Django 웹 프레임워크

장고를 사용하여 웹 애플리케이션을 만드는 과정

일반적인 특징

| MVC 패턴 기반 MVT |

장고는 MVCModel-View-Controller를 기반으로 한 프레임워크입니다. 하지만 장고에서는 View를 Template, Controller를 View라고 함

| 객체 관계 매핑 |

장고의 객체 관계 매핑ORM, Object-Relational Mapping은 데이터베이스 시스템과 데이터 모델 클래스를 연 결시키는 다리와 같은 역할

자동으로 구성되는 관리자 화면 |

장고는 웹 서버의 콘텐츠, 즉 데이터베이스에 대한 관리 기능을 위하여 프로젝트를 시작하는 시점 에 기본 기능으로 관리자 화면을 제공합니다

| 우아한 URL 설계 |

웹 프로그래밍에서 URL 디자인은 필수인데, 장고에서는 유연하면서도 강력한 기능을 제공. 장고에서는 우아한Elegant URL 방식을 채택하여 URL을 직관적이고 쉽게 표현

| 자체 템플릿 시스템 |

장고는 내부적으로 확장이 가능하고 디자인이 쉬운 강력한 템플릿 시스템을 갖고 있음. 이를 통해 화면 디자인과 로직에 대한 코딩을 분리하여 독립적으로 개발 진행

일반적인 특징

| 캐시 시스템 |

동적인 페이지를 만들기 위해서 데이터베이스 쿼리를 수행하고 템플릿을 해석하며, 관련 로직을 실행해서 페이지를 생성하는 일은 서버에 엄청난 부하를 주는 작업

다국어 지원 |

장고는 동일한 소스코드를 다른 나라에서도 사용할 수 있도록 텍스트의 번역, 날짜/시간/숫자의 포맷, 타임존의 지정 등과 같은 다국어 환경을 제공

풍부한 개발 환경 |

장고는 개발에 도움이 될 수 있는 여러 가지 기능을 제공

소스 변경사항 자동 반영 |

장고에서는 *.py 파일의 변경 여부를 감시하고 있다가 변경이 되면 실행 파일에 변경 내역을 바로 반영

장고 프로그램 설치

아래 명령으로 파이썬 설치 여부를 알 수 있으며, 파이썬이 설치되었다면 버전이 표시됩니다

```
C:₩Users₩wshkim>python -V
```

만일 설치되지 않았다면 www.python.org에서 파이썬 프로그램을 다운로드해서 설치하면 됩니다.

장고 프로그램 설치

윈도우에서 장고 설치

파이썬 3.x 버전을 설치하면 pip 프로그램도 같이 설치.
pip Python Install Package 프로그램은 파이썬의 오픈소스 저장소인 PyPI Python Package Index 에 있는 SW 패키지를 설치하고 관리해주는 명령

```
C:₩Users₩wshkim>pip install Django
```

장고 프로그램 설치

장고 프로그램 설치 확인

장고는 파이썬 환경에서 동작하는 패키지이므로, 장고가 정상적으로 설치되었는지 확인하기 위해서 아래와 같이 명령을 입력

```
C:₩Users₩shkim>python -m django --version
2.0.5
```

장고에서의 애플리케이션 개발 방식

웹 사이트의 전체 프로그램 또는 모듈화된 단위 프로그램을 애플리케이션이라고 합니다. 즉, 프로그램으로 코딩할 대상을 애플리케이션이라고 부른다

사이트에 대한 전체 프로그램을 프로젝트Project라 하고, 모듈화된 단위 프로그램을 애플리케이션Application이라 부른다.

장고에서의 애플리케이션 개발 방식

개발 시 일반적으로 언급되는 MVC$_{Model\text{-}View\text{-}Controller}$ 패턴이란 데이터$_{Model}$, 사용자 인터페이스$_{View}$, 데이터를 처리하는 로직$_{Controller}$을 구분해서 한 요소가 다른 요소들에 영향을 주 지 않도록 설계하는 방식

