03장 Django 웹 프레임워크

3.1 일반적인 특징

■ 장고는 파이썬 웹 프레임워크이며 주요 기능별 특징은 아래와 같다.

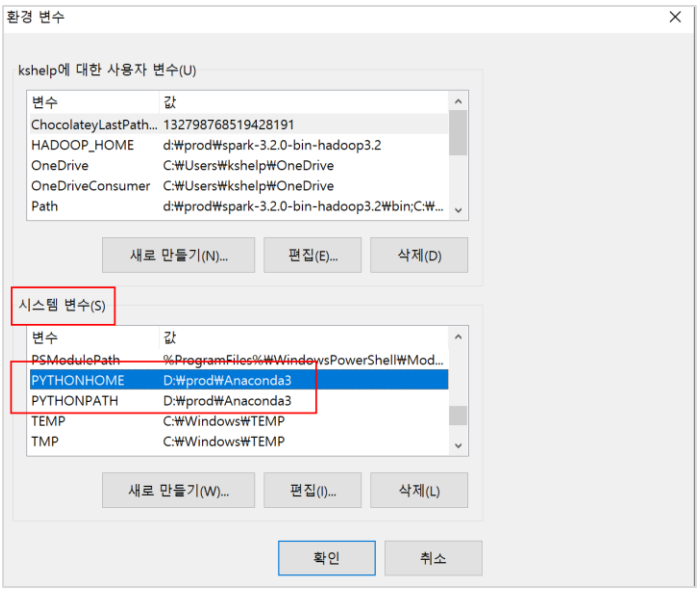
- MVC 패턴 기반 MVT(Model View Template)
  - View를 Template, Controller를 View라고 부른다.
- 객체 관계 매핑
  - 데이터베이스 시스템과 모델이라는 파이썬 클래스를 연결시키는 다리와 같은 역할을 한다.
- 자동으로 구성되는 관리자 화면
  - 웹 서버의 콘텐츠, 즉 데이터베이스에 대한 관리 기능을 위하여 프로젝트를 시작하는 시점에 기본 기능으로 관리자 화면을 제공한다.
- 우아한 URL 설계
  - 정규표현식을 사용하여 복잡한 URL도 표현할 수 있으며, 각 URL 형태를 파이썬 함수에 1:1로 연결하도록 되어 있어 개발이 편리하며 이해하기도 쉽다.
- 자체 템플릿 시스템
- 캐시 시스템
- 다국어 지원
- 풍부한 개발 환경
  - 테스트용 웹 서버를 포함하고 있어서 개발 과정에서 아파치 등의 웹 서버가 없어도 테스트를 진행할 수 있다.
- 소스 변경사항 자동 반영
  - 개발 과정에서 장고는 *.py 파일의 변경 여부를 감시하고 있다가 변경이 되면 실행 파일에 변경 내역을 바로 반영해 준다.

3.2 장고 프로그램 설치

3.2.1 윈도우에서 장고 설치

■ 시스템 환경변수 설정

- PYTHONHOME=D:\prod\Anaconda3
- PYTHONPATH=D:\prod\Anaconda3



- PIP(Python Install Package) 프로그램은 파이썬의 오픈소스 저장소인 PyPI(Python Package Index)에 있는 SW 패키지를 설치하고 관리해주는 명령이다.

```
D:\dev\workspace\django1>pip install Django
D:\dev\workspace\django1>python -m django  --version
3.2.9
```

3.2.2 기존 장고 프로그램 삭제

- pip 프로그램으로 설치하는 경우는 pip 프로그램이 예전 버전의 장고 프로그램을 자동으로 처리해주므로 삭제 과정은 필요하지 않다.

```
# 장고 프로그램 설치 확인
D:\dev\workspace\django1>python -m django --version
3.2.9

# 장고가 설치된 디렉토리 위치 파악
D:\dev\workspace\django1>python -c "import django; print(django.__path__)"
['D:\\prod\\Anaconda3\\lib\\site-packages\\django']
```

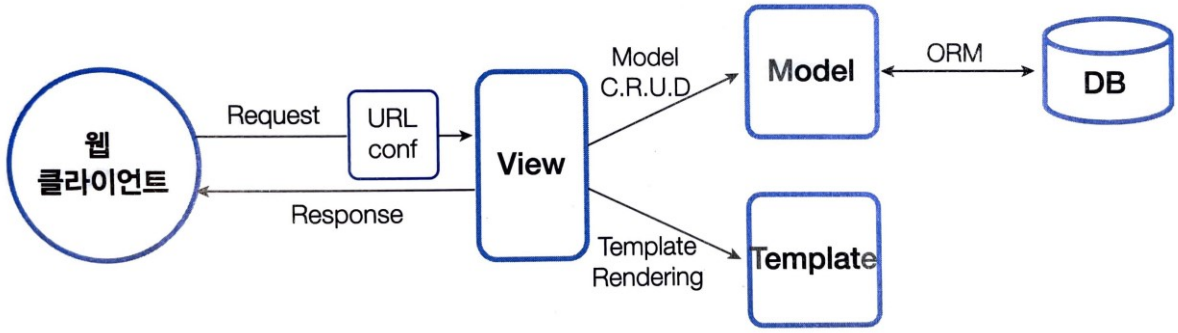
3.3 장고에서의 애플리케이션 개발 방식

- 장고에서는 용어를 사용할 때, 웹 사이트에 대한 전체 프로그램을 프로젝트(Project)라 하고 모듈화된 단위 프로그램을 애플리케이션(Application)이라 부르고 있다. 즉, 애플리케이션 프로그램들을 모와서 프로젝트를 개발하는 개념이다.

