Django with Python

장고를 활용한 파이썬 웹프로그래밍

CH1. 장고 개발의 기본 사항
CH2. Bookmark 앱

장예훈

1. 장고 개발의 기본 사항

- 장고(Django) 웹프로그래밍의 가장 큰 장점 : 쉽고, 빠르다

- 대표적인 장고의 특징

1. 일정한 룰에 의해 진행 – MTV 방식

2. 웹프로그래밍에서 공통적으로 필요한 기능들을 미리 만들어 둔 후에 단축 함수, 제너릭 뷰 등으로 제공

3. 풍부한 외부 라이브러리

4. 해외에서 가장 많이 사용하는 웹 프레임 워크 – 레퍼런스가 많음

1. 장고 개발의 기본 사항

- 장고의 개발 방식 = MTV 방식
- MTV (Model-Template-View)방식?
  - Model : 테이블을 정의
  - Template : 사용자가 보게 될 화면의 모습 정의
  - View : 애플리케이션의 제어 흐름 및 처리 로직 정의
- MTV 방식의 장점
  - 각 모듈 간의 독립성 유지
  - 소프트웨어 개발의 중요 원칙인 느슨한 결합(Loose Coupling) 설계의 원칙에 부합

MTV 코딩 순서

- **Model – View & Template 순으로 코딩**

- Model은 독립적으로 개발 할 수 있기 때문에 먼저 코딩

→ View 와 Template은 서로 영향을 미치므로 Model 이후에 함께 코딩

- **프로젝트 뼈대 만들기 – Model 코딩 – URLconf 코딩 – View 코딩 – Template 코딩**

각 단계별 주요 사항

- Setting.py 주요 사항
  - 프로젝트 설정 파일
  - 필수 사항 : 데이터베이스 설정, 템플릿 항목 설정, 정적 파일 항목 설정, 애플리케이션 등록, 타임존 지정

- Models.py 주요 사항
  - 테이블을 정의
  - ORM(Object Relation Mapping) 기법 사용
    : 테이블을 클래스로 매핑, 테이블에 대한 cloud기능을 클래스 객체에 대해 수행하면 알아서 DB에 반영
  - Migration 개념 도입

각 단계별 주요 사항

- **URLconf 주요 사항**
  - URLconf: URL과 View를 매핑해주는 urls.py 파일을 지칭
  - 프로젝트 URL과 앱URL, 2계층으로 나눠서 작업하는 것을 권장

    → 수정 및 확장 용이 & 다른 프로젝트에서 urls.py를 수정 없이 재활용 가능

- **Views.py 주요 사항**
  - View 로직을 코딩하는 가장 중요한 파일
  - 가독성, 유지보수 편리성, 재활용 가능 여부가 아주 중요
  - 함수형 뷰(Function-based View)와 클래스형 뷰(Class-based View)로 구분

- **Templates 주요 사항**
  - 템플릿은 여러 개, 따라서 템플릿 디렉터리가 필요
  - 프로젝트 템플릿, 앱 템플릿으로 구분

Admin 사이트 & 개발용 웹 서버 - runserver

- **Admin 사이트**
  - 테이블의 내용을 열람, 수정하는 기능 제공
  - Contents를 편집하는 기능을 제공

- **개발용 웹 서버 - runserver**
  - 개발 과정에 현재의 웹 프로그램을 실행해 볼 수 있도록 runserver라는 테스트용 웹 서버 제공

Bookmark 앱 개발

- 로직이 간단하므로 웹 프로그래밍을 시작하기에 좋은 예제
- 작업/코딩 순서

작업 순서	관련 명령/파일	필요한 작업 내용
뼈대 만들기	Startproject	Mysite 프로젝트 생성
	Settings.py	프로젝트 설정 항목 변경
	migrate	User/Group 테이블 생성
	createsuperuser	프로젝트 관리자인 슈퍼유저 만듦
	startapp	북마크 앱 생성
	Setting.py	북마크 앱 등록
모델 코딩하기	Models.py	모델(테이블) 정의
	Admin.py	Admin 사이트에 모델 등록
	Makemigrations	모델을 데이터베이스에 반영
	Migrate	
URLconf 코딩하기	Urls.py	URL 정의
View 코딩하기	Views.py	View 로직 작성
Template	Templates 디렉터리	Template 파일 작성