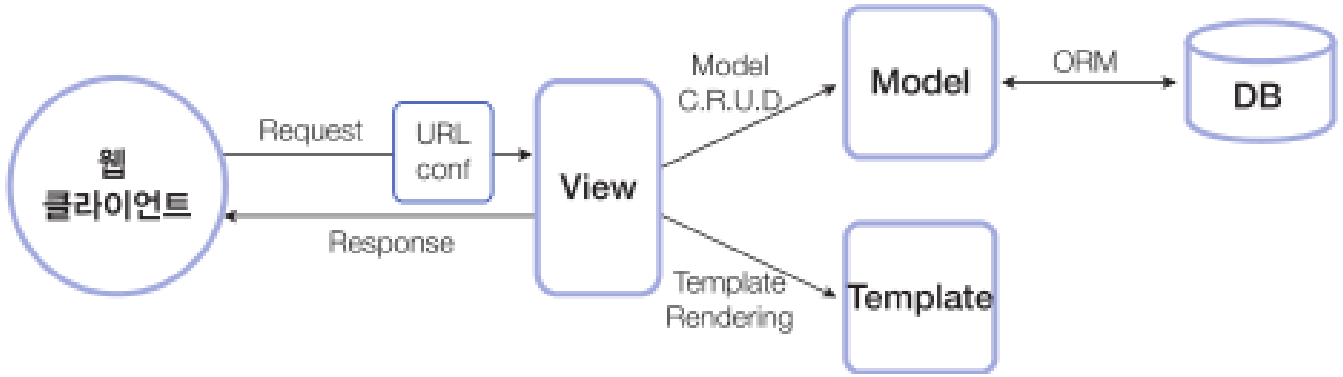


그림 **3-5** 장고의 MVT 패턴

장고에서의 애플리케이션 개발 방식

Model – 데이터베이스 정의

모델이란 사용될 데이터에 대한 정의를 담고 있는 장고의 클래스입니다. 장고는 ORM 기법을 사용하여 애플리케이션에서 사용할 데이터베이스를 클래스로 매핑해서 코딩할 수있음

```
from django.db import models

class Person(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
```

```
CREATE TABLE myapp_person (
    "id" serial NOT NULL PRIMARY KEY,
    "first_name" varchar(30) NOT NULL,
    "last_name" varchar(30) NOT NULL
);
```

장고는 테이블 및 컬럼을 자동으로 생성하기 위해 필요한 많은 규칙을 갖고 있습니다. 위의 예제는 그중에서 다음과 같은 규칙이 적용.

장고에서의 애플리케이션 개발 방식

URLconf – URL 정의

파이썬의 URL 정의 방식은 전통적인 자바나 PHP 계열의 URL보다 직관적이고 이해하기가 쉬움.
그래서 이런 방식을 우아한(Elegant) URL이라고 부르는 것.
URL을 정의하기 위해서는 다음 예제처럼 **urls.py** 파일에 URL과 처리 함수(뷰(View)라고 부름)를 매핑하는 파이썬 코드를 작성하면 됨

예제 3-1 URLconf 예시

```
from django.urls import path

from . import views

urlpatterns = [
    path('articles/2003/', views.special_case_2003),
    path('articles/<int:year>/', views.year_archive),
    path('articles/<int:year>/<int:month>/', views.month_archive),
    path('articles/<int:year>/<int:month>/<slug:slug>/', views.article_detail),
]
```

장고에서의 애플리케이션 개발 방식

View – 로직 정의

뷰는 웹 요청을 받아서 데이터베이스 접속 등 해당 애플리케이션의 로직에 맞는 처리 를 하고, 그 결과 데이터를 HTML로 변환하기 위하여 템플릿 처리를 한 후에, 최종 HTML로 된 응답 데이터를 웹 클라이언트로 반환하는 역할

```
from django.http import HttpResponse
import datetime

def current_datetime(request):
    now = datetime.datetime.now()
    html = "<html><body>It is now %s.</body></html>" % now
    return HttpResponse(html)
```

장고에서의 애플리케이션 개발 방식

Template – 화면 UI 정의

개발자가 응답에 사용할 *.html 파일을 작성하면, 장고는 이를 해석해서 최종 HTML 텍스트 응답을 생성하고, 이를 클라 이언트에게 보내줍니다. 클라이언트(보통 웹 브라우저)는 응답으로 받은 HTML 텍스트를 해석해 서 우리가 보는 웹 브라우저 화면에 UI를 보여주는 것

템플릿 파일은 *.**html** 확장자를 가지며, 장고의 템플릿 시스템 문법에 맞게 작성

장고에서 템플릿 파일을 찾을 때는 TEMPLATES 및 INSTALLED_APPS에서 지정된 앱의 디렉 토리를 검색.
이 항목들은 프로젝트 설정 파일인 **settings.py** 파일에 정의되어 있음