3.3.1 MVT 패턴

- 장고 프레임워크에서는 View를 Template, Controller는 View라고 표현하며, MVC 대신에 MVT

패턴이라고 한다.



3.3.2 Model - 데이터베이스 정의

- 모델이란 사용될 데이터에 대한 정의를 담고 있는 장고의 클래스이다.
- 장고는 ORM 기법을 사용하여 애플리케이션에서 사용할 데이터베이스를 클래스로 매핑해서 코딩할 수 있다. 즉, 하나의 모델 클래스는 하나의 테이블에 매핑되고 모델 클래스의 속성은 테이블의 컬럼에 매핑된다.
- 또한 SQLite3, MySQL, PostgreSQL 등 데이터베이스 엔진을 변경하더라도 ORM을 통한 API는 변경할 필요가 없기 때문에, 필요에 따라 데이터베이스 엔진을 훨씬 쉽게 변경할 수 있다.
- 모델 클래스는 models.py 파일에 정의한다.
  - 04: Primary Key는 Question 클래스에서 정의하지 않아도 장고에서 자동으로 부여한다. 개발자가 직접 지정할 수도 있다.

[ch3/polls/models.py]

```
01  from django.db import models
02
03
04  class Question(models.Model):
05      question_text = models.CharField(max_length=200)
06      pub_date = models.DateTimeField('date published')
07
08      def __str__(self):
09          return self.question_text
10
11
12  class Choice(models.Model):
13      question = models.ForeignKey(Question, on_delete=models.CASCADE)
14      choice_text = models.CharField(max_length=200)
15      votes = models.IntegerField(default=0)
16
17      def __str__(self):
18          return self.choice_text
```

[꿀팁] ORM이란?

- ORM(Object-Relation Mapping)은 쉽게 표현하면 객체와 관계형 데이터베이스를 연결해주는

역할을 한다.

- 개발자는 객체를 대상으로 필요한 작업을 실행하면 ORM이 자동으로 적절한 SQL 구문이나 데이터베이스 API를 호출해서 처리해 준다.

3.3.3 URLconf - URL 정의

- 클라이언트로부터 요청을 받으면 장고는 가장 먼저 요청에 들어있는 URL을 분석한다. 즉 요청에 들어있는 URL이 urls.py 파일에 정의된 URL 패턴과 매칭되는지를 분석한다.

[ch3/polls/urls.py]

```
01  from django.urls import path
02  from polls import views
03
04
05  app_name = 'polls'
06  urlpatterns = [
07      path('', views.index, name='index'),      # /polls/
08      path('<int:question_id>/', views.detail, name='detail'),       # /polls/5/
09      path('<int:question_id>/results/', views.results, name='results'),     # /polls/5/results/
10      path('<int:question_id>/vote/', views.vote, name='vote'),      # /polls/5/vote/
11  ]
```

3.3.4 View - 로직 정의

- 장고는 웹 요청에 있는 URL을 분석하고, 그 결과로 해당 URL에 매핑된 뷰를 호출한다.

[ch3/polls/views.py]

```
01  from django.shortcuts import get_object_or_404, render
02  from django.http import HttpResponseRedirect
03  from django.urls import reverse
04
05  from polls.models import Choice, Question
06
07
08  def index(request):
09      latest_question_list = Question.objects.all().order_by('-pub_date')[:5]
10      context = {'latest_question_list': latest_question_list}
11      return render(request, 'polls/index.html', context)
12
13  def detail(request, question_id):
14      question = get_object_or_404(Question, pk=question_id)
15      return render(request, 'polls/detail.html', {'question': question})
16
17  def results(request, question_id):
18      question = get_object_or_404(Question, pk=question_id)
19      return render(request, 'polls/results.html', {'question': question})
20
21  def vote(request, question_id):
22      question = get_object_or_404(Question, pk=question_id)
23      try:
24          selected_choice = question.choice_set.get(pk=request.POST['choice'])
25      except (KeyError, Choice.DoesNotExist):
26          # Redisplay the question voting form.
27          return render(request, 'polls/detail.html', {
```

```
            'question': question,
            'error_message': "You didn't select a choice.",
        })
    else:
        selected_choice.votes += 1
        selected_choice.save()
        # Always return an HttpResponseRedirect after successfully dealing
        # with POST data. This prevents data from being posted twice if a
        # user hits the Back button.
        return HttpResponseRedirect(reverse('polls:results', args=(question.id,)))
```

3.3.5 Template - 화면 UI 정의

- 개발자가 작성하는 *.html 파일을 템플릿이라 부르며, 여기에 화면 UI 모습을 템플릿 문법에 맞게 작성한다.

