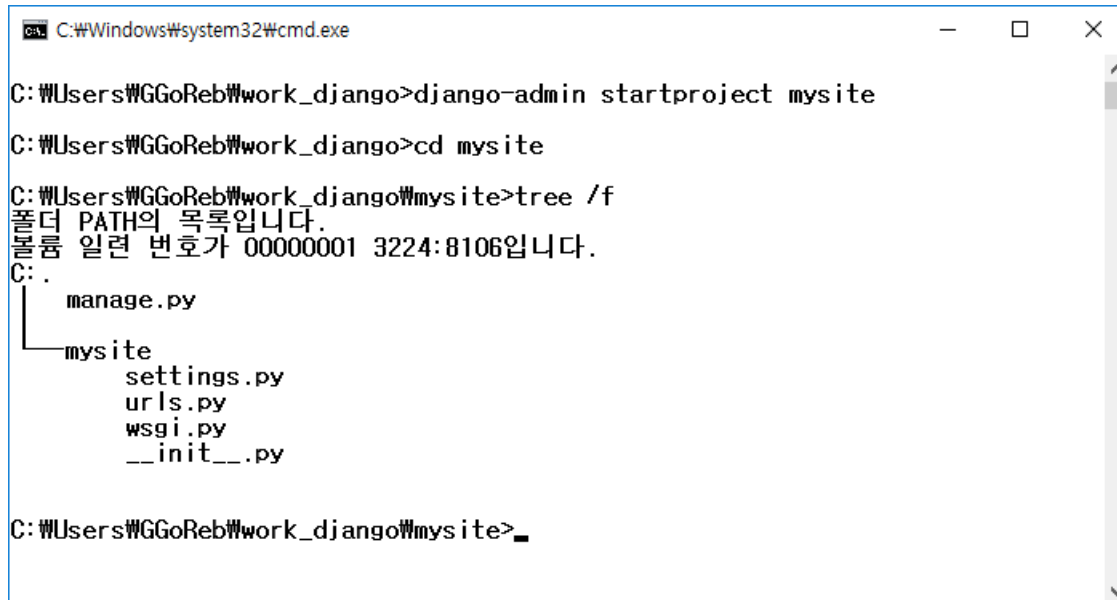


■ 장고 튜토리얼 투표앱

● 프로젝트 생성 및 구조 확인

– django-admin startproject mysite



```
C:\Windows\system32\cmd.exe

C:\Users\WGoReb\work_django>django-admin startproject mysite

C:\Users\WGoReb\work_django>cd mysite

C:\Users\WGoReb\work_django\mysite>tree /f
폴더 PATH의 목록입니다.
볼륨 일련 번호가 00000001 3224:8106입니다.
C:.
|
|_ manage.py
|_ mysite
|   |_ settings.py
|   |_ urls.py
|   |_ wsgi.py
|   |_ __init__.py

C:\Users\WGoReb\work_django\mysite>
```

■ 장고 튜토리얼 투표앱

● 앱 생성

– python manage.py startapp polls

```
C:\Windows\system32\cmd.exe

C:\Users\WGoReb\work_django\mysite>python manage.py startapp polls

C:\Users\WGoReb\work_django\mysite>tree /f
폴더 PATH의 목록입니다.
볼륨 일련 번호가 00000030 3224:8106입니다.
C:
|-- db.sqlite3
|-- manage.py
|-- mysite
|   |-- settings.py
|   |-- urls.py
|   |-- wsgi.py
|   |-- __init__.py
|   |-- __pycache__
|   |-- settings.cpython-36.pyc
|   |-- urls.cpython-36.pyc
|   |-- wsgi.cpython-36.pyc
|   |-- __init__.cpython-36.pyc
|-- __pycache__
|-- settings.cpython-36.pyc
|-- urls.cpython-36.pyc
|-- wsgi.cpython-36.pyc
|-- __init__.cpython-36.pyc
```

```
C:\Windows\system32\cmd.exe

C:\Users\WGoReb\work_django\mysite>python manage.py startapp polls

C:\Users\WGoReb\work_django\mysite>tree /f
폴더 PATH의 목록입니다.
볼륨 일련 번호가 00000030 3224:8106입니다.
C:
|-- db.sqlite3
|-- manage.py
|-- mysite
|   |-- settings.py
|   |-- urls.py
|   |-- wsgi.py
|   |-- __init__.py
|   |-- __pycache__
|   |-- settings.cpython-36.pyc
|   |-- urls.cpython-36.pyc
|   |-- wsgi.cpython-36.pyc
|   |-- __init__.cpython-36.pyc
|-- __pycache__
|-- settings.cpython-36.pyc
|-- urls.cpython-36.pyc
|-- wsgi.cpython-36.pyc
|-- __init__.cpython-36.pyc
|-- polls
|   |-- admin.py
|   |-- apps.py
|   |-- models.py
|   |-- tests.py
|   |-- views.py
|   |-- __init__.py
|-- migrations
|   |-- __init__.py
```

■ 장고 튜토리얼 투표앱

● 뷰 작성

– polls/views.py

```
from django.http import HttpResponseRedirect

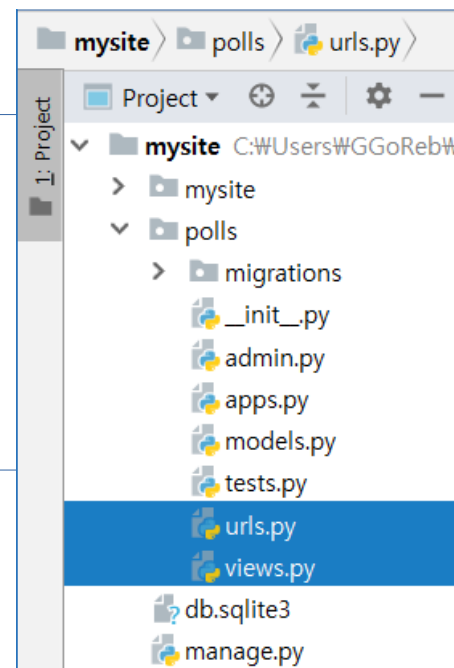
def index(request):
    return HttpResponseRedirect("Hello, world. You're at the polls index.")
```

● 접속 주소(url) 생성

– polls/urls.py

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
]
```



■ 장고 튜토리얼 투표앱

● 접속 주소(url) 생성

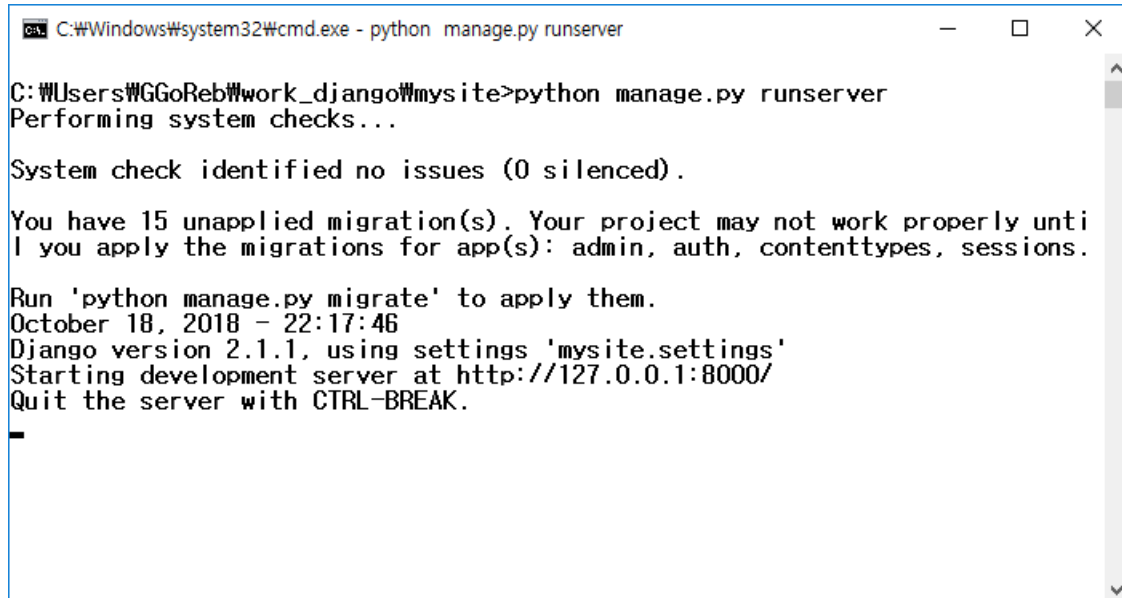
– mysite/urls.py

```
from django.http import HttpResponseRedirect

def index(request):
    return HttpResponseRedirect("Hello, world. You're at the polls index.")
```

● 서버 구동

– python manage.py runserver



```
C:\Windows\system32\cmd.exe - python manage.py runserver

C:\Users\WGoReb\work_django\mysite>python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).