장고에서의 애플리케이션 개발 방식

MVT 코딩 순서

모델, 뷰, 템플릿 셋 중에서 무엇을 먼저 코딩해야 하는지에 대해 정해진 순서는 없음.
MVT 방식에 따르면 화면 설계는 뷰와 템플릿 코딩으로 연결되고, 테이블 설계는 모델 코딩에 반영. 그렇기 때문에 독립적으로 개발할 수 있는 모델을 먼저 코딩하고, 뷰와 템플릿은 서로 영향을 미치므로 모델 이후에 같이 코딩하는 것이 일반적

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)
- 템플릿 코딩하기 : 화면 UI 개발(templates/ 디렉토리 하위의 *.html 파일들)
- 뷰 코딩하기 : 애플리케이션 로직 개발(views.py 파일)

애플리케이션 설계하기

- index.html : 최근에 실시하고 있는 질문의 리스트를 보여줌.
- detail.html : 하나의 질문에 대해 투표할 수 있도록 답변 항목을 폼으로 보여줌
- results.html : 질문에 따른 투표 결과를 보여줌.

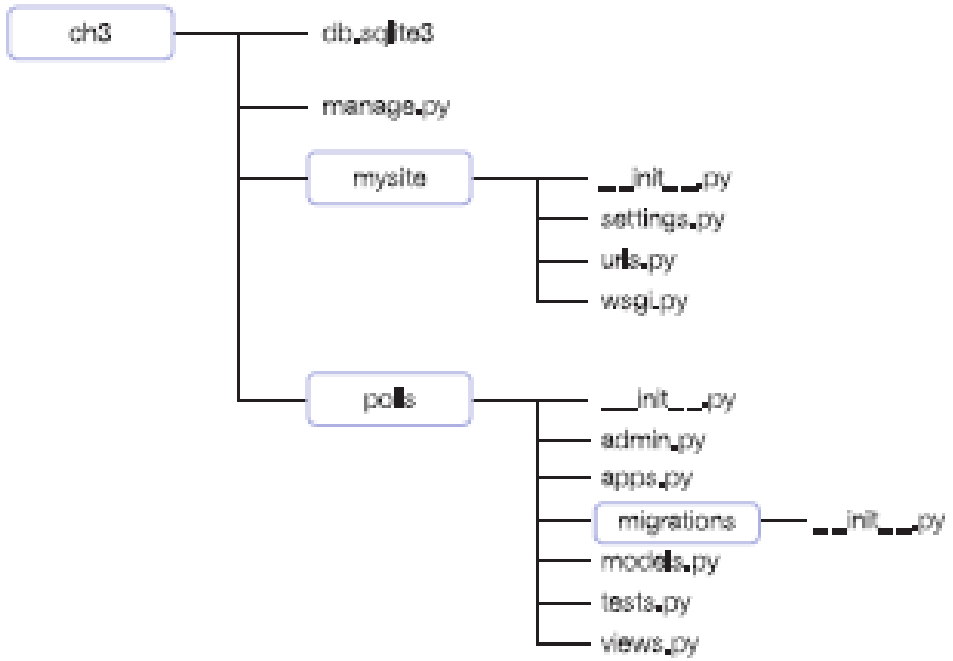
표 **3-2** Question 테이블 설계

컬럼명	타입	제약 조건	설명
id	integer	NotNull, PK, AutoIncrement	Primary Key
question_text	varchar(200)	NotNull	질문 문장
pub_date	datetime	NotNull	질문 생성 시각

표 **3-3** Choice 테이블 설계

컬럼명	타입	제약 조건	설명
id	integer	NotNull, PK, AutoIncrement	Primary Key
choice_text	varchar(200)	NotNull	답변 항목 문구
votes	integer	NotNull	투표 카운트
question	integer	NotNull, FK (Question.id), Index	Foreign Key

