서부터 웹 컨트롤 처럼 내부 처리 결과를 직접 덤프하는 등등 여러 용도로 쓰일 수 있다.

```
{% if count > 0 %}  
  Data Count = {{ count }}  
{% else %}  
  No Data  
{% endif %}  
  
{% for item in dataList %}  
  <li>{{ item.name }}</li>  
{% endfor %}  
  
{% csrf_token %}
```

Python Django

➤ Django 템플릿 언어 (Django Template Language)

- **템플릿 필터** : 변수의 값을 특정한 포맷으로 변형하는 기능을 한다.

날짜 포맷 지정

```
{{ createDate|date:"Y-m-d" }}
```

소문자로 변경

```
{{ lastName|lower }}
```

- **코멘트** : 한 라인에 코멘트를 적용할 때는 코멘트를 {# 과 #} 으로 둘러싸면 된다.
복수 라인 문장을 코멘트할 경우는 문장들을 {% comment %} 태그와 {% endcomment %}로 둘러싸면 된다.

{# 1 라인 코멘트 #}

```
{% comment %}
```

```
<div>
```

```
<p>
```

불필요한 블록

```
</p>
```

```
<span></span>
```

```
</div>
```

```
{% endcomment %}
```

Python Django

➤ Django 템플릿 언어 (Django Template Language)

- **HTML Escape** : HTML 내용 중에 <, >, ', ", & 등과 같은 문자들이 있을 경우 이를 그 문자에 상응하는 HTML Entity로 변환해 주어야 하는데, Django 템플릿에서 이러한 작업을 자동으로 처리해 주기 위해 {% autoescape on %} 템플릿 태그나 escape 라는 필터를 사용한다.
- HTML escape 혹은 HTML 인코딩 기능을 사용하지 않고, <, >, ', ", & 이 들어간 문자열을 HTML에서 사용하고자 한다면, 각 문자를 HTML Entity로 미리 변환해 주어야 한다.

```
{% autoescape on %}    # autoescape 태그
    {{ content }}
{% endautoescape %}

{{ content|escape }}   # escape 필터
```

- 템플릿 파일에서 URL을 추출하는 문법 – get_absolute_url(), {% url %} 템플릿 태그, URL 패턴명을 이용
get_absolute_url()는 모델 클래스의 메소드로 정의되어 있어야 가능

Python Django

➤ Django 모델 (Model)

- Django에서 Model은 데이터 서비스를 제공하는 Layer이다. (ORM 기법 사용)
- Django의 Model은 각 Django App안에 기본적으로 생성되는 models.py 모듈 안에 정의한다.
- models.py는 테이블을 정의하는 파일
- models.py 모듈 안에 하나 이상의 모델 클래스를 정의할 수 있으며, 하나의 모델 클래스는 데이터베이스에서 하나의 테이블에 해당된다
- Django 모델은 "django.db.models.Model" 의 파생 클래스이며, 모델의 필드는 클래스의 Attribute로 표현되며 테이블의 컬럼에 해당한다.
- 모델 클래스는 필드를 정의하기 위해 인스턴스 변수가 아닌 클래스 변수를 사용(변수가 테이블 컬럼의 내용을 갖는 것이 아니라, 테이블의 컬럼 메타 데이터를 정의하는 것)
- 필드를 정의하는 각각의 클래스 변수는 models.CharField(), models.IntegerField(), models.DateTimeField(), models.TextField() 등의 각 필드 타입에 맞는 Field 클래스 객체를 생성하여 할당한다.
- Field 클래스는 여러 종류가 있는데, 생성자 호출시 필요한 옵션들을 지정할 수 있다.
- 각 Field 클래스마다 반드시 지정해야 주어야 하는 옵션이 있을 수 있(예 : CharField (와 그 서브클래스들)은 필드의 최대 길이를 나타내는 max_length를 항상 지정)
- 마이그레이션(migration) – 테이블 및 필드의 생성, 삭제 변경 등과 같이 데이터베이스에 대한 변경 사항을 알려주는 정보
- 애플리케이션 디렉토리별로 migrations/디렉토리 하위에 마이그레이션 파일들이 존재
- 마이그레이션 정보를 이용해 변경 사항을 실제 데이터베이스에 반영하는 makemigrations 및 migrate 명령 제공

Python Django

➤ Django 모델 (Model)

- migrate : 데이터베이스 스키마를 자동으로 관리하는 명령
- sqlmigrate : 마이그레이션이 실행되는 SQL 확인
- Django가 제공하는 python 셸을 이용한 데이터베이스 API 사용

Python Django

➤ Django 모델 (Model)

- 모든 필드 타입 클래스들은 추상클래스인 "Field" 클래스의 파생클래스들이다.