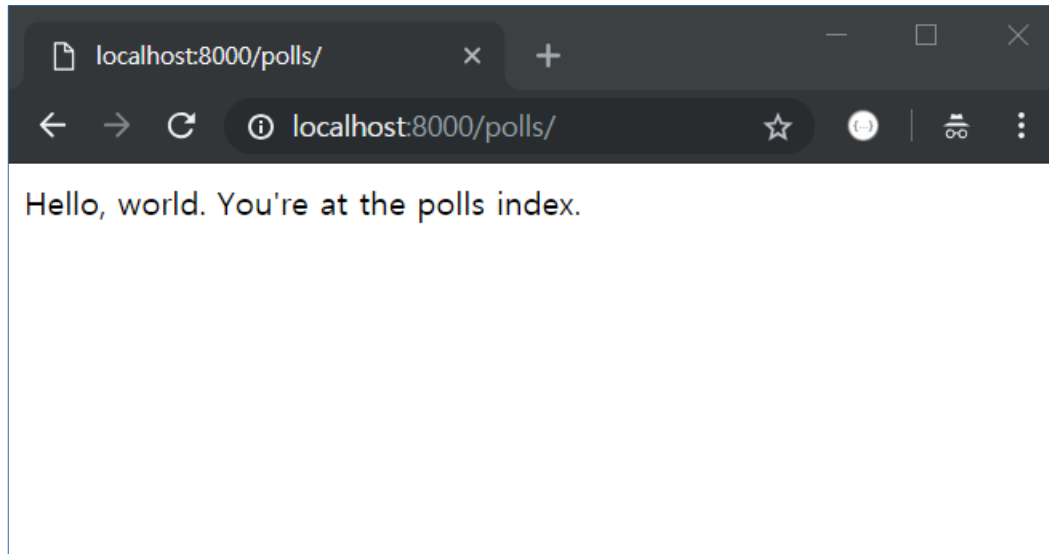
You have 15 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.

Run 'python manage.py migrate' to apply them.
October 18, 2018 - 22:17:46
Django version 2.1.1, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
_
```

■ 장고 튜토리얼 투표앱

● 프로젝트 구동 확인

– <http://localhost:8000/polls> (<http://127.0.0.1:8000/polls>)



■ 장고 튜토리얼 투표앱

● 모델 작성 - 데이터베이스 활용을 위해

- polls/models.py

```
from django.db import models

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

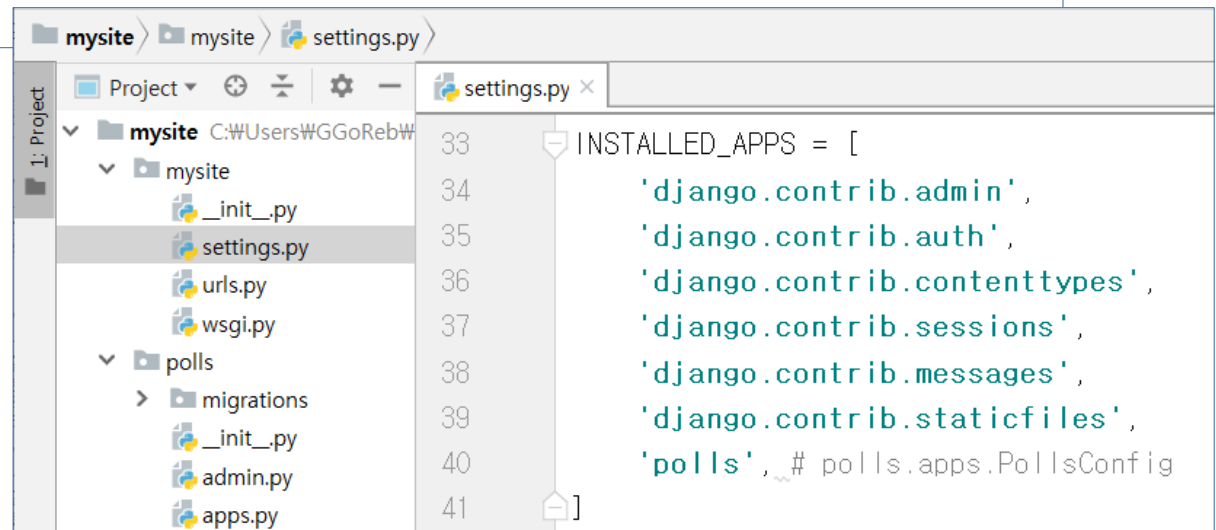
class Choice(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)
```

■ 장고 튜토리얼 투표앱

● 작성된 모델 활성화 (프로젝트 등록)

– mysite/settings.py

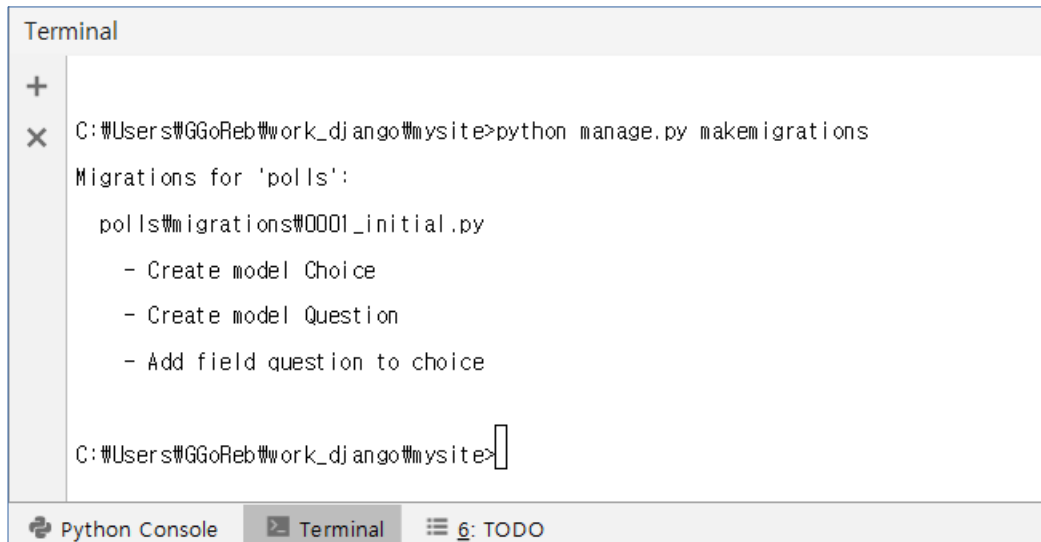
```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'polls', # polls.apps.PollsConfig  
]
```



■ 장고 튜토리얼 투표앱

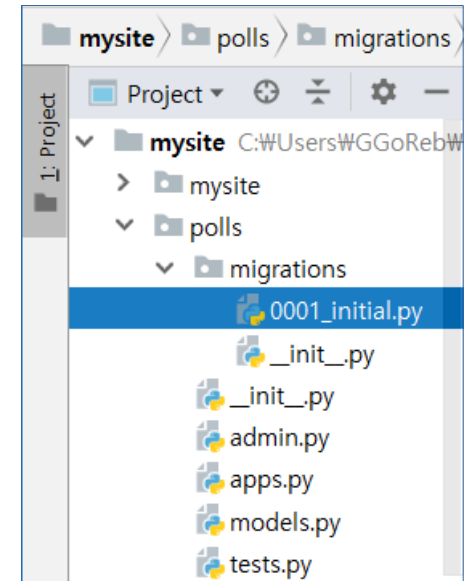
● 모델의 변경사항(추가/수정/삭제) 파일 생성

– python manage.py makemigrations



```
Terminal
+
X C:\Users\GGoReb\work_django\mysite>python manage.py makemigrations
Migrations for 'polls':
  polls\migrations\0001_initial.py
    - Create model Choice
    - Create model Question
    - Add field question to choice

C:\Users\GGoReb\work_django\mysite>
```



■ 장고 튜토리얼 투표앱

● 모델의 변경사항(추가/수정/삭제) 데이터베이스 적용

– python manage.py migrate

```
Terminal
+
X C:\Users\GGoreb\work_django\mysite>python manage.py migrate

Operations to perform:
  Apply all migrations: admin, auth, contenttypes, polls, sessions

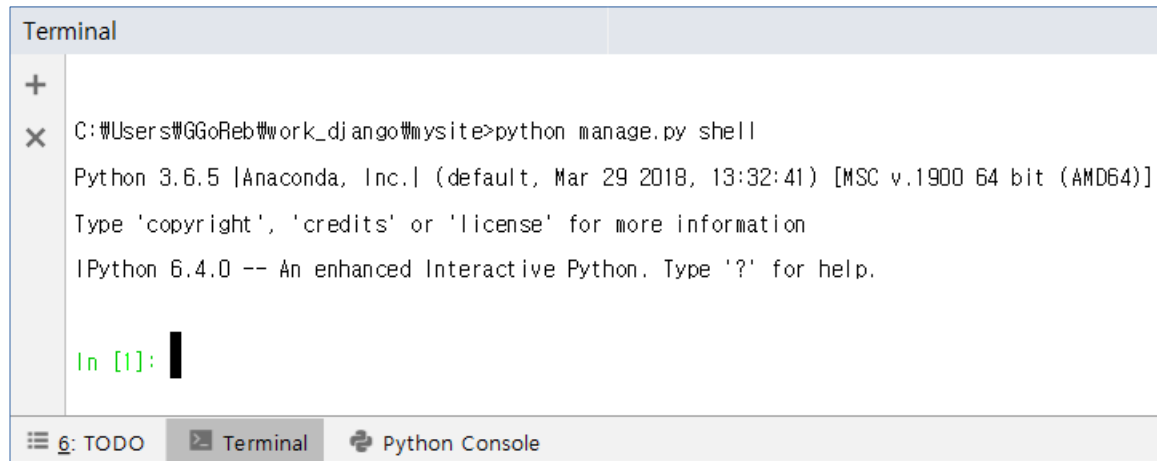
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
```

테이블 (13)	
>	auth_group
>	auth_group_permissions
>	auth_permission
>	auth_user
>	auth_user_groups
>	auth_user_user_permissions
>	django_admin_log
>	django_content_type
>	django_migrations
>	django_session
>	polls_choice
>	polls_question
>	sqlite_sequence

■ 장고 튜토리얼 투표앱

● 데이터베이스 관련 API 테스트

– shell 접속 : `python manage.py shell`