프로젝트 뼈대 만들기



프로젝트 뼈대의 최종 디렉토리 모습

항목명	설명
ch3 디렉토리	프로젝트 관련 디렉토리 및 파일을 모아주는 최상위 루트 디렉토리입니다. 보통 settings.py 파일의 BASE_DIR 항목으로 지정됩니다.
db.sqlite3	SQLite3 데이터베이스 파일입니다. 테이블이 들어있습니다.
manage.py	장고의 명령어를 처리하는 파일입니다.
mysite 디렉토리	프로젝트명으로 만들어진 디렉토리입니다. 프로젝트 관련 파일들이 들어있습니다.
__init__.py	디렉토리에 이 파일이 있으면 파이썬 패키지로 인식합니다.
settings.py	프로젝트 설정 파일입니다.
urls.py	프로젝트 레벨의 URL 패턴을 정의하는 최상위 URLconf입니다. 보통은 애플리케이션 디렉토리마다 하위 urls.py 파일이 또 있습니다.
wsgi.py	Apache와 같은 웹 서버와 WSGI 규격으로 연동하기 위한 파일입니다.
polls 디렉토리	애플리케이션명으로 만들어진 애플리케이션 디렉토리입니다. 해당 애플리케이션 관련 파일들이 들어있습니다.
__init__.py	디렉토리에 이 파일이 있으면 파이썬 패키지로 인식합니다.
admin.py	Admin 사이트에 모델 클래스를 등록해주는 파일입니다.
apps.py	애플리케이션의 설정 클래스를 정의하는 파일입니다.
migrations 디렉토리	데이터베이스 변경사항을 관리하기 위한 디렉토리입니다. 데이터베이스에 추가, 삭제, 변경 등이 발생하면 변경 내역을 기록한 파일들이 위치합니다.
models.py	데이터베이스 모델 클래스를 정의하는 파일입니다.
tests.py	단위 테스트용 파일입니다. 이 책에서는 사용하지 않습니다.
views.py	뷰 함수를 정의하는 파일입니다. 함수형 뷰 및 클래스형 뷰 모두 이 파일에 정의합니다.
templates 디렉토리	프로젝트를 진행하면서 추가됩니다. 템플릿 파일들이 들어 있습니다. 보통은 프로젝트 레벨과 애플리케이션 레벨의 템플릿으로 구분하여 ch3/templates 및 ch3/polls/templates 위치에 생성됩니다.
static 디렉토리	프로젝트를 진행하면서 추가됩니다. CSS, Image, Javascript 파일들이 들어있습니다. 보통은 프로젝트 레벨과 애플리케이션 레벨로 구분하여 ch3/static 및 ch3/polls/static 위치에 생성됩니다.
logs 디렉토리	프로젝트를 진행하면서 추가됩니다. 로그 파일들이 들어있습니다. 로그 파일의 위치는 settings.py 파일의 LOGGING 항목으로 지정합니다.

프로젝트 뼈대 만들기

프로젝트 생성

```
C:₩RedBook>django-admin startproject mysite

C:₩RedBook>dir
 C 드라이브의 볼륨: Windows
 볼륨 일련 번호: 2AA7-7AEC

 C:₩RedBook 디렉터리

2018-05-05  오후 05:53    <DIR>          .
2018-05-05  오후 05:53    <DIR>          ..
2018-04-07  오후 10:05    <DIR>          ch2
2018-04-15  오후 07:45    <DIR>          ch2-test-server
2018-03-18  오후 01:53    <DIR>          del-proxy-server
2018-05-05  오후 05:53    <DIR>          mysite
               0개 파일                   0 바이트
               6개 디렉터리  210,570,416,128 바이트 남음

C:₩RedBook>dir mysite
 C 드라이브의 볼륨: Windows
 볼륨 일련 번호: 2AA7-7AEC

 C:₩RedBook₩mysite 디렉터리

2018-05-05  오후 05:53    <DIR>          .
2018-05-05  오후 05:53    <DIR>          ..
2018-05-05  오후 05:53               553 manage.py
2018-05-05  오후 05:53    <DIR>          mysite
               1개 파일                 553 바이트
               3개 디렉터리  210,570,416,128 바이트 남음

C:₩RedBook>dir mysite₩mysite
 C 드라이브의 볼륨: Windows
 볼륨 일련 번호: 2AA7-7AEC

 C:₩RedBook₩mysite₩mysite 디렉터리

2018-05-05  오후 05:53    <DIR>          .
2018-05-05  오후 05:53    <DIR>          ..
2018-05-05  오후 05:53             3,208 settings.py
2018-05-05  오후 05:53               769 urls.py
2018-05-05  오후 05:53               405 wsgi.py
2018-05-05  오후 05:53                 0 __init__.py
               4개 파일               4,382 바이트
               2개 디렉터리  210,570,350,592 바이트 남음

C:₩RedBook>
```

그림 **3-8** startproject 명령 실행 후 디렉토리 모습

프로젝트 뼈대 만들기

애플리케이션 생성

```
C:\Users\shkim>cd C:\RedBook\ch3
C:\RedBook\ch3>python manage.py startapp polls
```

