

2장. 모델과 템플릿 구현



데이터베이스와 모델

장고는 모델(Model)로 데이터를 관리한다. 데이터를 저장하고 조회, 삭제하는 등의 관리를 기존의 SQL 언어로 하지 않고 클래스(객체)를 사용한다.

◆ 장고 서버 구동시 경고 메시지

```
(mysite) c:\Webproject\Wpolls>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly
without migrations.
Run 'python manage.py migrate' to apply them.
```

※ 아직 적용되지 않은 18개의 migration이 있다.

데이터베이스와 모델

◆ migrate 명령으로 앱 설정

```
(mysite)C:\projects\polls>python manage.py migrate
```

```
(mysite) c:\Webproject\polls>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
```

◆ config/settings.py – 기본 설치 앱 확인

```
INSTALLED_APPS = [
    'poll.apps.PollConfig',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
```

데이터베이스와 모델

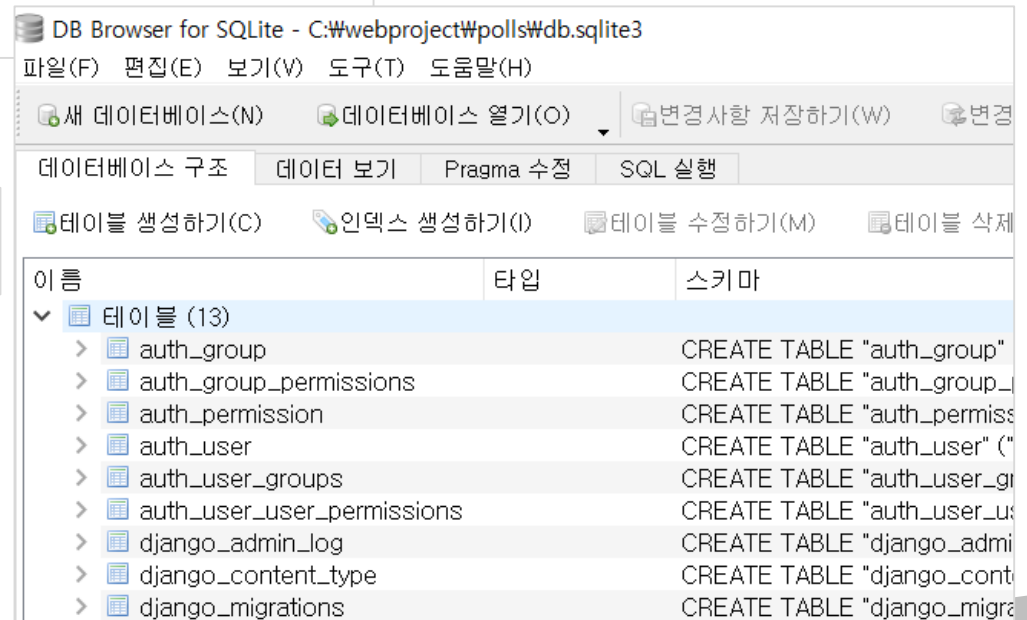
◆ DB Browser for SQLite로 데이터베이스 열기

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

config/setting.py

db 위치

C:\projects\polls\db.sqlite3



ORM(Object Relational Mapping)

◆ ORM(Object Relational Mapping) – 객체 관계 매핑 방식

파이썬은 현재 가장 많이 사용하고 있는 관계형 데이터베이스 방식을 개선한 ORM 방식을 사용한다.

테이블을 모델화(클래스의 객체)하여 DB를 획기적으로 관리할 수 있으며, 특히 SQL 언어의 문법을 사용하지 않는다.

SQL을 사용했을 경우 단점은

- SQL 쿼리문은 개발자마다 다양하게 작성되므로 일관성이나 유지 보수 측면에서 심플하지 않고 복잡할 수 있다.
- 또한, 개발자가 쿼리문 자체를 잘못 작성하여 시스템의 성능이 저하 될 수 있다.