프로젝트 생성

$ django-admin.py startproject mysite

```
(django_project) C:\workspace\django_project>django-admin.py startproject ch2
```

```
(django_project) C:\workspace\django_project>dir
 C 드라이브의 볼륨에는 이름이 없습니다.
 볼륨 일련 번호: C64C-D2CB

 C:\workspace\django_project 디렉터리

2017-05-11  오후 02:01    <DIR>          .
2017-05-11  오후 02:01    <DIR>          ..
2017-05-11  오후 02:42    <DIR>          ch2
               0개 파일                   0 바이트
               3개 디렉터리  171,716,984,832 바이트 남음
```

프로젝트 설정 파일 변경

- Database, INSTALLED_APPS, TIME_ZONE, TEMPLATES, MEDIA, STATIC, LANGUAGE_CODE 수정

$ C:₩workspace₩django_project₩ch2₩mysite

〈mysite/setting.py〉

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

- 장고의 기본 데이터베이스가 SQLite3인 것을 확인

프로젝트 설정 파일 변경

- 코드 추가 및 수정

```
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

- 코드 추가 및 수정

```
STATIC_URL = '/static/'

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

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

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

```
# LANGUAGE_CODE = 'en-us'
LANGUAGE_CODE = 'ko-kr'
```

기본 테이블 생성

- DB에 변경사항을 반영해주는 명령

$ python manage.py

```
(django_project) C:\workspace\django_project\ch2>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
```

Super User 생성

- Admin 사이트에 로그인하기 위한 관리자(Superuser) 생성

$ python manage.py createsuperuser

```
(django_project) C:\workspace\django_project\ch2>python manage.py createsuperuser
Username (leave blank to use 'jangy'): yehoon
Email address: jangyh0420@naver.com
Password:
Password (again):
This password is too short. It must contain at least 8 characters.
Password:
Password (again):
Superuser created successfully.
```

애플리케이션 생성

- Bookmark 앱을 만드는 명령

$ python manage.py startapp bookmark

```
(django_project) C:\workspace\django_project\ch2>python manage.py startapp bookmark
```

애플리케이션 등록

- 프로젝트에 포함되는 App들은 모두 settings.py에 지정되어야 함

〈mysite/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',
]
```

모델 – 테이블 정의

<bookmark/models.py>

```python
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
```

- __str__ 함수는 객체를 문자열로 표현 할 때 사용하는 함수
  이 함수를 정의하지 않으면 테이블 명이 제대로 표현되지 않음

모델 – Admin 사이트에 테이블 반영

- models.py에서 정의한 테이블도 Admin사이트에 보이도록 등록

〈bookmark/admin.py〉

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

# Register your models here.

class BookmarkAdmin(admin.ModelAdmin):
    list_dispaly = ('title', 'url')

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

- BookmarkAdmin 클래스: Admin 사이트에서 Bookmark 클래스가 어떻게 보여질지 정의
- admin.site.register() 함수 : Bookmark 클래스, BookmarkAdmin클래스 등록

모델 – DB 변경 사항 반영

- 테이블 신규 생성, 테이블 정의 변경 등의 작업을 해줬으므로 DB에 반영해줘야 함

$ python manage.py makemigrations

$ python manage.py migrate

```
(django_project) C:\workspace\django_project\ch2>python manage.py makemigrations
Migrations for 'bookmark':
  bookmark\migrations\0001_initial.py:
    - Create model Bookmark

(django_project) C:\workspace\django_project\ch2>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, bookmark, contenttypes, sessions
Running migrations:
  Applying bookmark.0001_initial... OK
```

모델 – 테이블 모습 확인하기

- Admin 사이트에 접속하여 테이블 모습 확인

$ python manage.py runserver

```
(django_project) C:\workspace\django_project\ch2>python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).
May 11, 2017 - 14:15:03
Django version 1.10.5, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

- 웹 브라우저를 열고 주소창에 'localhost:8000/admin' 입력하여 확인

URLconf 코딩

<mysite/urls.py>

```
from django.conf.urls import url
from django.contrib import admin

from bookmark.views import BookmarkLV, BookmarkDV

urlpatterns = [
    url(r'^admin/', admin.site.urls),

    #Class-based views for Bookmark app
    url(r'^bookmark/$', BookmarkLV.as_view(), name='index'),
    url(r'^bookmark/(?P<pk>\d+)/$', BookmarkDV.as_view(), name='detail'),
]
```

View 코딩

〈bookmark/views.py〉

```python
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
```

Template 코딩

- Bookmark_list.html

〈bookmark/templates/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>
```

Template 코딩

- Bookmark_detail.html

〈bookmark/templates/bookmark/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>
```

THANK
Y O U