```
Terminal
+
x C:\Users\GGoreb\work_django\mysite>python manage.py shell
Python 3.6.5 [Anaconda, Inc.] (default, Mar 29 2018, 13:32:41) [MSC v.1900 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 6.4.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]:
```

■ 장고 튜토리얼 투표앱

● 데이터베이스 관련 API 테스트

- 모델(Choice, Question) import : `from polls.models import *`
- Question 테이블 데이터 조회 : `Question.objects.all()`

```
Terminal
+
x In [1]: from polls.models import *

In [2]: Question.objects.all()
Out[2]: <QuerySet []>

In [3]:
```

6: TODO Terminal Python Console

Choice / Question 테이블 스키마

polls_choice	
id	integer
choice_text	varchar(200)
votes	integer
question_id	integer
polls_question	
id	integer
question_text	varchar(200)
pub_date	datetime

■ 장고 튜토리얼 투표앱

● 데이터베이스 관련 API 테스트

- Question 데이터 입력 : Question(컬럼=값)
- 데이터 저장 : save()

```
Terminal
+
x In [3]: from django.utils import timezone

In [4]: q = Question(question_text="What's new?", pub_date=timezone.now())

In [5]: q.save()
```

6: TODO Terminal Python Console

	id	question_text	pub_date
	필터	필터	필터
1	1	What's new?	2018-10-19 00:01:22.65153

Choice / Question 테이블 스키마

polls_choice	
id	integer
choice_text	varchar(200)
votes	integer
question_id	integer
polls_question	
id	integer
question_text	varchar(200)
pub_date	datetime

■ 장고 튜토리얼 투표앱

● 데이터베이스 관련 API 테스트

– Question 데이터 확인

```
Terminal
+
x In [6]: print(q.id, q.question_text, q.pub_date)
1 What's new? 2018-10-19 00:01:22.651530+00:00

In [7]: Question.objects.all()
Out [7]: <QuerySet [<Question: Question object (1)>]>
```

– Question 데이터 수정 : 기존 객체의 속성 값을 수정한 후 save()

```
Terminal
+
x In [8]: q.question_text = "What's up?"

In [9]: q.save()

In [10]: []
```

	id	question_text	pub_date
	필터	필터	필터
1	1	What's up?	2018-10-19 00:01:22.651530

■ 장고 튜토리얼 투표앱

● 데이터베이스 관련 API 테스트

– 데이터 출력 형식 변경 : `__str__` 오버라이드

```
from django.db import models

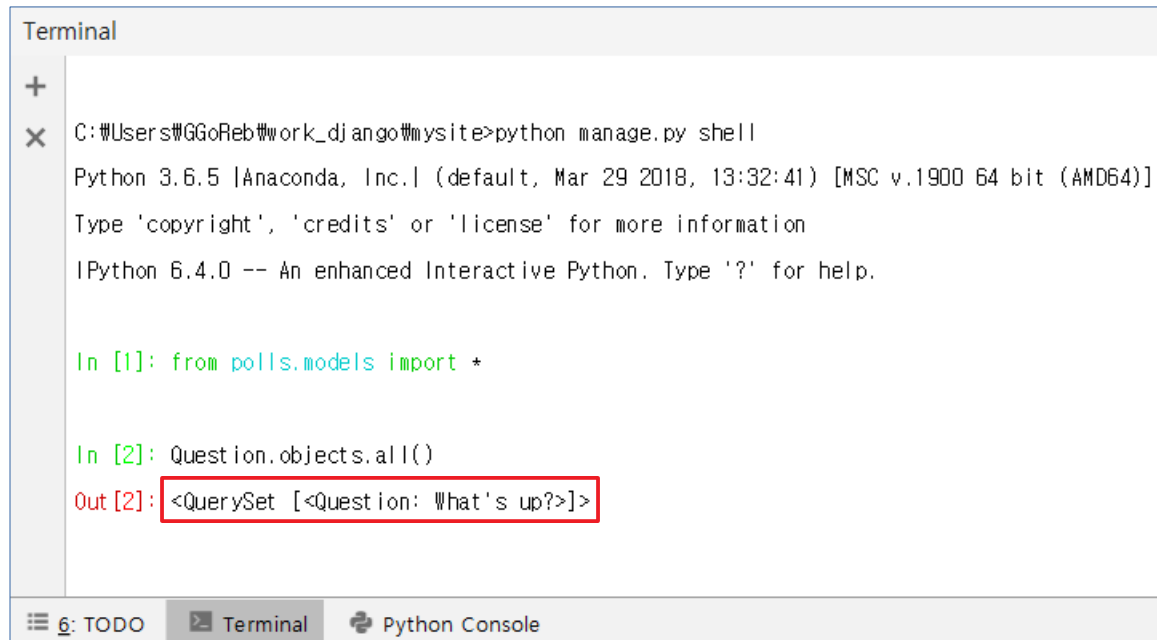
class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')

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

■ 장고 튜토리얼 투표앱

● 데이터베이스 관련 API 테스트

– shell 재실행 후 데이터 확인



The screenshot shows a terminal window titled "Terminal" with a light gray background. On the left side of the terminal, there are icons for a plus sign (+) and a close sign (X). The terminal content shows the following sequence of commands and output:

```
C:\Users\GGoreb\work_django\mysite>python manage.py shell
Python 3.6.5 [Anaconda, Inc.] (default, Mar 29 2018, 13:32:41) [MSC v.1900 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 6.4.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: from polls.models import *

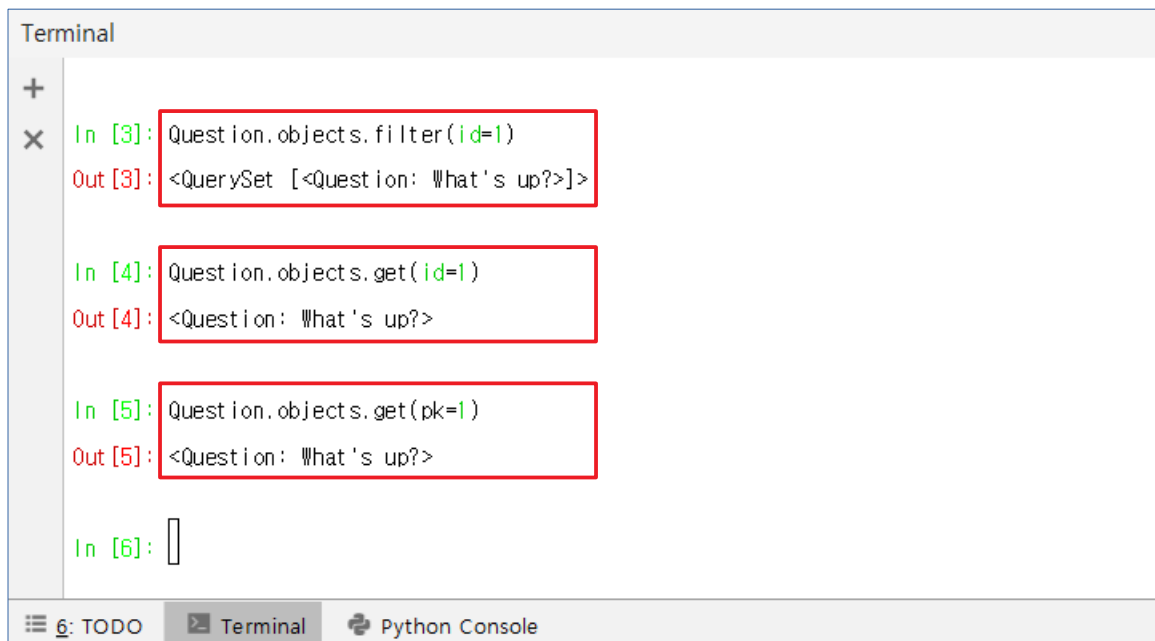
In [2]: Question.objects.all()
Out [2]: <QuerySet [<Question: What's up?>]>
```

At the bottom of the terminal window, there is a tab bar with three tabs: "6: TODO", "Terminal" (which is the active tab), and "Python Console".