ORM(Object Relational Mapping)

◆ Django Documentation > The model layer > introduction

This example model defines a **Person**, which has a **first_name** and **last_name**:

```
from django.db import models

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

first_name and **last_name** are **fields** of the model. Each field is specified as a class column.

The above **Person** model would create a database table like this:

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

모델(Model) – 필드 타입(type)

◆ 필드(속성) 타입

Field Type	설 명
CharField	제한된 문자열 타입, max length 옵션 사용
TextField	제한이 없는 가변 길이 대용량 문자열 타입
IntegerField	32bit 정수 타입 (-21억 ~ 21억)
BigIntegerField	64bit 정수 타입 (조, 경 이상)
DateTimeField	날짜와 시간을 갖는 타입
ImageField	이미지 업로드를 위한 필드
FileField	파일 업로드를 위한 필드

설문 투표 모델 – Question, Choice

◆ 설문 투표 앱에 사용할 Question, Choice 모델

Question 모델

칼럼 이름	자료형	설명
question_text	CharField(max_length=200)	질문내용
pub_date	DateTimeField()	출판일

Choice 모델

칼럼 이름	자료형	설명
choice_text	CharField(max_length=200)	답변내용
votes	IntegerField(default=0)	투표수
question	ForeignKey(Question, on_delete=models.CASCADE)	질문내용 (외래키)

Question 모델 만들기

- poll/models.py에 Question 모델 생성

```
from django.db import models
```

```
class Question(models.Model):
```

models.Model 을 상속받은 Question 클래스

```
    question_text = models.CharField(max_length=200)
```

```
    pub_date = models.DateTimeField()
```

```
    def __str__(self):
```

```
        return self.question_text
```

객체 정보를 문자열로 출력하는 함수

Question 모델 만들기

1. makemigrations 명령으로 테이블 관리 시작하기

```
(mysite)C:\projects\polls>python manage.py makemigrations
```

```
operations = [  
    migrations.CreateModel(  
        name='Question',  
        fields=[  
            ('id', models.BigAutoField(auto_created=True, primary_key=True,  
            ('subject', models.CharField(max_length=200)),  
            ('content', models.TextField()),  
            ('create_date', models.DateTimeField()),  
        ],  
    ),  
]
```

migrations/0001.initial.py

2. makemigrations 후 다시 migrate 명령 실행

```
(mysite)C:\projects\polls>python manage.py migrate
```

데이터 저장 및 조회하기

◆ 장고 셸 실행하기

```
(mysite)C:\projects\polls>python manage.py shell
```

◆ 데이터 입력 – Question 모델의 객체 생성

```
>>> from poll.models import Question, Choice
>>> from django.utils import timezone
>>> Question.objects.all()
<QuerySet []>
>>> q = Question(question_text= " 접종한 백신은 무엇인가요?", pub_date=timezone.now())
>>> q.save()
>>> q.id
1
>>> q.question_text
" 접종한 백신은 무엇인가요?"
>>> q.pub_date
datetime.datetime(2021, 7, 16, 13, 13, 45, 965555, tzinfo=<UTC>)
```

데이터 저장 및 조회하기

◆ 데이터 조회 – 주요 함수

```
>>> Question.objects.all() : 모두 조회(리스트)
```

```
>>> Question.objects.filter(id=1) : 1개 조회(리스트)
```

```
>>> Question.objects.get(id=1) : 1개의 데이터 조회
```

```
>>> quit() # 셸을 나갈때 사용
```

데이터 변경 및 삭제하기

◆ 데이터 수정

```
>>>q = Question.objects.get(id=1)
>>>q.subject = '접종받은 백신의 종류는?'
>>>q.save()
```

◆ 데이터 삭제

```
>>>q = Question.objects.get(id=1)
>>>q.delete()
```

Choice 모델 만들기

- poll/models.py에 Choice모델 생성

```
from django.db import models
```

```
class Choice(models.Model):
```

```
    choice_text = models.CharField(max_length=200)
```

```
    votes = models.IntegerField(default=0)
```

```
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
```

```
    #외래키(CASCADE-외래키에 연결된 데이터 삭제)
```

```
    def __str__(self):
```

```
        return self.choice_text
```

Choice 모델 만들기

◆ makemigrations 명령으로 테이블 관리 시작하기

```
(mysite)C:\projects\polls>python manage.py makemigrations
```

◆ makemigrations 후 다시 migrate 명령 실행

```
(mysite)C:\projects\polls>python manage.py migrate
```

Choice 모델 이용하기

◆ Choice 모델 – 데이터 입력 및 출력

```
>>> from poll.models import Question, Choice
>>> from django.utils import timezone
>>> Question.objects.all()
<QuerySet [<Question: 어떤 백신을 접종했나요?>]>
>>> q = Question.objects.get(id=1) : 1개의 질문 가져오기
>>> c = Choice(choice_text='화이자', votes=0, question=q)
>>> c.save()
>>> c = Choice(choice_text='얀센', votes=0, question=q)
>>> c.save()
>>> c = Choice(choice_text='모더나', votes=0, question=q)
>>> c.save()
>>> q.choice_set.all() : 연결된 데이터로 조회하기
<QuerySet [<Choice: 화이자>, <Choice: 얀센>, <Choice: 모더나>]>
```


장고 Admin

◆ 장고 Admin

장고는 Admin은 관리자 기능을 갖춘 app이다.
모델 관리를 shell 환경이 아닌 사이트 화면에서 할 수 있다.
127.0.0.1:8000/admin으로 접속.
먼저 슈퍼 유저를 생성해야 함.