다음은 프로젝트 루트 디렉토리 ch3으로 이동해서 **polls**라는 애플리케이션을 만드는 명령을 실행.
polls는 원하는 애플리케이션 명칭을 입력.

```
C:\RedBook>cd ch3

C:\RedBook\ch3>python manage.py startapp polls

C:\RedBook\ch3>dir
 C 드라이브의 볼륨: Windows
 볼륨 일련 번호: 2AA7-7AEC

 C:\RedBook\ch3 디렉터리

2018-05-05  오후 06:09    <DIR>          .
2018-05-05  오후 06:09    <DIR>          ..
2018-05-05  오후 05:53               553 manage.py
2018-05-05  오후 06:09    <DIR>          mysite
2018-05-05  오후 06:09    <DIR>          polls
               1개 파일                 553 바이트
               4개 디렉터리  210,568,044,544 바이트 남음

C:\RedBook\ch3>dir polls
 C 드라이브의 볼륨: Windows
 볼륨 일련 번호: 2AA7-7AEC

 C:\RedBook\ch3\polls 디렉터리

2018-05-05  오후 06:09    <DIR>          .
2018-05-05  오후 06:09    <DIR>          ..
2018-05-05  오후 06:09                66 admin.py
2018-05-05  오후 06:09                90 apps.py
2018-05-05  오후 06:09    <DIR>          migrations
2018-05-05  오후 06:09                60 models.py
2018-05-05  오후 06:09                63 tests.py
2018-05-05  오후 06:09                66 views.py
2018-05-05  오후 06:09                 0 __init__.py
               6개 파일                 345 바이트
               3개 디렉터리  210,568,040,448 바이트 남음

C:\RedBook\ch3>
```

그림 3-9 startapp 명령 실행 후 디렉토리 모습

프로젝트 뼈대 만들기

프로젝트 설정 파일 변경

```
C:\Users\shkim>cd C:\RedBook\ch3\mysite

C:\RedBook\ch3\mysite>notepad settings.py
```

첫 번째로 ALLOWED_HOSTS 항목을 적절하게 지정

```
ALLOWED_HOSTS = [ '192.168.56.101', 'localhost', '127.0.0.1' ]
```

두 번째로 프로젝트에 포함되는 애플리케이션들은 모두 설정 파일에 등록
따라서 우리가 개발하고 있는 polls 애플리케이션도 등록

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'polls.apps.PollsConfig',          # 추가
]
```

번째로 프로젝트에 사용할 데이터베이스 엔진입니다

```
# TIME_ZONE = 'UTC'

TIME_ZONE = 'Asia/Seoul'
```

네 번째는 타임존 지정. 최초에는 세계표준시 (UTC)로 되어 있는데, 한국 시간으로 변경

프로젝트 뼈대 만들기

기본 테이블 생성

```
C:\Users\shkim>cd C:\RedBook\ch3
C:\RedBook\ch3>python manage.py migrate
```

다음은 기본 테이블 생성을 위하여 아래 명령을 실행.
migrate 명령은 데이터베이스에 변경사항이 있을 때
이를 반영해주는 명령

그림 **3-10** migrate 명령 실행 후 디렉토리 모습

```
C:\RedBook\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 auth.0009_alter_user_last_name_max_length... OK
  Applying sessions.0001_initial... OK

C:\RedBook\ch3>dir
 C 드라이브의 볼륨: Windows
 볼륨 일련 번호: 2AA7-7AEC

 C:\RedBook\ch3 디렉터리

2018-05-05  오후 06:48    <DIR>          .
2018-05-05  오후 06:48    <DIR>          ..
2018-05-05  오후 06:48           131,072 db.sqlite3
2018-05-05  오후 05:53               553 manage.py
2018-05-05  오후 06:09    <DIR>          mysite
2018-05-05  오후 06:09    <DIR>          polls
               2개 파일             131,625 바이트
               4개 디렉터리  210,560,782,336 바이트 남음

C:\RedBook\ch3>
```

애플리케이션 개발하기 - Model 코딩

```
>notepad models.py                          // 데이블을 정의함
>notepad admins.py                          // 정의된 데이블이 Admin 화면에 보이게 함
>python manage.py makemigrations            // 데이터베이스에 변경이 필요한 사항을 추출함
>python manage.py migrate                   // 데이터베이스에 변경사항을 반영함
>python manage.py runserver                 // 현재까지 작업을 개발용 웹 서버로 확인함
```

애플리케이션 개발하기 - Model 코딩

테이블 정의

예제 3-3 Question, Choice 테이블 정의 — 소스제공: ch3₩polls₩models.py