- 장고에서 템플릿 파일을 찾을 때는 TEMPLATES 및 INSTALLED_APPS에서 지정된 앱의 디렉토리를 검색한다. 이 항목들은 프로젝트 설정 파일인 settings.py 파일에 정의되어 있다.
  - 19: BASE_DIR은 'D:\dev\workspace\django1\ch3'이다. 장고는 TEMPLATES 항목에 정의된 디렉토리를 먼저 찾고, 그 다음엔 INSTALLED_APPS 항목에 등록된 각 앱의 templates 디렉토리를 찾는다.

[ch3/mysite/settings.py 파일의 일부]

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

...(생략)...

ROOT_URLCONF = 'mysite.urls'

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

3.3.6 MVT 코딩 순서

■ 모델, 뷰, 템플릿 셋 중에서 무엇을 먼저 코딩해야 하는지에 대해 정해진 순서는 없다.

3.4 애플리케이션 설계하기

■ 요구사항 분석
- 우리가 개발하게 될 애플리케이션의 내용은 설문에 해당하는 질문을 보여준 후 질문에 포함되어 있는 답변 항목에 투표하면 그 결과를 알려주는 예제이다.

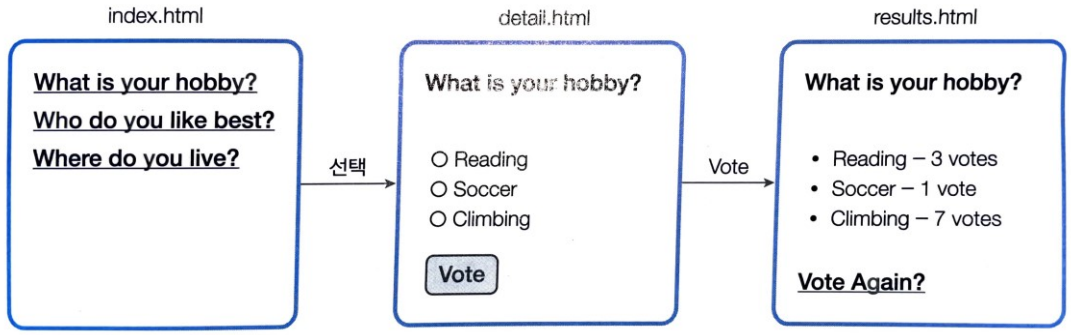
■ DB 설계
- Question 테이블: 질문을 저장하는 테이블이다.

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

- Choice 테이블: 질문별로 선택용 답변 항목을 저장하는 테이블이다.

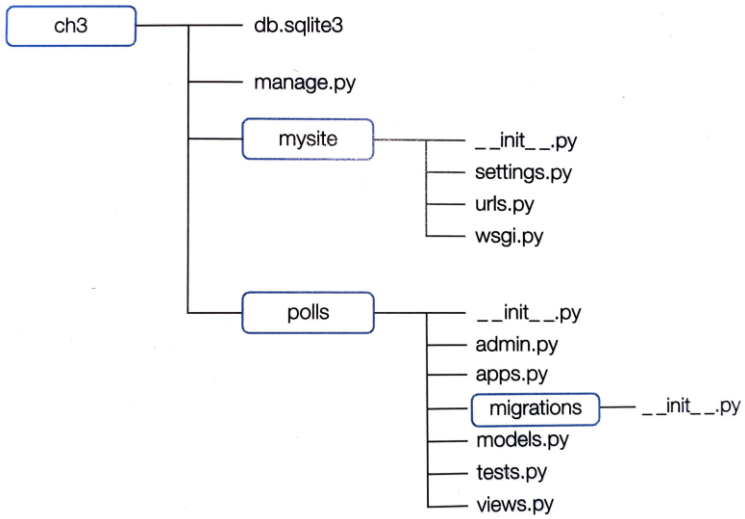
컬럼명	타입	제약 조건	설명
id	integer	NotNull, PK, AutoIncrement	Primary Key
choice_text	varchar(200)	NotNull	답변 항목 문구
votes	integer	NotNull	투표 카운트
question	integer	NotNull, FK(Question.id)	Foreign Key

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



3.5 프로젝트 뼈대 만들기

■ 프로젝트 뼈대가 완성된 후의 디렉토리 체계는 아래와 같다.