```
(mysite)C:\projects\polls>python manage.py createsuperuser
```

```
사용자 이름 : admin
```

```
이메일 주소 :
```

```
Password:*****
```

```
Password(again):*****
```

장고 Admin

◆ localhost:8000/admin 에 접속하기

Django 관리

사용자 이름:

비밀번호:

로그인

admin / 12345

Django 관리

사이트 관리

POLL

Choices + 추가 ✎ 변경

Questions + 추가 ✎ 변경

인증 및 권한

그룹 + 추가 ✎ 변경

사용자(들) + 추가 ✎ 변경

장고 Admin

- ◆ 장고 Admin에서 모델 관리하기
 - poll/admin.py 에 등록

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

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

장고 Admin

◆ Question 추가하기

question 추가

Question text:	<input type="text" value="당신의 취미는 무엇입니까?"/>	
Pub date:	날짜: <input type="text" value="2021-12-12"/> 오늘 	
	시각: <input type="text" value="07:06:45"/> 현재 	



장고 Admin

◆ Choice 추가하기

choice 추가


Choice text:

Votes:

Question:  

✓ Choice "독서"가 성공적으로 추가되었습니다.

변경할 choice 선택

액션: 

<input type="checkbox"/>	CHOICE
<input type="checkbox"/>	독서
<input type="checkbox"/>	운동
<input type="checkbox"/>	영화 감상
<input type="checkbox"/>	컴퓨터 게임
<input type="checkbox"/>	모더나
<input type="checkbox"/>	안센
<input type="checkbox"/>	화이자

DB Browser for SQLite3

◆ DB Browser

DB Browser for SQLite - C:\webproject\polls\polls.db.sqlite3

파일(F) 편집(E) 보기(V) 도구(T) 도움말(H)

새 데이터베이스(N) 데이터베이스 열기(O) 변경사항 저장(S)

데이터베이스 구조 데이터 보기 Pragma 수정 SQL 실행

테이블(T): poll_question

	id	question_text	pub_date
...	필터		필터
1	1	어떤 백신을 접종했나요?	2021-12-11 13:29:46.879587
2	2	당신의 취미는 무엇입니까?	2021-12-11 22:04:04

새 데이터베이스(N) 데이터베이스 열기(O)

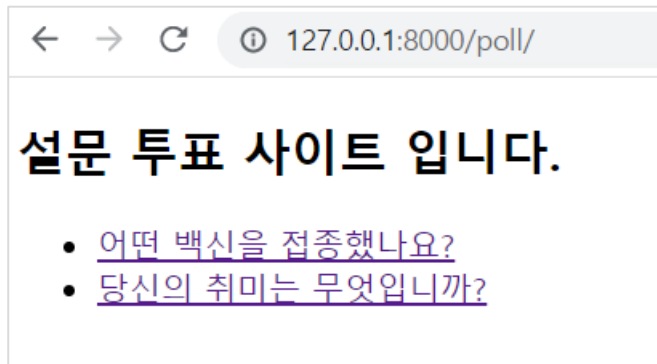
데이터베이스 구조 데이터 보기 Pragma 수정

테이블(T): poll_choice

	id	choice_text	votes	question_id
...	필터		필터	필터
1	1	화이자	3	1
2	2	얀센	2	1
3	3	모더나	1	1
4	4	컴퓨터 게임	0	2
5	5	영화 감상	0	2
6	6	운동	0	2
7	7	독서	0	2

템플릿으로 질문 목록 페이지 만들기

index.html 만들기- templates/poll/index.html



```
{% if question_list %}
    <ul>
        {% for question in question_list %}
            <li>
                <a href="/poll/{{question.id}}">
                    {{ question.question_text }}
                </a>
            </li>
        {% endfor %}
    </ul>
{% else %}
    <p>설문이 없습니다.</p>
{% endif %}
```

템플릿으로 질문 목록 페이지 만들기

◆ 템플릿(template) 태그

파이썬을 웹에 적용한 언어로 **{% %}** 블록을 사용한다.

템플릿 태그	설 명
<code>{% if item_list %}</code> .. 내용 .. <code>{% endif %}</code>	item_list가 있다면(조건문)
<code>{% for item in item_list %}</code> .. 내용 .. <code>{% endfor %}</code>	item_list를 반복하며 순차적으로 item에 대입(반복문)
<code>{{ id }}</code>	id 출력(출력문)
<code>{{ forloop.counter0 }}</code> <code>{{ forloop.counter }}</code>	for 반복문의 인덱스로 0부터 시작 0, 1 , 2, 3 ...

질문 상세 페이지 만들기

◆ 설문 상세 페이지

1. detail 페이지 URL 매핑하기 – pybo/urls.py