```
C:₩Users₩shkim>cd C:₩RedBook₩ch3₩polls
C:₩RedBook₩ch3₩polls>notepad models.py

from django.db import models

class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')

    def __str__(self):
        return self.question_text

class Choice(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)

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

애플리케이션 개발하기 - Model 코딩

Admin 사이트에 테이블 반영

예제 3-4 Admin 사이트에 반영 — 소스제공: ch3₩polls₩admin.py

```
C:₩Users₩shkim>cd C:₩RedBook₩ch3₩polls
C:₩RedBook₩ch3₩polls>notepad admin.py

from django.contrib import admin
from polls.models import Question, Choice

admin.site.register(Question)
admin.site.register(Choice)
```

애플리케이션 개발하기 - Model 코딩

데이터베이스 변경사항 반영

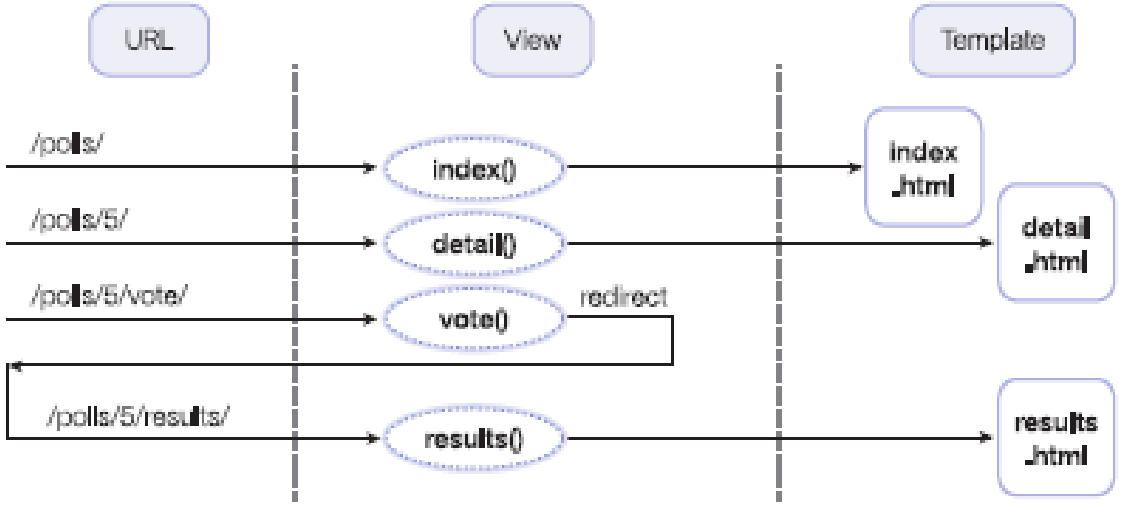
```
C:\Users\shkim>cd C:\RedBook\ch3
C:\RedBook\ch3>python manage.py makemirations
C:\RedBook\ch3>python manage.py migrate
```

마이그레이션(migrations)이란 용어는 장고 1.7 버전부터 사용된 개념으로, 테이블 및 필드의 생성, 삭제, 변경 등과 같이 데이터베이스에 대한 변경사항을 알려주는 정보

```
C:\RedBook\ch3>python manage.py makemigrations
Migrations for 'polls':
  polls\migrations\0001_initial.py
    - Create model Choice
    - Create model Question
    - Add field question to choice

C:\RedBook\ch3>dir polls\migrations
 C 드라이브의 볼륨: Windows
 볼륨 일련 번호: 2AA7-7AEC

 C:\RedBook\ch3\polls\migrations 디렉터리

2018-05-05  오후 09:50    <DIR>          .
2018-05-05  오후 09:50    <DIR>          ..
2018-05-05  오후 09:50             1,201 0001_initial.py
2018-05-05  오후 06:09                 0 __init__.py
2018-05-05  오후 09:50    <DIR>          __pycache__
               2개 파일               1,201 바이트
               3개 디렉터리  210,540,396,544 바이트 남음

C:\RedBook\ch3>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, polls, sessions
Running migrations:
  Applying polls.0001_initial... OK

C:\RedBook\ch3>
```

makemigrations/migrate 명령 실행 시 로그

애플리케이션 개발하기 - View 및 Template 코딩



URLconf 설계 – URL과 뷰 매핑

URL 패턴	뷰 이름	뷰가 처리하는 내용
/polls/	index()	index.html 템플릿을 보여줍니다.
/polls/5/	detail()	detail.html 템플릿을 보여줍니다.
/polls/5/vote/	vote()	detail.html에 있는 폼을 POST 방식으로 처리합니다.
/polls/5/results/	results()	results.html 템플릿을 보여줍니다.
/admin/	(장고 기능)	Admin 사이트를 보여줍니다(장고에서 기본으로 제공함).