■ 뼈대 디렉토리 및 파일에 대한 설명

항목명	설명
ch3 디렉토리	프로젝트 관련 디렉토리 및 파일을 모아주는 최상위 루트 디렉토리이다.
db.sqlite3	SQLite3 데이터베이스 파일이다. 테이블이 들어있다.
manage.py	장고의 명령어를 처리하는 파일이다.
mysite 디렉토리	프로젝트명으로 만들어진 디렉토리이다. 프로젝트 관련 파일들이 들어있다.
__init__.py	디렉토리에 이 파일이 있으면 파이썬 패키지로 인식한다.
settings.py	프로젝트 설정 파일이다.
urls.py	프로젝트 레벨의 URL 패턴을 정의하는 최상위 URLconf이다.
polls 디렉토리	애플리케이션명으로 만들어진 애플리케이션 디렉토리이다. 해당 애플리케이션 관련 파일들이 들어있다.
__init__.py	디렉토리에 이 파일이 있으면 파이썬 패키지로 인식한다.
admin.py	Admin 사이트에 모델 클래스를 등록해주는 파일이다.
apps.py	애플리케이션의 설정 클래스를 정의하는 파일이다.
migrations 디렉토리	데이터베이스 변경사항을 관리하기 위한 디렉토리이다. 데이터베이스에 추가, 삭제, 변경 등이 발생하면 변경 내역을 기록한 파일들이 위치한다.
models.py	데이터베이스 모델 클래스를 정의하는 파일이다.
tests.py	단위 테스트용 파일이다.
views.py	뷰 함수를 정의하는 파일이다. 함수형 뷰 및 클래스형 뷰 모두 이 파일에 정의한다.
templates 디렉토리	프로젝트를 진행하면서 추가된다. 템플릿 파일들이 들어있다.
static 디렉토리	프로젝트를 진행하면서 추가된다. CSS, Image, Javascript 파일들이 들어있다.
logs 디렉토리	프로젝트를 진행하면서 추가된다. 로그 파일들이 들어있다. 로그 파일의 위치는 settings.py 파일에서 LOGGING 항목으로 지정한다.

3.5.1 프로젝트 생성

■ 아래의 명령으로 mysite라는 프로젝트를 만든다.

```
D:\dev\workspace\django1>django-admin startproject mysite

D:\dev\workspace\django1>rename mysite ch3Lab

D:\dev\workspace\django1>cd ch3Lab

D:\dev\workspace\django1\ch3Lab>dir
 D 드라이브의 볼륨: disk2
 볼륨 일련 번호: 8493-73C8
```

```
 D:\dev\workspace\django1\ch3Lab 디렉터리

2021-11-11  오전 11:59    <DIR>          .
2021-11-11  오전 11:59    <DIR>          ..
2021-11-11  오전 11:59               684 manage.py
2021-11-11  오전 11:59    <DIR>          mysite
               1개 파일                 684 바이트
               3개 디렉터리  240,851,427,328 바이트 남음

D:\dev\workspace\django1\ch3Lab>cd mysite

D:\dev\workspace\django1\ch3Lab\mysite>dir
 D 드라이브의 볼륨: disk2
 볼륨 일련 번호: 8493-73C8

 D:\dev\workspace\django1\ch3Lab\mysite 디렉터리

2021-11-11  오전 11:59    <DIR>          .
2021-11-11  오전 11:59    <DIR>          ..
2021-11-11  오전 11:59               405 asgi.py
2021-11-11  오전 11:59             3,364 settings.py
2021-11-11  오전 11:59               769 urls.py
2021-11-11  오전 11:59               405 wsgi.py
2021-11-11  오전 11:59                 0 __init__.py
               5개 파일               4,943 바이트
               2개 디렉터리  240,851,427,328 바이트 남음
```

3.5.2 애플리케이션 생성

■ polls라는 애플리케이션을 만드는 명령을 실행한다.

```
D:\dev\workspace\django1\ch3Lab>python manage.py startapp polls

D:\dev\workspace\django1\ch3Lab>dir
 D 드라이브의 볼륨: disk2
 볼륨 일련 번호: 8493-73C8

 D:\dev\workspace\django1\ch3Lab 디렉터리

2021-11-11  오후 12:10    <DIR>          .
2021-11-11  오후 12:10    <DIR>          ..
2021-11-11  오전 11:59               684 manage.py
2021-11-11  오후 12:10    <DIR>          mysite
2021-11-11  오후 12:10    <DIR>          polls
               1개 파일                 684 바이트
               4개 디렉터리  240,851,419,136 바이트 남음

D:\dev\workspace\django1\ch3Lab>dir polls
 D 드라이브의 볼륨: disk2
 볼륨 일련 번호: 8493-73C8

 D:\dev\workspace\django1\ch3Lab\polls 디렉터리

2021-11-11  오후 12:10    <DIR>          .
2021-11-11  오후 12:10    <DIR>          ..
2021-11-11  오후 12:10                66 admin.py
2021-11-11  오후 12:10               148 apps.py
2021-11-11  오후 12:10    <DIR>          migrations
2021-11-11  오후 12:10                60 models.py
2021-11-11  오후 12:10                63 tests.py
```

```
2021-11-11  오후 12:10                66 views.py
2021-11-11  오후 12:10                 0 __init__.py
               6개 파일                 403 바이트
               3개 디렉터리  240,851,419,136 바이트 남음
```

3.5.3 프로젝트 설정 파일 변경

■ 프로젝트에 필요한 설정값들은 settings.py 파일에 지정한다.

- 26: DEBUG=True이면 개발 모드로, False이면 운영 모드로 인식한다.
- 29: 운영 모드인 경우 ALLOWED_HOSTS에 반드시 서버의 IP나 도메인을 지정해야 하고, 개발 모두의 경우에는 값을 지정하지 않아도 ['localhost', '127.0.0.1']로 간주한다.
- 41: polls 앱의 설정 클래스는 startapp polls 명령 시에 자동 생성된 apps.py 파일에 PollsConfig라고 정의되어 있다. 그래서 장고가 설정 클래스를 찾을 수 있도록 모듈 경로까지 포함하여 'polls.apps.PollsConfig'라고 등록한다.
- 78~83: 데이터베이스 설정 항목을 확인할 수 있다.
- 111: 타임존 지정으로 최초에는 세계표준시(UTC)로 되어 있는데 한국 시간으로 변경한다.