■ 장고 튜토리얼 투표앱

● 데이터베이스 관련 API 테스트

– id 속성이 1인 데이터 조회



The screenshot shows a Jupyter Notebook interface with a terminal window. The terminal displays three Django ORM queries and their results, each highlighted with a red box. The first query filters for a question with id=1, returning a queryset. The second and third queries retrieve the specific question object using id and pk respectively, both returning the same object: <Question: What's up?>. The fourth input line is empty.

```
Terminal
+
x In [3]: Question.objects.filter(id=1)
Out [3]: <QuerySet [<Question: What's up?>]>

In [4]: Question.objects.get(id=1)
Out [4]: <Question: What's up?>

In [5]: Question.objects.get(pk=1)
Out [5]: <Question: What's up?>

In [6]:
```

6: TODO Terminal Python Console

■ 장고 튜토리얼 투표앱

● 데이터베이스 관련 API 테스트

- question_text 속성의 값이 'What'으로 시작하는 데이터 조회

```
Terminal
+ In [6]: Question.objects.filter(question_text__startswith='What')
x Out [6]: <QuerySet [<Question: What's up?>]>

In [7]:
```

6: TODO Terminal Python Console

- pub_date (작성일자) 속성의 값이 현재 연도와 같은 데이터 조회

```
Terminal
+
x In [7]: from django.utils import timezone

In [8]: current_year = timezone.now().year

In [9]: Question.objects.get(pub_date__year=current_year)
Out [9]: <Question: What's up?>

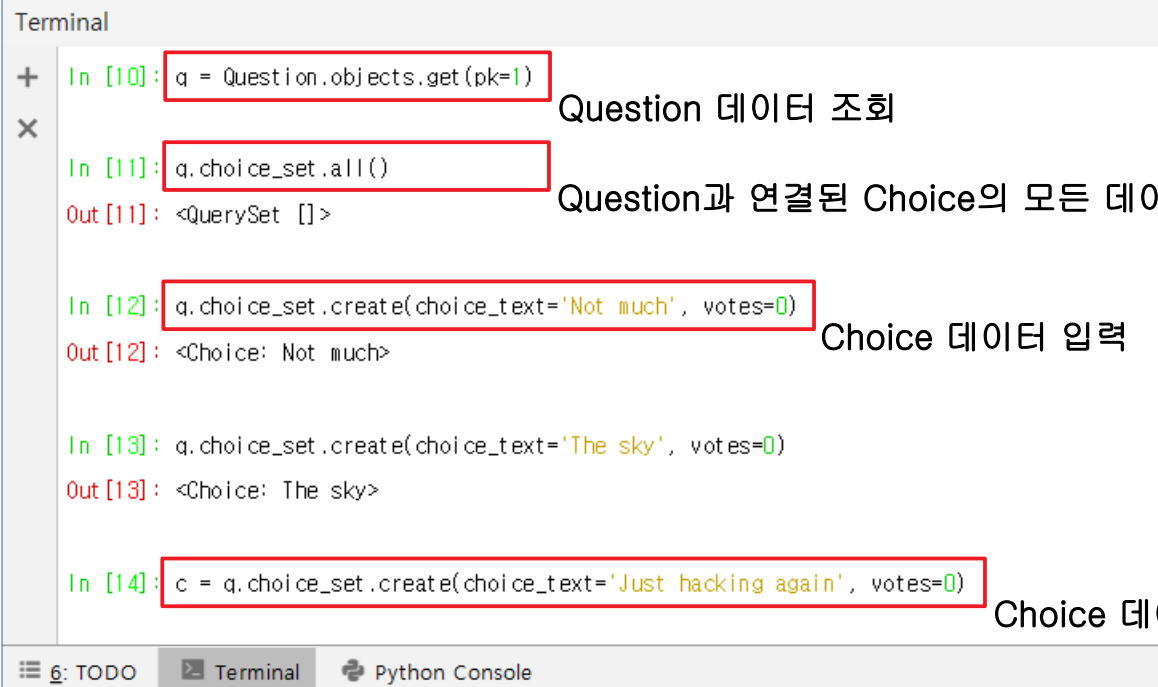
In [10]:
```

6: TODO Terminal Python Console

■ 장고 튜토리얼 투표앱

● 데이터베이스 관련 API 테스트

– Question과 연결되어 있는 Choice의 데이터 조회 / 입력



The screenshot shows a terminal window with the following content:

```
Terminal
+ In [10]: q = Question.objects.get(pk=1)
x In [11]: q.choice_set.all()
Out [11]: <QuerySet []>
In [12]: q.choice_set.create(choice_text='Not much', votes=0)
Out [12]: <Choice: Not much>
In [13]: q.choice_set.create(choice_text='The sky', votes=0)
Out [13]: <Choice: The sky>
In [14]: c = q.choice_set.create(choice_text='Just hacking again', votes=0)
```

Annotations on the right side of the terminal window:

- Question 데이터 조회 (next to In [10])
- Question과 연결된 Choice의 모든 데이터 조회 (next to In [11])
- Choice 데이터 입력 (next to In [12])
- Choice 데이터 입력 후 객체 저장 (next to In [14])

The terminal window has a tab bar at the bottom with three tabs: "6: TODO", "Terminal", and "Python Console". The "Terminal" tab is currently selected.

■ 장고 튜토리얼 투표앱

● 데이터베이스 관련 API 테스트

– Choice 객체를 이용한 조회

```
Terminal
+ In [18]: c.question Choice와 연결된 Question 데이터 조회
x Out [18]: <Question: What's up?>

In [19]: q.choice_set.all() Question과 연결된 Choice의 모든 데이터 조회
Out [19]: <QuerySet [<Choice: Not much>, <Choice: The sky>, <Choice: Just hacking again>]>

In [20]: q.choice_set.count() Question과 연결된 Choice의 개수 조회
Out [20]: 3

In [21]: Choice.objects.filter(question__pub_date__year=current_year) Question의 작성일자로 Choice 데이터 조회
Out [21]: <QuerySet [<Choice: Not much>, <Choice: The sky>, <Choice: Just hacking again>]>

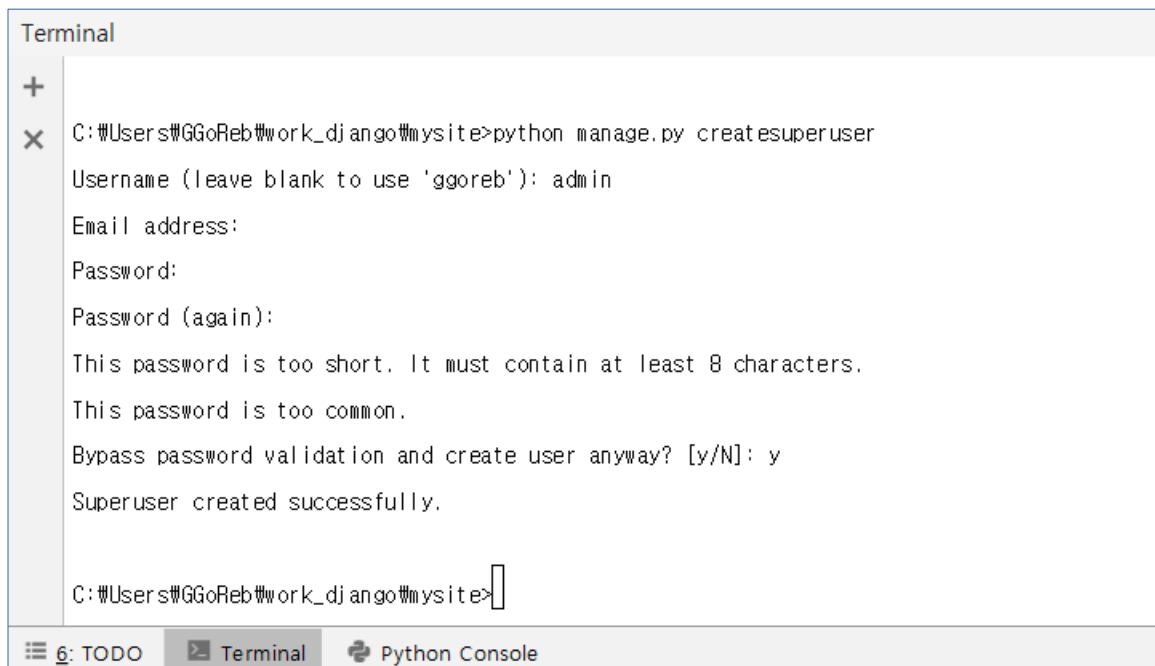
In [22]: c = q.choice_set.filter(choice_text__startswith='Just')
In [23]: c 'Just'로 시작하는 Choice 데이터 조회
Out [23]: <QuerySet [<Choice: Just hacking again>]>
```

6: TODO Terminal Python Console

■ 장고 튜토리얼 투표앱

● 데이터 관리를 위한 관리자 생성

– `python manage.py createsuperuser`



```
Terminal
+
x C:\Users\GGoreb\work_django\mysite>python manage.py createsuperuser
Username (leave blank to use 'ggoreb'): admin
Email address:
Password:
Password (again):
This password is too short. It must contain at least 8 characters.
This password is too common.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.

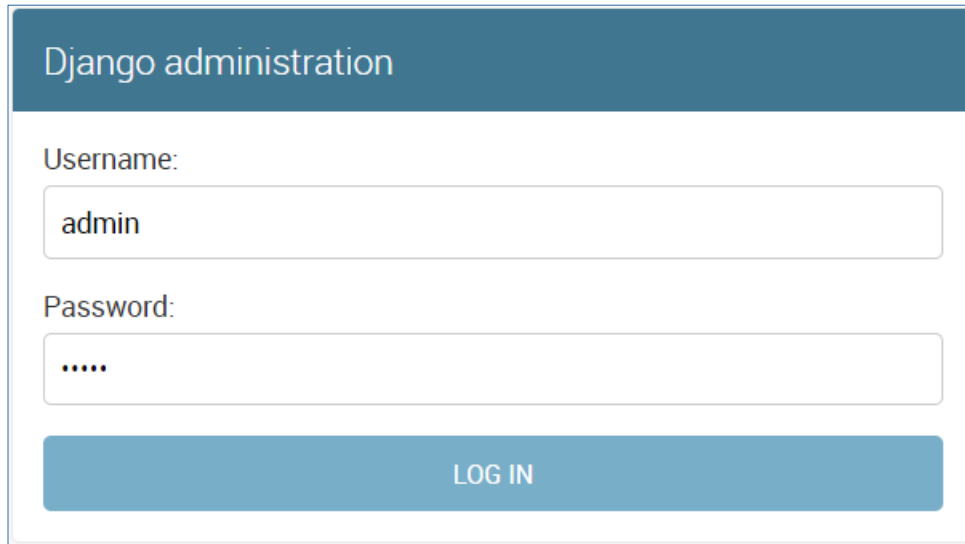
C:\Users\GGoreb\work_django\mysite>
```

6: TODO Terminal Python Console

■ 장고 튜토리얼 투표앱

● 관리자 사이트 접속

– <http://localhost:8000/admin> (<http://127.0.0.1:8000/admin>)



The image shows the Django administration login interface. It features a dark blue header with the text "Django administration". Below the header, there are two input fields: "Username:" with the value "admin" and "Password:" with masked characters ".....". At the bottom, there is a blue button labeled "LOG IN".