※ URL 패턴에서 **5**는 예시로, 질문번호가 채워지는 자리

```
urls.py 작성                    // URLconf 내용을 코딩
views.index() 함수 작성         // index.html 템플릿도 같이 작성
views.detail() 함수 작성        // detail.html 템플릿도 같이 작성
views.vote() 함수 작성          // 리다이렉션 처리 들어있음
views.results() 함수 작성       // results.html 템플릿도 같이 작성
```

애플리케이션 개발하기 - View 및 Template 코딩

URLconf 코딩

예제 3-5 urls.py 파일 코딩

```
C:₩Users₩shkim>cd C:₩RedBook₩ch3₩mysite
C:₩RedBook₩ch3₩mysite>notepad urls.py

from django.contrib import admin
from django.urls import path
from polls import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('polls/', views.index, name='index'),
    path('polls/<int:question_id>/', views.detail, name='detail'),
    path('polls/<int:question_id>/results/', views.results, name='results'),
    path('polls/<int:question_id>/vote/', views.vote, name='vote'),
]
```

❶ ❷ ❸ ❹

애플리케이션 개발하기 - View 및 Template 코딩

뷰 함수 index() 및 템플릿 작성

index.html

What is your hobby?
Who do you like best?
Where do you live?

화면 UI 설계 - index.html

예제 3-8 템플릿 파일 입력 — 소스제공: ch3₩polls₩templates₩polls₩index.html

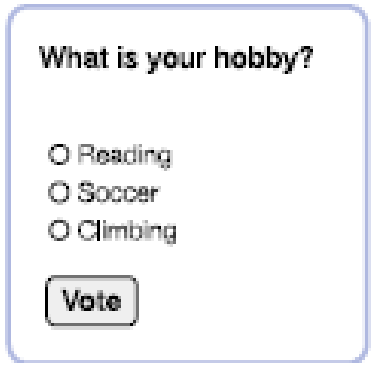
```
C:₩Users₩shkim>cd C:₩RedBook₩ch3₩polls
C:₩RedBook₩ch3₩polls>mkdir templates
C:₩RedBook₩ch3₩polls>mkdir templates₩polls
```

```
C:₩RedBook₩ch3₩polls>cd templates₩polls
C:₩RedBook₩ch3₩polls₩templates₩polls>notepad index.html

{% if latest_question_list %}                                          ①
    <ul>
    {% for question in latest_question_list %}
        <li><a href="/polls/{{ question.id }}/">{{ question.question_text }}</a></li>   ②
    {% endfor %}
    </ul>
{% else %}
    <p>No polls are available.</p>                                     ③
{% endif %}
```

애플리케이션 개발하기 - View 및 Template 코딩

뷰 함수 detail() 및 폼 템플릿 작성

detail.html



화면 UI 설계 - detail.html

예제 3-10 〈form〉 태그 추가 — 소스제공: ch3₩polls₩templates₩polls₩detail.html

```
C:₩Users₩shkim>cd C:₩RedBook₩ch3₩polls₩templates₩polls
C:₩RedBook₩ch3₩polls₩templates₩polls>notepad detail.html

<h1>{{ question.question_text }}</h1>  ---------- ❶

{% if error_message %}<p><strong>{{ error_message }}</strong></p>{% endif %}  ---------- ❷

<form action="{% url 'polls:vote' question.id %}" method="post">  ---------- ❸
{% csrf_token %}  ---------- ❹
{% for choice in question.choice_set.all %}  ---------- ❺
    <input type="radio" name="choice" id="choice{{ forloop.counter }}" value="{{  ---- ❻
choice.id }}" />
    <label for="choice{{ forloop.counter }}">{{ choice.choice_text }}</label><br />  ---- ❼
{% endfor %}
<input type="submit" value="Vote" />  ---------- ❽
</form>
```