```
from django.urls import path
from pybo import views
urlpatterns = [
    path("", views.index),
    path('<int:question_id>/', views.detail),
]
```

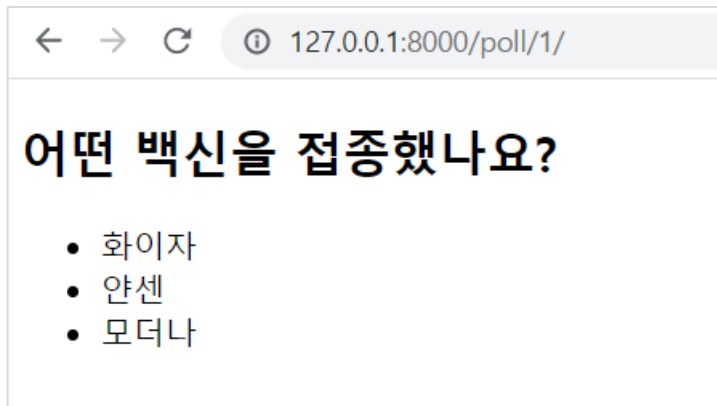
2. detail() 구현 – poll/views.py

```
def detail(request, question_id):
    question = Question.objects.get(id=question_id)
    return render(request, 'poll/detail.html', {'question': question})
```

설문 상세 페이지 만들기

◆ 설문 상세 페이지

3. poll/detail.html 작성



← → ↻ ⓘ 127.0.0.1:8000/poll/1/

어떤 백신을 접종했나요?

- 화이자
- 얀센
- 모더나

```
<h2>{{ question.question_text }}</h2>
<ul>
{% for choice in question.choice_set.all %}
    <li>{{ choice.choice_text }}</li>
{% endfor %}
</ul>
```

URL 네임 스페이스

◆ URL 별칭 사용하기 > 네임 스페이스

```
# 네임 스페이스 사용하기
app_name = 'poll' # app_name 변수에 이름 저장

urlpatterns = [
    path('', views.index, name='index'),
    path('<int:question_id>/', views.detail, name='detail'),
]
```

```
<ul>
{% for question in question_list %}
<!-- <li><a href="/poll/{{ question.id }}">
    {{ question.question_text }}</a></li>-->
    <li><a href="{% url 'poll:detail' question.id %}">
        {{ question.question_text }}</a></li>
{% endfor %}
</ul>
```

설문 투표하기

어떤 백신을 접종했나요?

☐ 화이자

☐ 얀센

☐ 모더나

투표

← → ↻ ⓘ 127.0.0.1:8000/poll/1/vote/

투표 결과

어떤 백신을 접종했나요?

- 화이자 -- 1votes
- 얀센 -- 1votes
- 모더나 -- 1votes

어떤 백신을 접종했나요?

선택을 확인하세요

☐ 화이자

☐ 얀센

☐ 모더나

투표

선택하지 않은 경우 에러처리

설문 투표하기

◆ 설문 상세 페이지 – 투표하기 폼으로 바꾸기

1. URL 매핑 추가하기 – poll/urls.py

```
# 네임 스페이스 사용하기
app_name = 'poll' # app_name 변수에 이름 저장

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

설문 투표하기

2. vote 함수 정의 – poll/views.py

```
def vote(request, question_id):
    question = Question.objects.get(id=question_id)
    try:
        sel_choice = question.choice_set.get(id=request.POST['choice'])
        # 선택한 name 속성 값을 저장
    except KeyError:
        return render(request, 'poll/detail.html',
                      {'question': question, 'error': '선택을 확인하세요'})
    else:
        sel_choice.votes = sel_choice.votes + 1 # 투표수 1 증가
        sel_choice.save()
        return render(request, 'poll/result.html', {'question': question})
```

설문 투표하기

3. 투표 양식 만들기 – poll/detail.html

```
<form action="{% url 'poll:vote' question.id %}" method="post">
    {% csrf_token %} <!-- 보안 필수 -->
    <fieldset>
        <legend>{{ question.question_text }}</legend>
        {% if error %} <!-- 선택하지 않은 경우 에러 처리 -->
            <p>{{ error }}</p>
        {% endif %}
        {% for choice in question.choice_set.all %}
            <p>
                <input type="radio" name="choice" id="choice" value="{{ choice.id }}">
                <label for="choice">{{ choice.choice_text }}</label>
            </p>
        {% endfor %}
    </fieldset>
    <p><input type="submit" value="투표"></p>
</form>
```

설문 투표하기

4. 투표 결과 페이지 – poll/results.html

```
<h2>{{ question.question_text }}</h2>
<ul>
    {% for choice in question.choice_set.all %}
        <li>
            {{ choice.choice_text }} -- {{ choice.votes }}votes
        </li>
    {% endfor %}
</ul>
```


설문 투표하기

➤ csrf_token 이 없는 경우 에러 발생

Forbidden (403)

CSRF 검증에 실패했습니다. 요청을 중단하였습니다.