Django administration

Username:
admin

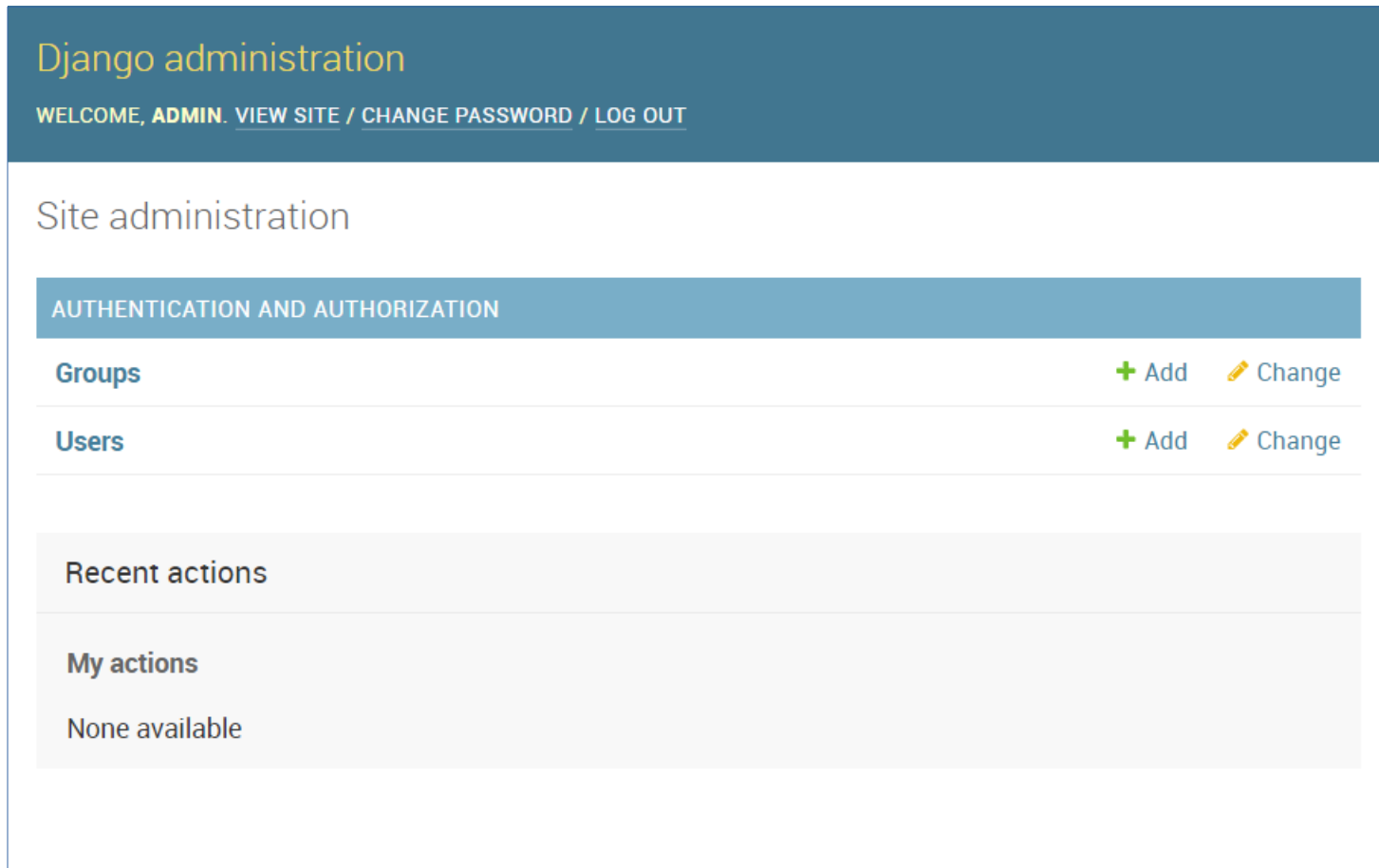
Password:
.....

LOG IN

■ 장고 튜토리얼 투표앱

● 관리자 사이트 접속

– <http://localhost:8000/admin> (<http://127.0.0.1:8000/admin>)



The screenshot displays the Django administration interface. At the top, a dark blue header contains the text "Django administration" in yellow, followed by "WELCOME, ADMIN." and links for "VIEW SITE", "CHANGE PASSWORD", and "LOG OUT". Below the header, the main content area is titled "Site administration". A light blue bar highlights the "AUTHENTICATION AND AUTHORIZATION" section. Under this section, there are two rows: "Groups" and "Users". Each row has a green plus icon and the word "Add" followed by a yellow pencil icon and the word "Change". Below these rows, there are two light gray boxes. The first box is titled "Recent actions" and the second box is titled "My actions". Both boxes contain the text "None available".

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

■ 장고 튜토리얼 투표앱

● 관리자 사이트에서 데이터를 관리할 수 있도록 등록

– polls/admin.py

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

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

Django administration

WELCOME, **ADMIN**. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups

[+ Add](#) [Change](#)

Users

[+ Add](#) [Change](#)

POLLS

Choices

[+ Add](#) [Change](#)

Questions

[+ Add](#) [Change](#)

Recent actions

My actions

None available

■ 장고 튜토리얼 투표앱

● 질문 / 결과 / 투표 페이지 (view) 작성

– polls/views.py

```
def detail(request, question_id): # 질문 상세 페이지
    return HttpResponse("You're looking at question %s." % question_id)

def results(request, question_id): # 투표 결과 페이지
    response = "You're looking at the results of question %s."
    return HttpResponse(response % question_id)

def vote(request, question_id): # 투표 페이지
    return HttpResponse("You're voting on question %s." % question_id)
```


■ 장고 튜토리얼 투표앱

● 질문 / 결과 / 투표 페이지 (view) 작성

– polls/urls.py

```
from django.urls import path
from . import views

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

polls/1/
polls/100/

polls/1/results/
polls/100/results/

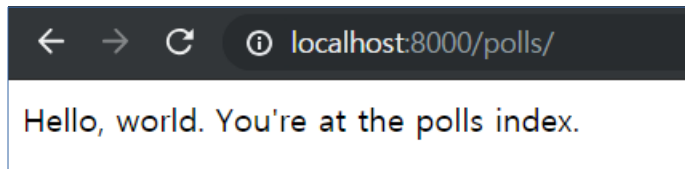
polls/1/vote/
polls/100/vote/

```
def detail(request, question_id):
    ...
def results(request, question_id):
    ...
def vote(request, question_id):
    ...
```

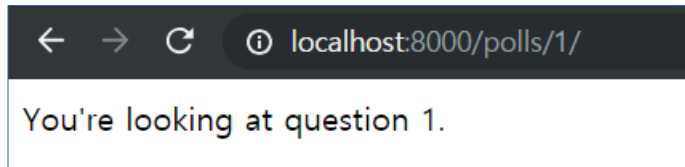
■ 장고 튜토리얼 투표앱

● 질문 / 결과 / 투표 페이지 확인

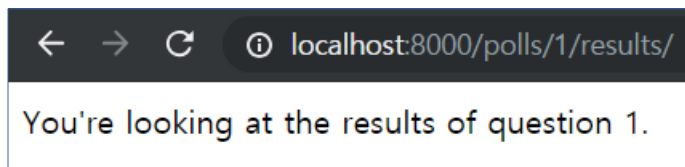
– <http://localhost:8000/polls/>



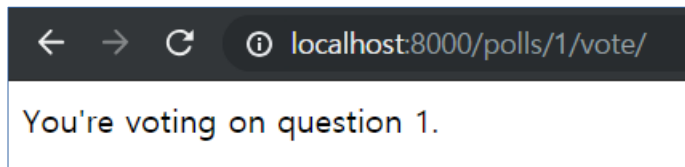
– <http://localhost:8000/polls/1/>



– <http://localhost:8000/polls/1/results/>



– <http://localhost:8000/polls/1/vote/>

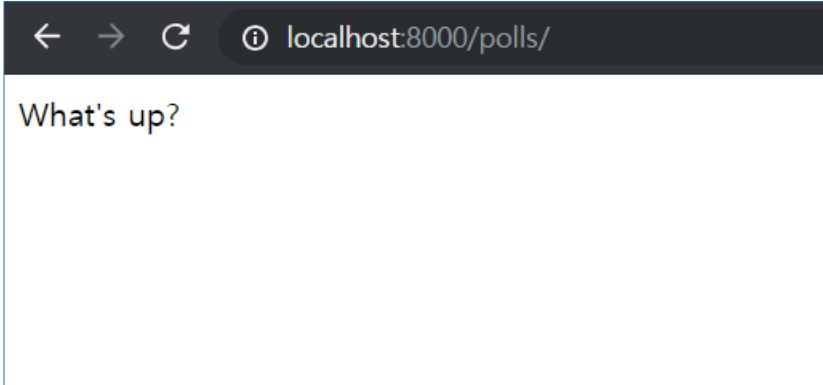


■ 장고 튜토리얼 투표앱

● 모델을 활용하여 데이터 출력

– polls/views.py