Field Type	설명
CharField	제한된 문자열 필드 타입. 최대 길이를 max_length 옵션에 지정해야 한다. 문자열의 특별한 용도에 따라 CharField의 파생클래스로서, 이메일 주소를 체크를 하는 EmailField, IP 주소를 체크를 하는 GenericIPAddressField, 콤마로 정수를 분리한 CommaSeparatedIntegerField, 특정 폴더의 파일 패스를 표현하는 FilePathField, URL을 표현하는 URLField 등이 있다.
TextField	대용량 문자열을 갖는 필드
IntegerField	32 비트 정수형 필드. 정수 사이즈에 따라 BigIntegerField, SmallIntegerField 을 사용할 수도 있다.
BooleanField	true/false 필드. Null 을 허용하기 위해서는 NullBooleanField를 사용한다.
DateTimeField	날짜와 시간을 갖는 필드. 날짜만 가질 경우는 DateField, 시간만 가질 경우는 TimeField를 사용한다.
DecimalField	소숫점을 갖는 decimal 필드
BinaryField	바이너리 데이터를 저장하는 필드
FileField	파일 업로드 필드
ImageField	FileField의 파생클래스로서 이미지 파일인지 체크한다.
UUIDField	GUID (UUID)를 저장하는 필드

Python Django

➤ Django 모델 (Model)

- 모델의 필드는 필드 타입에 따라 여러 옵션(혹은 Argument)를 가질 수 있다.
- 모든 필드 타입에 적용 가능한 옵션들 중 자주 사용되는 몇가지를 요약

Field 옵션	설명
null (Field.null)	null=True 이면, Empty 값을 DB에 NULL로 저장한다. DB에서 Null이 허용된다. 예: <code>models.IntegerField(null=True)</code>
blank (Field.blank)	blank=False 이면, 필드가 Required 필드이다. blank=True 이면, Optional 필드이다. 예: <code>models.DateTimeField(blank=True)</code>
primary_key (Field.primary_key)	해당 필드가 Primary Key임을 표시한다. 예: <code>models.CharField(max_length=10, primary_key=True)</code>
unique (Field.unique)	해당 필드가 테이블에서 Unique함을 표시한다. 해당 컬럼에 대해 Unique Index를 생성한다. 예: <code>models.IntegerField(unique=True)</code>
default (Field.default)	필드의 디폴트값을 지정한다. 예: <code>models.CharField(max_length=2, default="WA")</code>
db_column (Field.db_column)	컬럼명은 디폴트로 필드명을 사용하는데, 만약 다르게 쓸 경우 지정한다.

Python Django

➤ URLconf

- `urls.py` : URL과 뷰(함수 또는 메소드)를 매핑
- 프로젝트 전체 URL을 정의하는 프로젝트 URL과 앱마다 정의하는 앱 URL, 2계층으로 나눠서 코딩하는 방식을 권장
- URLconf 모듈을 계층적으로 구성하므로 변경도 쉽고 확장도 용이
- URL 패턴별로 이름을 지정할 수 있고, 패턴 그룹에 대해 이름공간을 지정할 수도 있다
- `reverse()` , `{% url %}` 템플릿 태그 - 소스에 URL을 하드 코딩하지 않아도 필요한 URL을 추출 할 수 있는 기능 제공

Python Django

➤ Admin 사이트

- 테이블의 내용을 열람하고 수정하는 기능을 제공
- 테이블에 들어 있는 내용(콘텐츠)를 편집하는 기능 제공
- admin.py

Python Django

➤ 테스트용 웹 서버 - `runserver`

- 상용 웹 서버에 비해 처리 능력도 떨어지고 보안에도 취약
- Apache 또는 Nginx 상용 웹 서버 사용 권장

Python Django

➤ Django Admin

- 모델 용 관리 인터페이스 생성을 자동화
- 관리자 사이트에 로그인 할 수 있는 사용자 생성

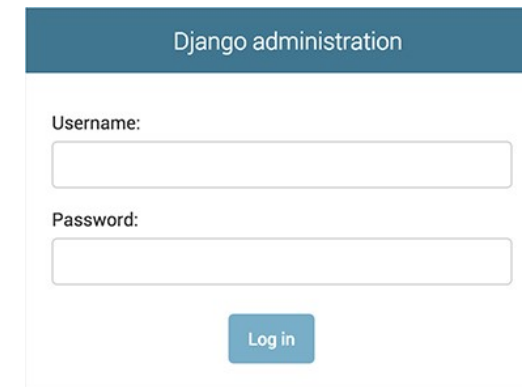
```
$ python manage.py createsuperuser
```

- 사용자 이름을 입력
- 이메일 주소를 입력
- 비밀번호를 입력

```
Username: admin
Email address: admin@example.com
Password: *****
Password (again): *****
Superuser created successfully.
```

- 개발 서버를 시작, `http://127.0.0.1:8000/admin/` 으로 이동

```
$ python manage.py runserver
```

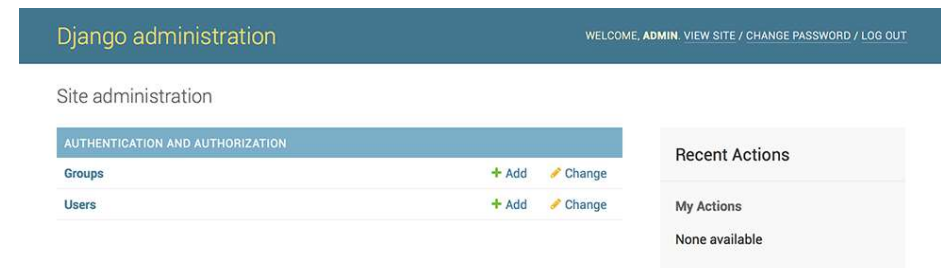


Django administration

Username:

Password:

Log in



Django administration WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION	
Groups	+ Add Change
Users	+ Add Change

Recent Actions

My Actions

None available

그룹과 사용자 관리 및 편집
django.contrib.auth 인증 프레임워크 제공

Python Django

➤ Django Admin

- 관리자에서 설문 조사 앱 모델(Question)개체에 관리자 인터페이스를 제공하기 위해
- 앱 폴더/admin.py에 `admin.site.register(Question)` 추가