Help

Reason given for failure:
CSRF token missing or incorrect.

In general, this can occur when there is a genuine Cross Site Request Forgery, or when [Django's CSRF mechanism](#) is

➤ csrf_token 템플릿 태그 사용하기

csrf_token

This tag is used for CSRF protection, as described in the documentation for [Cross Site Request Forgeries](#).

```
<form method="post">{% csrf_token %}
```

CSRF(cross-site request forgery)

➤ csrf

사이트 간 요청 위조

한글 20개 언어 ▼

위키백과, 우리 모두의 백과사전.

사이트 간 요청 위조(또는 **크로스 사이트 요청 위조**, **영어**: Cross-site request forgery, **CSRF**, **XSRF**)는 **웹사이트 취약점 공격**의 하나로, 사용자가 자신의 의지와는 무관하게 공격자가 의도한 행위(수정, 삭제, 등록 등)를 특정 웹사이트에 요청하게 하는 공격을 말한다.

유명 경매 사이트인 옥션에서 발생한 개인정보 유출 사건에서 사용된 공격 방식 중 하나다.

사이트 간 스크립팅(XSS)을 이용한 공격이 사용자가 특정 웹사이트를 신용하는 점을 노린 것이라면, 사이트간 요청 위조는 특정 웹사이트가 사용자의 **웹 브라우저**를 신용하는 상태를 노린 것이다. 일단 사용자가 웹사이트에 **로그인**한 상태에서 사이트간 요청 위조 공격 코드가 삽입된 페이지를 열면, 공격 대상이 되는 웹사이트는 위조된 공격 명령이 믿을 수 있는 사용자로부터 발송된 것으로 판단하게 되어 공격에 노출된다.

공격 과정 [편집]

1. 이용자는 웹사이트에 로그인하여 정상적인 **쿠키**를 발급받는다
2. 공격자는 다음과 같은 **링크**를 이메일이나 게시판 등의 경로를 통해 이용자에게 전달한다.
`http://www.geocities.com/attacker`
3. 공격용 **HTML** 페이지는 다음과 같은 이미지태그를 가진다.

```
<img src= "https://travel.service.com/travel_update?.src=Korea&.dst=Hell">
```

해당 링크는 클릭시 정상적인 경우 출발지와 도착지를 등록하기위한 링크이다. 위의 경우 도착지를 변조하였다.

4. 이용자가 공격용 페이지를 열면, 브라우저는 이미지 파일을 받아오기 위해 공격용 URL을 연다.
5. 이용자의 승인이나 인지 없이 출발지와 도착지가 등록됨으로써 공격이 완료된다. 해당 서비스 페이지는 등록 과정에 대해 단순히 쿠키를 통한 본인확인 밖에 하지 않으므로 공격자가 정상적인 이용자의 수정이 가능하게 된다.

csrf_token

➤ csrf_token

django는 csrf 공격에 대한 방어로 csrf_token을 발급 체크한다.

동작 과정

1. 사용자가 해당 페이지에 접속하면 Django에서 자동으로 csrf_token을 클라이언트로 보내어 cookie에 저장
2. 사용자가 form을 모두 입력한 후 제출버튼을 클릭한다.
3. form과 cookie의 csrf_token을 함께 POST로 전송한다.
4. 전송된 token의 유효성을 검증
5. 유효한 요청이면 요청을 처리
 1. token이 유효하지 않거나(없거나 값이 잘못된 경우) 검증 오류 시에는 403 Forbidden Response 반환

설문 투표하기

※ 투표수 초기화하기 – python manage.py shell

```
>>> from poll.models import Choice
>>> Choice.objects.all()
<QuerySet [<Choice: 화이자>, <Choice: 얀센>, <Choice: 모더나>,
서]>
>>> Choice.objects.get(id=1)
<Choice: 화이자>
>>> c = Choice.objects.get(id=1)
>>> c.choice_text
'화이자'
>>> c.votes
7
>>> c.votes = 0
>>> c.save()
```