[ch2Lab/mysite/settings.py]

```
01  """
02  Django settings for mysite project.
03
04  Generated by 'django-admin startproject' using Django 3.2.9.
05
06  For more information on this file, see
07  https://docs.djangoproject.com/en/3.2/topics/settings/
08
09  For the full list of settings and their values, see
10  https://docs.djangoproject.com/en/3.2/ref/settings/
11  """
12
13  from pathlib import Path
14
15  # Build paths inside the project like this: BASE_DIR / 'subdir'.
16  BASE_DIR = Path(__file__).resolve().parent.parent
17
18
19  # Quick-start development settings - unsuitable for production
20  # See https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/
21
22  # SECURITY WARNING: keep the secret key used in production secret!
23  SECRET_KEY = 'django-insecure-_-mmz+e_uuh-(87iq650$lh((x+1nq7n@v#b5wznw(8n(w0*ld'
24
25  # SECURITY WARNING: don't run with debug turned on in production!
26  DEBUG = True
27
28  # ALLOWED_HOSTS = []
29  ALLOWED_HOSTS = ['192.168.18.163', 'localhost', '127.0.0.1']    # 변경
30
31
32  # Application definition
33
34  INSTALLED_APPS = [
35      'django.contrib.admin',
36      'django.contrib.auth',
37      'django.contrib.contenttypes',
38      'django.contrib.sessions',
```

```
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'polls.apps.PollsConfig',           # 추가
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'mysite.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        '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',
            ],
        },
    },
]

WSGI_APPLICATION = 'mysite.wsgi.application'


# Database
# https://docs.djangoproject.com/en/3.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


# Password validation
# https://docs.djangoproject.com/en/3.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

```

```
104
105  # Internationalization
106  # https://docs.djangoproject.com/en/3.2/topics/i18n/
107
108  LANGUAGE_CODE = 'en-us'
109
110  # TIME_ZONE = 'UTC'
111  TIME_ZONE = 'Asia/Seoul'      # 변경
112
113
114  USE_I18N = True
115
116  USE_L10N = True
117
118  USE_TZ = True
119
120
121  # Static files (CSS, JavaScript, Images)
122  # https://docs.djangoproject.com/en/3.2/howto/static-files/
123
124  STATIC_URL = '/static/'
125
126  # Default primary key field type
127  # https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field
128
129  DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

3.5.4 기본 테이블 생성

■ migrate 명령은 데이터베이스에 변경사항이 있을 때 이를 반영해주는 명령이다.

```
D:\dev\workspace\django1\ch3Lab>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 auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK

D:\dev\workspace\django1\ch3Lab>dir
 D 드라이브의 볼륨: disk2
 볼륨 일련 번호: 8493-73C8

 D:\dev\workspace\django1\ch3Lab 디렉터리
```

```
2021-11-11  오후 12:52    <DIR>          .
2021-11-11  오후 12:52    <DIR>          ..
2021-11-11  오후 12:52           131,072 db.sqlite3
2021-11-11  오전 11:59               684 manage.py
2021-11-11  오후 12:10    <DIR>          mysite
2021-11-11  오후 12:52    <DIR>          polls
               2개 파일             131,756 바이트
               4개 디렉터리  240,851,283,968 바이트 남음
```

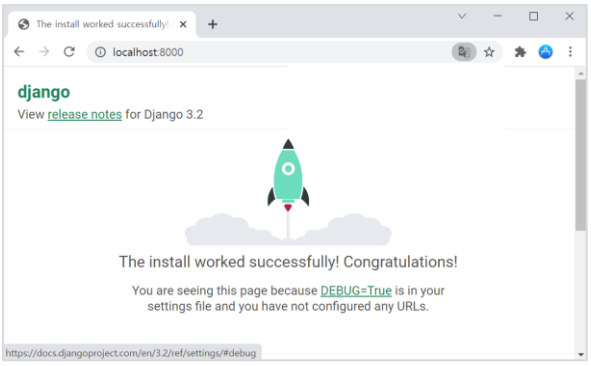
3.5.5 지금까지 작업 확인하기

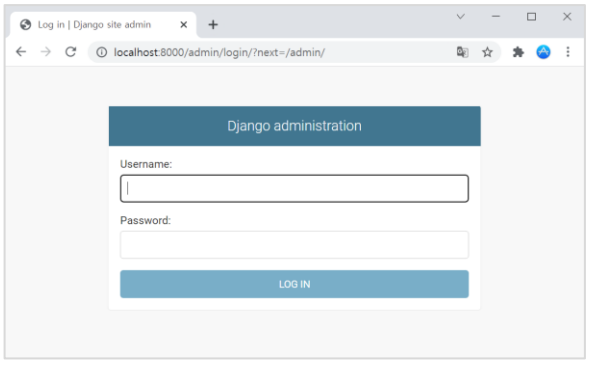
■ 명령 입력 시 0.0.0.0이란 IP 주소의 의미는 현재 명령을 실행 중인 서버의 IP 주소가 무엇으로 설정되어 있더라도 그와는 무관하게 웹 접속 요청을 받겠다는 의미이다.