```
def index(request):  
    latest_question_list = Question.objects.order_by('-pub_date')[:5]  
    output = ', '.join([q.question_text for q in latest_question_list])  
    # return HttpResponse("Hello, world. You're at the polls index.")  
    return HttpResponse(output)
```



← → ↻ ⓘ localhost:8000/polls/

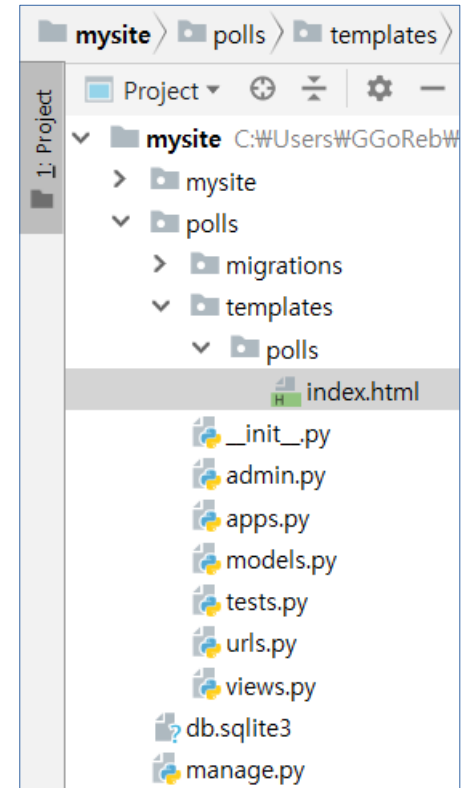
What's up?

■ 장고 튜토리얼 투표앱

● 템플릿 활용

- app/templates/app 구조로 디렉토리 생성
- app/views.py 에서 불러올 html 파일 작성
- 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 %}
```



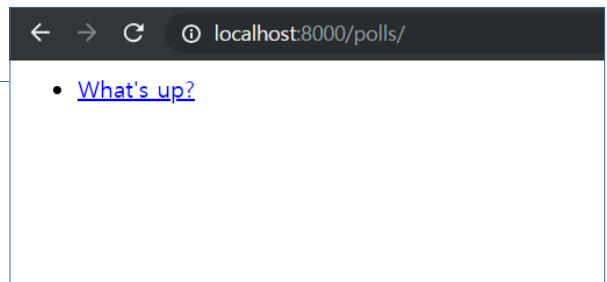
■ 장고 튜토리얼 투표앱

● 템플릿 활용

- app/templates/app 구조로 디렉토리 생성
- app/views.py 에서 불러올 html 파일 작성
- polls/view.py

```
from django.http import HttpResponse
from django.template import loader
from .models import Question
```

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



■ 장고 튜토리얼 투표앱

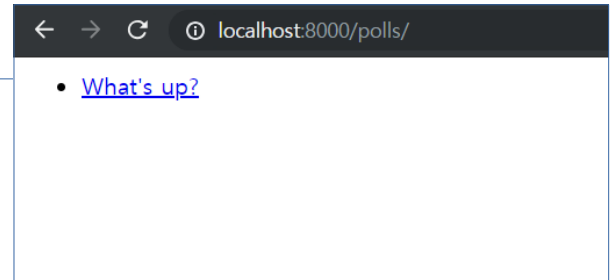
● 템플릿 활용

– render() 함수

– polls/view.py

```
from django.shortcuts import render
from .models import Question
```

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



■ 장고 튜토리얼 투표앱

● 모델을 활용하여 질문 상세 페이지 데이터 출력

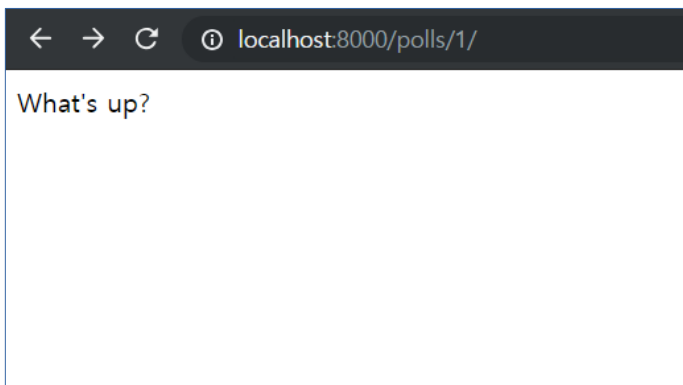
– polls/view.py

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

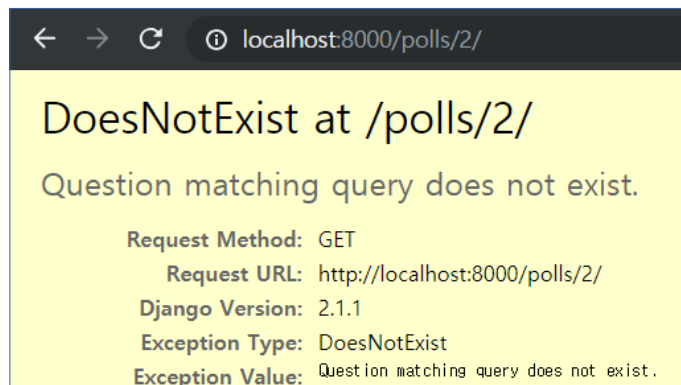
– polls/templates/polls/detail.html

```
{{ question }}
```

1번 질문 조회



2번 질문 조회



■ 장고 튜토리얼 투표앱

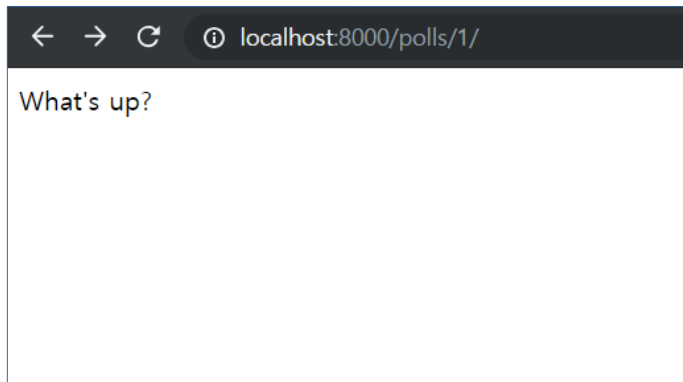
● DoesNotExist 예외 처리 (404)

– polls/view.py

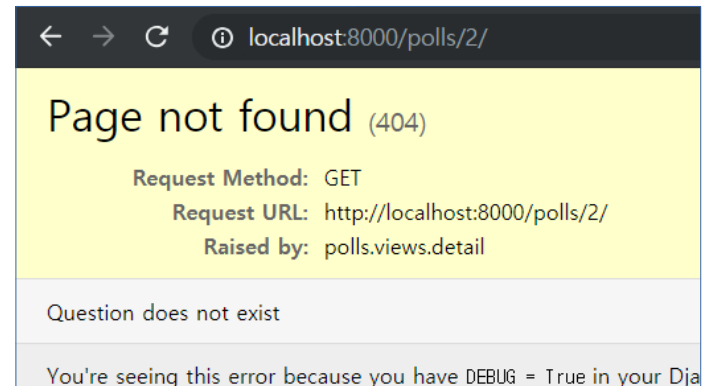
```
from django.http import HttpResponseRedirect, Http404

def detail(request, question_id):
    try:
        question = Question.objects.get(pk=question_id)
    except Question.DoesNotExist:
        raise Http404("Question does not exist")
    return render(request, 'polls/detail.html', {'question': question})
```

1번 질문 조회



2번 질문 조회



■ 장고 튜토리얼 투표앱

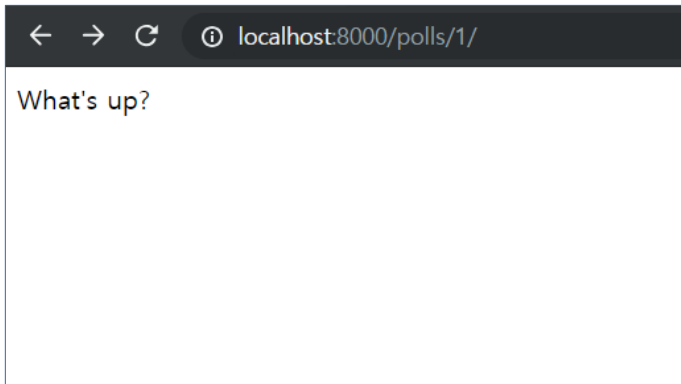
● DoesNotExist 예외 처리 (404)