애플리케이션 개발하기 - View 및 Template 코딩

뷰 함수 vote() 및 리다이렉션 작성

예제 3-12 polls₩views.py – vote( ) 함수 작성

```
C:₩Users₩shkim>cd C:₩RedBook₩ch3₩polls
C:₩RedBook₩ch3₩polls>notepad views.py

from django.shortcuts import get_object_or_404, render
from django.http import HttpResponseRedirect, HttpResponse  ①
from django.core.urlresolvers import reverse  ②
from polls.models import Choice, Question  ③
# ... index() ··· 변경사항 없음
# ... detail() ··· 변경사항 없음

def vote(request, question_id):  ④
    question = get_object_or_404(Question, pk=question_id)  ⑤
    try:
        selected_choice = question.choice_set.get(pk=request.POST['choice'])
    except (KeyError, Choice.DoesNotExist):  ⑥
        # 설문 투표 폼을 다시 보여준다
        return render(request, 'polls/detail.html', {  ⑦
            'question': question,
            'error_message': "You didn't select a choice.",
        })
    else:  ⑧
        selected_choice.votes += 1  ⑨
        selected_choice.save()  ⑩
        # POST 데이터를 정상적으로 처리하였으면,
        # 항상 HttpResponseRedirect를 반환하여 리다이렉션 처리함
        return HttpResponseRedirect(reverse('polls:results', args=(question.id,)))  ⑪ ⑫
```

애플리케이션 개발하기 - View 및 Template 코딩

뷰 함수 results() 및 템플릿 작성

예제 3-12 polls₩views.py – vote( ) 함수 작성

```
C:₩Users₩shkim>cd C:₩RedBook₩ch3₩polls
C:₩RedBook₩ch3₩polls>notepad views.py

from django.shortcuts import get_object_or_404, render
from django.http import HttpResponseRedirect, HttpResponse ---------- 1
from django.core.urlresolvers import reverse ---------- 2
from polls.models import Choice, Question
# ... index() … 변경사항 없음 ---------- 3
# ... detail() … 변경사항 없음

def vote(request, question_id): ---------- 4
    question = get_object_or_404(Question, pk=question_id)
    try: ---------- 5
        selected_choice = question.choice_set.get(pk=request.POST['choice'])
    except (KeyError, Choice.DoesNotExist): ---------- 6
        # 설문 투표 폼을 다시 보여준다
        return render(request, 'polls/detail.html', { ---------- 7
            'question': question,
            'error_message': "You didn't select a choice.",
        })
    else: ---------- 8
        selected_choice.votes += 1 ---------- 9
        selected_choice.save() ---------- 10
        # POST 데이터를 정상적으로 처리하였으면,
        # 항상 HttpResponseRedirect를 반환하여 리다이렉션 처리함
        return HttpResponseRedirect(reverse('polls:results', args=(question.id,))) ---------- 11, 12
```

뷰 함수 results () 및 템플릿 작성

3.7.5 뷰 함수 results() 및 템플릿 작성

results() 뷰 함수의 호출과 연계된 URL은 votes() 뷰 함수의 리다이렉트 결과로 받습니다. 즉, 폼 데이터를 처리한 후에 그 결과를 보여주는 페이지로 리다이렉트시켜주기 위해 votes() 뷰 함수에서 다음 라인을 실행했습니다. 리다이렉트할 타깃 URL은 /polls/3/result/와 유사한 모습일 것입니다.

```
return HttpResponseRedirect(reverse('polls:results', args=(question.id,)))
```

리다이렉트하라는 응답을 받은 웹 브라우저는 리다이렉트 URL로 다시 요청을 보냅니다. 또한, urls.py에는 이미 다음과 같은 라인을 작성한 바 있습니다.

```
path('polls/<int:question_id>/results/', views.results, name='results'),
```

예제 3-13 polls₩views.py – results() 함수 작성

```
C:₩Users₩shkim>cd C:₩RedBook₩ch3₩polls
C:₩RedBook₩ch3₩polls>notepad views.py

from django.shortcuts import get_object_or_404, render
from django.http import HttpResponseRedirect, HttpResponse
from django.core.urlresolvers import reverse
from polls.models import Choice, Question
# ... index() … 변경사항 없음
# ... detail() … 변경사항 없음
# ... vote() … 변경사항 없음

def results(request, question_id):
    question = get_object_or_404(Question, pk=question_id)
    return render(request, 'polls/results.html', {'question': question})
```