```
#  Admin 사이트에 로그인하기 위한 관리자(슈퍼유저)를 만든다.
D:\dev\workspace\django1\ch3Lab>python manage.py createsuperuser
Username (leave blank to use 'admin'):
Email address: admin@naver.com
Password:
Password (again):
The password is too similar to the username.
This password is too short. It must contain at least 8 characters.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.

# 웹 서버를 실행한다.
D:\dev\workspace\django1\ch3Lab>python manage.py runserver 0.0.0.0:8000
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
November 11, 2021 - 12:55:02
Django version 3.2.9, using settings 'mysite.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CTRL-BREAK.
```





■ 프로젝트 디렉토리의 현재 모습을 보기 위해서 다음 명령을 입력한다.

```
D:\dev\workspace\django1>tree /F ch3Lab
disk2 볼륨에 대한 폴더 경로의 목록입니다.
볼륨 일련 번호가 00000071 8493:73C8입니다.
D:\DEV\WORKSPACE\DJANGO1\CH3LAB
|   db.sqlite3
```

```
|   manage.py
|
+---mysite
|   |   asgi.py
|   |   settings.py
|   |   urls.py
|   |   wsgi.py
|   |   __init__.py
|   |
|   \---__pycache__
|           settings.cpython-38.pyc
|           urls.cpython-38.pyc
|           wsgi.cpython-38.pyc
|           __init__.cpython-38.pyc
|
\---polls
    |   admin.py
    |   apps.py
    |   models.py
    |   tests.py
    |   views.py
    |   __init__.py
    |
    +---migrations
    |   |   __init__.py
    |   |
    |   \---__pycache__
    |           __init__.cpython-38.pyc
    |
    \---__pycache__
            admin.cpython-38.pyc
            apps.cpython-38.pyc
            models.cpython-38.pyc
            __init__.cpython-38.pyc
```

3.6 애플리케이션 개발하기 - Model 코딩

3.6.1 테이블 정의

■ polls 애플리케이션은 Question과 Choice 두 개의 테이블이 필요하다.

[ch3Lab/polls/models.py]

```
01  from django.db import models
02
03
04  class Question(models.Model):
05      question_text = models.CharField(max_length=200)
06      pub_date = models.DateTimeField('date published')
07
08      def __str__(self):
09          return self.question_text
10
11
12  class Choice(models.Model):
13      question = models.ForeignKey(Question, on_delete=models.CASCADE)
14      choice_text = models.CharField(max_length=200)
15      votes = models.IntegerField(default=0)
16
17      def __str__(self):
```

```
18          return self.choice_text
```

3.6.2 Admin 사이트에 테이블 반영

- Admin 사이트에 접속해보면 현재까지는 장고에서 기본적으로 제공하는 Users, Groups 테이블만 보인다. 이제 models.py 파일에서 정의한 테이블도 Admin 사이트에 보이도록 등록한다.

[ch3Lab/polls/admin.py]

```
01  from django.contrib import admin
02  from polls.models import Question, Choice
03
04  admin.site.register(Question)
05  admin.site.register(Choice)
```

3.6.3 데이터베이스 변경사항 반영

- 테이블의 신규 생성, 테이블의 정의 변경 등 데이터베이스에 변경이 필요한 사항이 있으면 이를 데이터베이스에 실제로 반영해주는 작업을 해야 한다. 아직까지는 클래스로 테이블 정의만 변경한 상태이다. 다음 명령으로 변경사항을 데이터베이스에 반영한다.

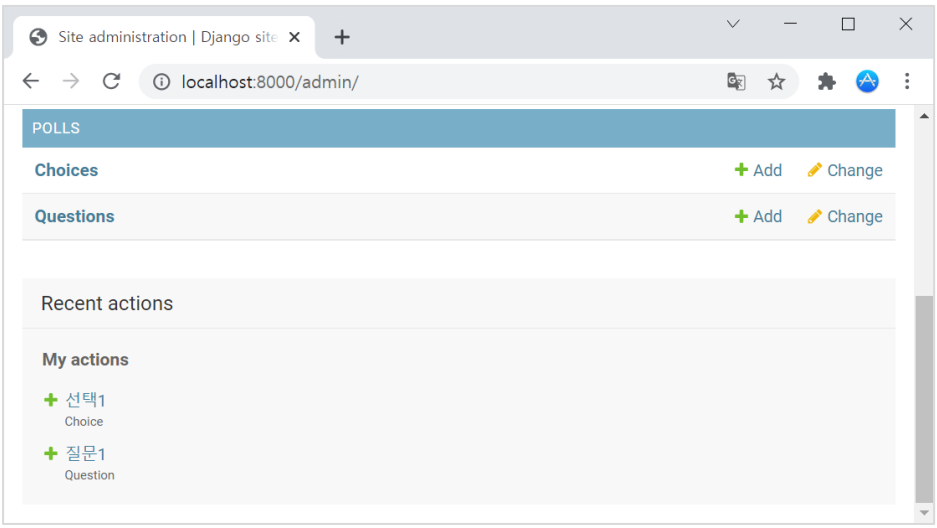
```
D:\dev\workspace\django1>cd ch3Lab