- get_object_or_404() 함수 활용 (≡ get_list_or_404)
- polls/view.py

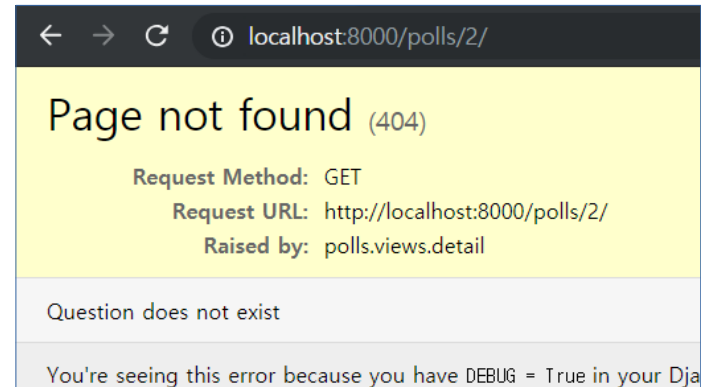
```
from django.shortcuts import render, get_object_or_404

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

1번 질문 조회



2번 질문 조회



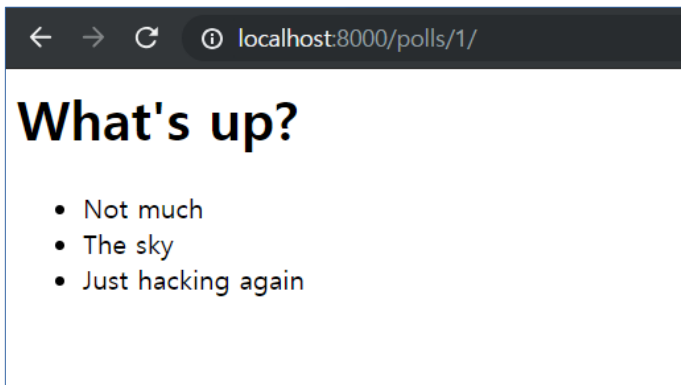
■ 장고 튜토리얼 투표앱

● 템플릿 태그 활용

– polls/templates/polls/detail.html

```
<h1>{{ question.question_text }}</h1>
<ul>
{% for choice in question.choice_set.all %}
    <li>{{ choice.choice_text }}</li>
{% endfor %}
</ul>
```

1번 질문 조회



■ 장고 튜토리얼 투표앱

● 하드코딩된 URL 변경

– polls/templates/polls/index.html

```
<li><a href="/polls/{{ question.id }}">{{ question.question_text }}</a></li>
```



```
<li><a href="{% url 'detail' question.id %}">{{ question.question_text }}</a></li>
```

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

■ 장고 튜토리얼 투표앱

● URL namespace 적용

– polls/urls.py

```
from django.urls import path
from . import views

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

– polls/templates/polls/index.html

```
<a href="{% url 'detail' question.id %}">
```




```
<a href="{% url 'polls:detail' question.id %}">
```

■ 장고 튜토리얼 투표앱

● 폼 만들기

– polls/templates/polls/detail.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>
```



← → ↻ ⓘ localhost:8000/polls/1/

What's up?

☐ Not much

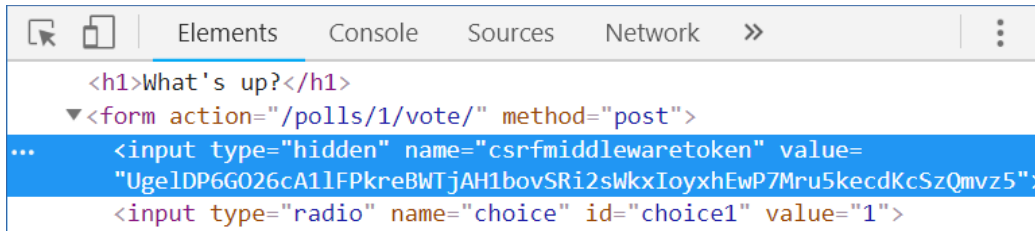
☐ The sky

☐ Just hacking again

■ 장고 튜토리얼 투표앱

● CSRF (Crose Site Request Forgery)

- 사이트 간 요청 위조
- `http://ggoreb.com/article/1` 과 같은 일정 패턴의 주소를 분석/변경하여 일반적 방법으로 접근 할 수 없는 페이지에 접근하는 등 특정 웹페이지의 취약점을 공격하는 방법
- referer (요청 전 페이지) 정보를 확인하거나 임의의 난수(토큰)를 발급하여 정상적인 접근인지 확인 가능
- 장고에서는 템플릿의 태그를 이용하여 토큰 제공
`{% csrf_token%}`



```
<h1>What's up?</h1>
▼<form action="/polls/1/vote/" method="post">
...   <input type="hidden" name="csrfmiddlewaretoken" value=
      "Uge1DP6G026cA11FPkreBWTjAH1bovSRi2sWkxIoyxhEwP7Mru5kecdKcSzQmvz5">
   <input type="radio" name="choice" id="choice1" value="1">
```

■ 장고 튜토리얼 투표앱

● 투표 기능 및 페이지 수정

– polls/view.py

```
from django.http import HttpResponseRedirect
from django.urls import reverse
from .models import Question, Choice

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):
        return render(request, 'polls/detail.html', {
            'question': question,
            'error_message': "You didn't select a choice.",
        })
    else:
        selected_choice.votes += 1
        selected_choice.save()
        return HttpResponseRedirect(reverse('polls:results', args=(question.id,)))
```

장고 튜토리얼 투표앱

투표 기능 확인

← → ↻ ⓘ localhost:8000/polls/1/

What's up?

- ☐ Not much
- ☐ The sky
- ☐ Just hacking again

Vote

답변 선택

답변 미선택

← → ↻ ⓘ localhost:8000/polls/1/vote/

What's up?

You didn't select a choice.

- ☐ Not much
- ☐ The sky
- ☐ Just hacking again

Vote

← → ↻ ⓘ localhost:8000/polls/1/results/

You're looking at the results of question 1.

	id	choice_text	votes	question_id
	필터	필터	필터	필터
1	1	Not much	0	1
2	2	The sky	0	1
3	3	Just hacking agl	1	1

■ 장고 튜토리얼 투표앱

● 투표 결과 데이터 조회 및 페이지 수정

– polls/view.py

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

– polls/templates/polls/results.html

```
<h1>{{ question.question_text }}</h1>
<ul>
{% for choice in question.choice_set.all %}
    <li>
        {{ choice.choice_text }}
        -- {{ choice.votes }}
        vote{{ choice.votes|pluralize }} <!-- 1보다 크면 's' 붙임 -->
    </li>
{% endfor %}
</ul>
<a href="{% url 'polls:detail' question.id %}">Vote again?</a>
```

■ 장고 튜토리얼 투표앱

● 투표 결과 확인



■ 장고 튜토리얼 투표앱

● 제네릭 뷰 (Class-based View)

– polls/view.py – index

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



```
class IndexView(generic.ListView):  
    template_name = 'polls/index.html'  
  
    context_object_name = 'latest_question_list'  
    def get_queryset(self):  
        return Question.objects.order_by('-pub_date')[:5]
```

■ 장고 튜토리얼 투표앱

● 제네릭 뷰 (Class-based View)

– polls/urls.py – index

```
from django.urls import path
from . import views

app_name = 'polls'
urlpatterns = [
    # path('', views.index, name='index'),
    path('', views.IndexView.as_view(), name='index'),

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

■ 장고 튜토리얼 투표앱

● 제네릭 뷰 (Class-based View)

– polls/view.py – detail

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



```
class DetailView(generic.DetailView):  
    model = Question  
    template_name = 'polls/detail.html'
```

■ 장고 튜토리얼 투표앱

● 제네릭 뷰 (Class-based View)

– polls/urls.py – detail

```
from django.urls import path
from . import views

app_name = 'polls'
urlpatterns = [
    path('', views.IndexView.as_view(), name='index'),

    # path('<int:question_id>/', views.detail, name='detail'),
    path('<int:pk>/', views.DetailView.as_view(), name='detail'),

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

■ 장고 튜토리얼 투표앱

● 제네릭 뷰 (Class-based View)

– polls/view.py – results

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



```
class ResultsView(generic.DetailView):  
    model = Question  
    template_name = 'polls/results.html'
```

■ 장고 튜토리얼 투표앱

● 제네릭 뷰 (Class-based View)

– polls/urls.py – results

```
from django.urls import path
from . import views

app_name = 'polls'
urlpatterns = [
    path('', views.IndexView.as_view(), name='index'),