D:\dev\workspace\django1\ch3Lab>python manage.py makemigrations
Migrations for 'polls':
  polls\migrations\0001_initial.py
    - Create model Question
    - Create model Choice

D:\dev\workspace\django1\ch3Lab>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, polls, sessions
Running migrations:
  Applying polls.0001_initial... OK
```

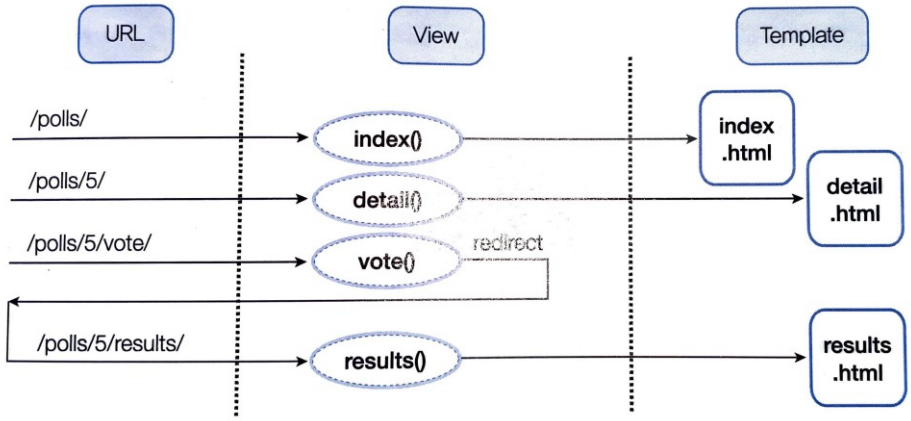
3.6.4 지금까지 작업 확인하기

- Admin 사이트에 접속한 후에 Questions와 Choices 테이블에 데이터를 입력해 본다.



3.7 애플리케이션 개발하기 - View 및 Template 코딩

■ 처리 흐름 설계



3.7.1 URLconf 코딩

■ URLconf를 코딩할 때 하나의 urls.py 파일에 작성할 수도 있고, 다음과 같이 mysite/urls.py와 polls/urls.py 2개의 파일에 작성할 수도 있다. 어떤 방식이 좋을까? 두 번째가 좋은 방법이다. 즉, URLconf 모듈을 계층적으로 구성하는 것이 변경도 쉬워지고, 확장도 용이해지지 때문이다.

[ch3Lab/mysite/urls.py]

```
01  """mysite URL Configuration
02
03  The `urlpatterns` list routes URLs to views. For more information please see:
04      https://docs.djangoproject.com/en/3.2/topics/http/urls/
05  Examples:
06  Function views
07      1. Add an import:  from my_app import views
08      2. Add a URL to urlpatterns:  path('', views.home, name='home')
```

```
09  Class-based views
10      1. Add an import:  from other_app.views import Home
11      2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
12  Including another URLconf
13      1. Import the include() function: from django.urls import include, path
14      2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
15  """
16  from django.contrib import admin
17  from django.urls import path, include
18
19
20  urlpatterns = [
21      path('admin/', admin.site.urls),
22
23      path('polls/', include('polls.urls')),  # 추가
24  ]
```

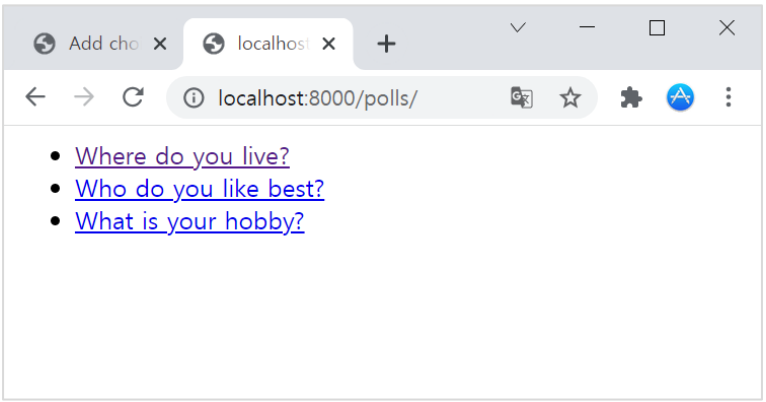
[ch3Lab/polls/urls.py]