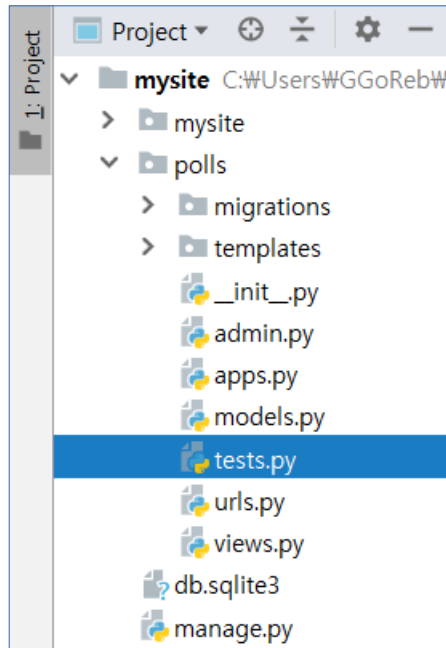
    # path('<int:question_id>/', views.detail, name='detail'),
    path('<int:pk>/', views.DetailView.as_view(), name='detail'),

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


■ 장고 튜토리얼 투표앱

● 테스트

- tests.py 를 통해 자동화 테스트 지원
- 특정 시점까지는 '제대로 작동하는지 확인' 하는 것만으로도 충분하지만
서비스의 규모가 커지고 로직이 복잡해지면서 기능끼리 상호 작용을 하면
'제대로 작동하는지 확인' 하기 위해 많은 시간이 소요됨
- 자동화 테스트를 통해 시간 절약 가능



■ 장고 튜토리얼 투표앱

● 테스트

- 질문 등록 날짜가 최신(하루 1일 이내)인지 판단하는 함수
- polls/models.py

```
import datetime
from django.utils import timezone

class Question(models.Model):
    ...

    def was_published_recently(self):
        return self.pub_date >= timezone.now() - datetime.timedelta(days=1)
```

■ 장고 튜토리얼 투표앱

● 테스트

– polls/tests.py

```
from django.test import TestCase
import datetime
from django.utils import timezone
from .models import Question
```

```
class QuestionModelTests(TestCase):
```

```
    def test_was_published_recently_with_future_question(self):
```

```
        time = timezone.now() + datetime.timedelta(days=30) # 30일 후
```

```
        future_question = Question(pub_date=time) # 질문 등록
```

```
        self.assertIs(future_question.was_published_recently(), False)
```

■ 장고 튜토리얼 투표앱

● 테스트

- 테스트 수행 결과 : `python manage.py test polls`

```
Terminal
+ C:\Users\GGGoReb\work_django\mysite>python manage.py test polls
x Creating test database for alias 'default'...
System check identified no issues (0 silenced).
F
=====
FAIL: test_was_published_recently_with_future_question (polls.tests.QuestionModelTests)
-----
Traceback (most recent call last):
  File "C:\Users\GGGoReb\work_django\mysite\polls\tests.py", line 11, in test_was_published_recently_with_future_question
    self.assertIs(future_question.was_published_recently(), False)
AssertionError: True is not False
-----
Ran 1 test in 0.001s
```

False 인 경우

```
Terminal
+ C:\Users\GGGoReb\work_django\mysite>python manage.py test polls
x Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.
-----
Ran 1 test in 0.001s
```

True 인 경우

■ 장고 튜토리얼 투표앱

● 테스트

- 버그 수정
- polls/models.py

```
import datetime
from django.utils import timezone

class Question(models.Model):
    ...

    def was_published_recently(self):
        now = timezone.now()
        return now - datetime.timedelta(days=1) <= self.pub_date <= now
```

■ 장고 튜토리얼 투표앱

● 테스트

- 다양한 테스트 추가
- polls/tests.py

```
...
class QuestionModelTests(TestCase):
    ...
    def test_was_published_recently_with_old_question(self): # 1일 1초 전
        time = timezone.now() - datetime.timedelta(days=1, seconds=1)
        old_question = Question(pub_date=time)
        self.assertIs(old_question.was_published_recently(), False)

    def test_was_published_recently_with_recent_question(self): # 23시 59분 59초 전
        time = timezone.now() - datetime.timedelta(hours=23, minutes=59, seconds=59)
        recent_question = Question(pub_date=time)
        self.assertIs(recent_question.was_published_recently(), True)
```

■ 장고 튜토리얼 투표앱

● 웹 자원(css, javascript 등) 활용

– polls/static/polls/style.css

```
li a {  
    color: green;  
}
```

– polls/templates/polls/index.html

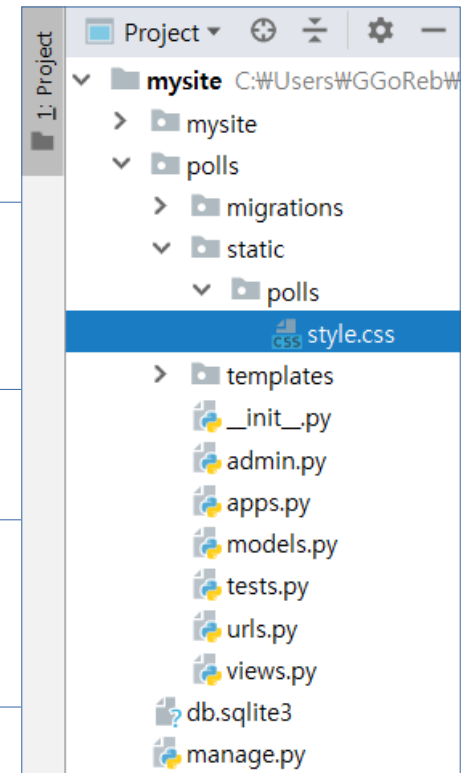
```
{% load static %}  
<link rel="stylesheet" href="{% static 'polls/style.css' %}">  
...
```

– mysite/settings.py

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



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



■ 관리자 사이트 커스터마이징

● 질문 등록 필드 순서 변경

– polls/admin.py

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

class QuestionAdmin(admin.ModelAdmin):
    fields = ['pub_date', 'question_text']


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


Home > Polls > Questions > What's up?

Change question

Question text:

Date published:

Date: Today | 

Time: Now | 


Note: You are 9 hours ahead of server time.




Home > Polls > Questions > What's up?

Change question

Date published:

Date: Today | 

Time: Now | 

Note: You are 9 hours ahead of server time.

Question text:

■ 관리자 사이트 커스터마이징

● 필드 항목 구분

– polls/admin.py

```
class QuestionAdmin(admin.ModelAdmin):  
    fieldsets = [  
        (None, {'fields': ['question_text']}),  
        ('Date information', {'fields': ['pub_date']}),  
    ]
```

Home › Polls › Questions › What's up?

Change question

Date published:

Date: Today |

Time: Now |

Note: You are 9 hours ahead of server time.

Question text:



Home › Polls › Questions › What's up?

Change question

Question text:

Date information

Date published:

Date: Today |

Time: Now |

Note: You are 9 hours ahead of server time.

■ 관리자 사이트 커스터마이징

● 질문 등록 페이지에서 보기 등록 기능 추가

– polls/admin.py

```
class ChoiceInline(admin.StackedInline):
    model = Choice
    extra = 3

class QuestionAdmin(admin.ModelAdmin):
    fieldsets = [
        (None, {'fields': ['question_text']}),
        ('Date information', {'fields': ['pub_date']}),
    ]
    inlines = [ChoiceInline]

admin.site.register(Question, QuestionAdmin)
```

The screenshot shows the Django Admin interface for adding a new question. The breadcrumb trail at the top is "Home > Polls > Questions > Add question". The main heading is "Add question". Below this, there is a "Question text:" label followed by a text input field. A section titled "Date information" contains "Date published:" with "Date:" and "Time:" input fields, and "Today" and "Now" buttons with calendar and clock icons. A note below states "Note: You are 9 hours ahead of server time." Below the date section, there is a "CHOICES" section. This section contains three choice entries, each with a "Choice text:" label and a text input field, and a "Votes:" label with a numeric input field showing "0". The "CHOICES" section is highlighted with a red border.

■ 관리자 사이트 커스터마이징

● 질문 등록 페이지에서 보기 등록 기능 추가

– polls/admin.py

```
class ChoiceInline(admin.TabularInline):
    model = Choice
    extra = 3

class QuestionAdmin(admin.ModelAdmin):
    fieldsets = [
        (None, {'fields': ['question_text']}),
        ('Date information', {'fields': ['pub_date']}),
    ]
    inlines = [ChoiceInline]

admin.site.register(Question, QuestionAdmin)
```

Home > Polls > Questions > Add question

Add question

Question text:

Date information

Date published:

Date: Today

Time: Now

Note: You are 9 hours ahead of server time.

CHOICES		
CHOICE TEXT	VOTES	DELETE?
<input type="text"/>	<input type="text" value="0"/>	
<input type="text"/>	<input type="text" value="0"/>	
<input type="text"/>	<input type="text" value="0"/>	
<input type="text"/>	<input type="text" value="0"/>	

■ 관리자 사이트 커스터마이징

● 질문 목록 페이지 항목 지정

– polls/admin.py

```
class QuestionAdmin(admin.ModelAdmin):  
    ...  
    list_display = ('question_text', 'pub_date', 'was_published_recently')
```

Home > Polls > Questions

Select question to change

ADD QUESTION +

Action: Go

0 of 1 selected

<input type="checkbox"/>	QUESTION
<input type="checkbox"/>	What's up?

1 question



Home > Polls > Questions

Select question to change

ADD QUESTION +

Action: Go

0 of 1 selected

<input type="checkbox"/>	QUESTION TEXT	DATE PUBLISHED	WAS PUBLISHED RECENTLY
<input type="checkbox"/>	What's up?	Oct. 19, 2018, 12:01 a.m.	False

1 question