```
01  from django.urls import path
02  from polls import views
03
04
05  app_name = 'polls'
06  urlpatterns = [
07      path('', views.index, name='index'),      # /polls/
08      path('<int:question_id>/', views.detail, name='detail'),       # /polls/5/
09      path('<int:question_id>/results/', views.results, name='results'),     # /polls/5/results/
10      path('<int:question_id>/vote/', views.vote, name='vote'),      # /polls/5/vote/
11  ]
```

3.7.2 뷰 함수 index() 및 템플릿 작성

■ 뷰 함수와 템플릿은 서로에게 영향을 미치기 때문에 보통 같이 작업하게 된다. 다만, UI 화면을 생각하면서 로직을 풀어나가는 것이 쉽기 때문에 뷰보다는 템플릿을 먼저 코딩하는 것을 추천한다.

■ 화면 UI 설계 - index.html



■ 위 화면의 내용을 구현하기 위해 템플릿 파일 index.html에 다음과 같이 입력한다.
- 01: latest_question_lisst 객체는 index() 뷰 함수에서 넘겨주는 파라미터이다.

[ch3Lab/polls/templates/polls/index.html]

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

■ index() 함수 작성

[ch3Lab/polls/views.py]

```
from django.shortcuts import get_object_or_404, render
from django.http import HttpResponseRedirect
from django.urls import reverse

from polls.models import Choice, Question


def index(request):
    latest_question_list = Question.objects.all().order_by('-pub_date')[:5]
    context = {'latest_question_list': latest_question_list}
    return render(request, 'polls/index.html', context)

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

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

def vote(request, question_id):
    question = get_object_or_404(Question, pk=question_id)
    try:
        selected_choice = question.choice_set.get(pk=request.POST['choice'])
    except (KeyError, Choice.DoesNotExist):
        # Redisplay the question voting form.
        return render(request, 'polls/detail.html', {
            'question': question,
            'error_message': "You didn't select a choice.",
        })
    else:
        selected_choice.votes += 1
        selected_choice.save()
        # Always return an HttpResponseRedirect after successfully dealing
        # with POST data. This prevents data from being posted twice if a
        # user hits the Back button.
        return HttpResponseRedirect(reverse('polls:results', args=(question.id,)))
```

3.7.3 뷰 함수 detail() 및 폼 템플릿 작성

■ 화면 UI 설계 - detail.html

■ 위 화면의 내용을 템플릿 파일인 detail.html에 다음과 같이 입력한다.

[ch3Lab/polls/templates/polls/details.html]

```
<h1>{{ question.question_text }}</h1>

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

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

3.7.4 뷰 함수 vote() 및 리다이렉션 작성

■ vote() 뷰 함수의 호출과 연계된 URL은 detail.html 템플릿 파일에서 받는다.

[ch3Lab/polls/views.py]

```
from django.shortcuts import get_object_or_404, render
from django.http import HttpResponseRedirect
from django.urls import reverse

from polls.models import Choice, Question


def index(request):
    latest_question_list = Question.objects.all().order_by('-pub_date')[:5]
    context = {'latest_question_list': latest_question_list}
    return render(request, 'polls/index.html', context)

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

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

def vote(request, question_id):
    question = get_object_or_404(Question, pk=question_id)
    try:
        selected_choice = question.choice_set.get(pk=request.POST['choice'])
    except (KeyError, Choice.DoesNotExist):
        # Redisplay the question voting form.
```

```
27          return render(request, 'polls/detail.html', {
28              'question': question,
29              'error_message': "You didn't select a choice.",
30          })
31      else:
32          selected_choice.votes += 1
33          selected_choice.save()
34          # Always return an HttpResponseRedirect after successfully dealing
35          # with POST data. This prevents data from being posted twice if a
36          # user hits the Back button.
37          return HttpResponseRedirect(reverse('polls:results', args=(question.id,)))
```

3.7.5 뷰 함수 results() 및 템플릿 작성

■ results() 뷰 함수의 호출과 연계된 URL은 votes() 뷰 함수의 리다이렉트 결과로 받는다.

[ch3Lab/polls/views.py]

```
01  from django.shortcuts import get_object_or_404, render
02  from django.http import HttpResponseRedirect
03  from django.urls import reverse
04
05  from polls.models import Choice, Question
06
07
08  def index(request):
09      latest_question_list = Question.objects.all().order_by('-pub_date')[:5]
10      context = {'latest_question_list': latest_question_list}
11      return render(request, 'polls/index.html', context)
12
13  def detail(request, question_id):
14      question = get_object_or_404(Question, pk=question_id)
15      return render(request, 'polls/detail.html', {'question': question})
16
17  def results(request, question_id):
18      question = get_object_or_404(Question, pk=question_id)
19      return render(request, 'polls/results.html', {'question': question})
20
21  def vote(request, question_id):
22      question = get_object_or_404(Question, pk=question_id)
23      try:
24          selected_choice = question.choice_set.get(pk=request.POST['choice'])
25      except (KeyError, Choice.DoesNotExist):
26          # Redisplay the question voting form.
27          return render(request, 'polls/detail.html', {
28              'question': question,
29              'error_message': "You didn't select a choice.",
30          })
31      else:
32          selected_choice.votes += 1
33          selected_choice.save()
34          # Always return an HttpResponseRedirect after successfully dealing
35          # with POST data. This prevents data from being posted twice if a
36          # user hits the Back button.
37          return HttpResponseRedirect(reverse('polls:results', args=(question.id,)))
```

3.7.6 지금까지 작업 확인하기

■ 지금까지 설계 로직에 따라 필요한 뷰와 템플릿 코딩을 마쳤다. 아래 그림처럼 polls 애플리케이션의 첫 화면이 나타나면 정상이다.