

[Django] #4 - Django ORM을 통한 데이터 관리 예제 — 나무늘보의 개발 블로그

노트북: 첫 번째 노트북

만든 날짜: 2020-10-17 오후 11:45

URL: <https://continuous-development.tistory.com/93?category=805760>

Django

[Django] #4 - Django ORM을 통한 데이터 관리 예제

2020. 9. 21. 09:58 수정 삭제 공개

MVT

model / view / Template

view 는 로직처리 부분을 담당하고 있다.

이런 로직처리를 통해 보여지는 화면은 template를 한다 .

model부분에서 ORM을 했다.

객체 관계 매핑을 하였다. 하이버네이트/ 마이바티스도 이런 ORM이다.

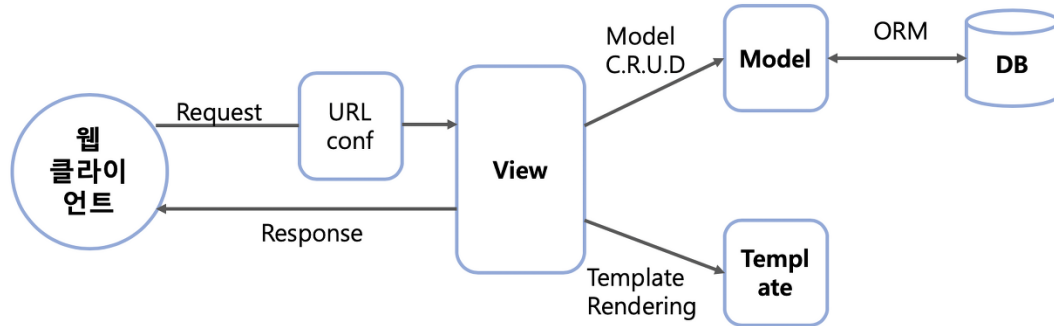
저번에 만든 클래스는 DB 의 테이블이고

여기서 생성된 객체는 하나의 컬럼이 된다.

지금 우리가 하는 것은 사용자의 리퀘스트에 따른 데이터를 지금 가져오고 있다.

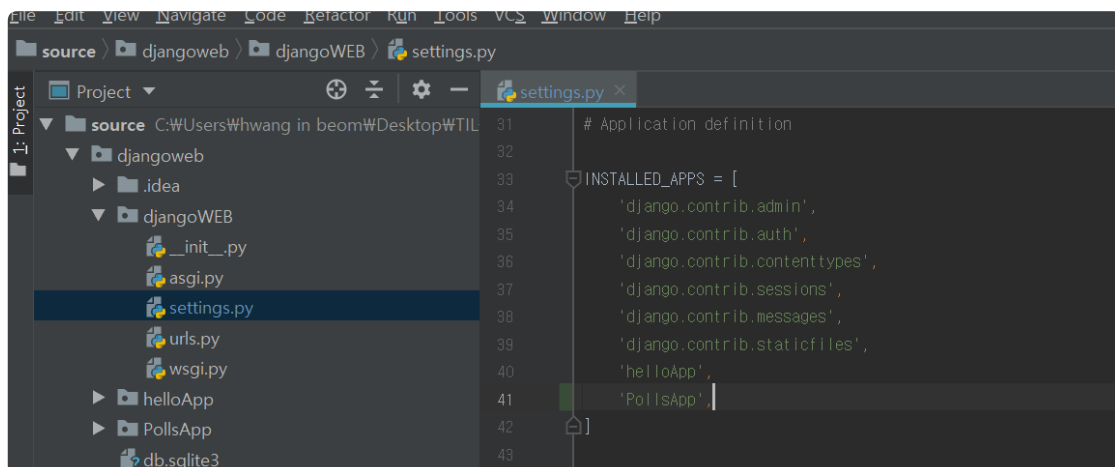
웹이라는 것은 사용자의 리퀘스트를 관리하는 url conf를 거친다.

이걸 거치면서 각각의 앱에서 view와 통신하는 방식을 가진다.



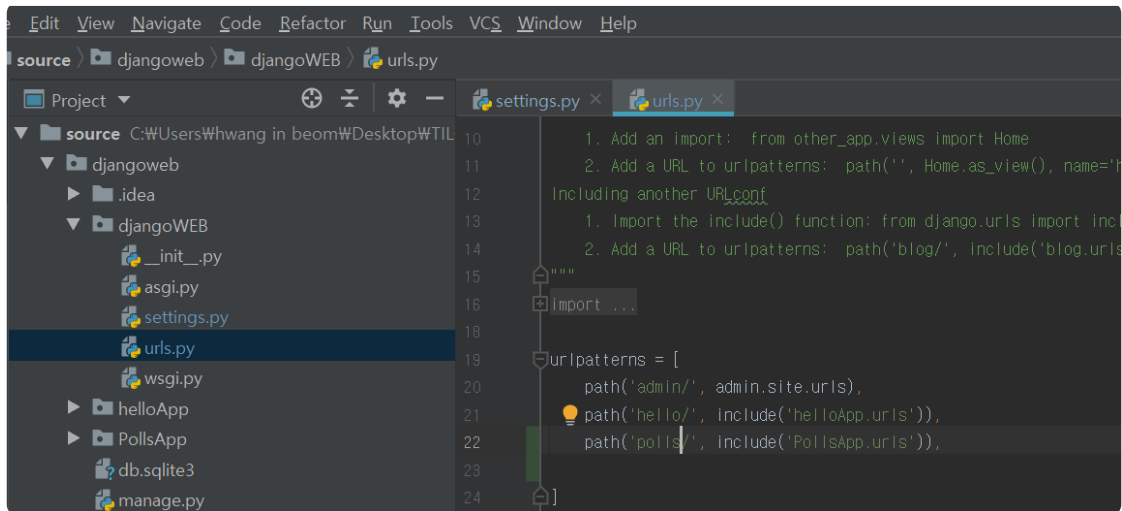
```
C:\Users\hwang in beom\Desktop\TIL\django\source\djangoWEB>python manage.py startapp PollsApp
```

처음에 app을 만든다음에

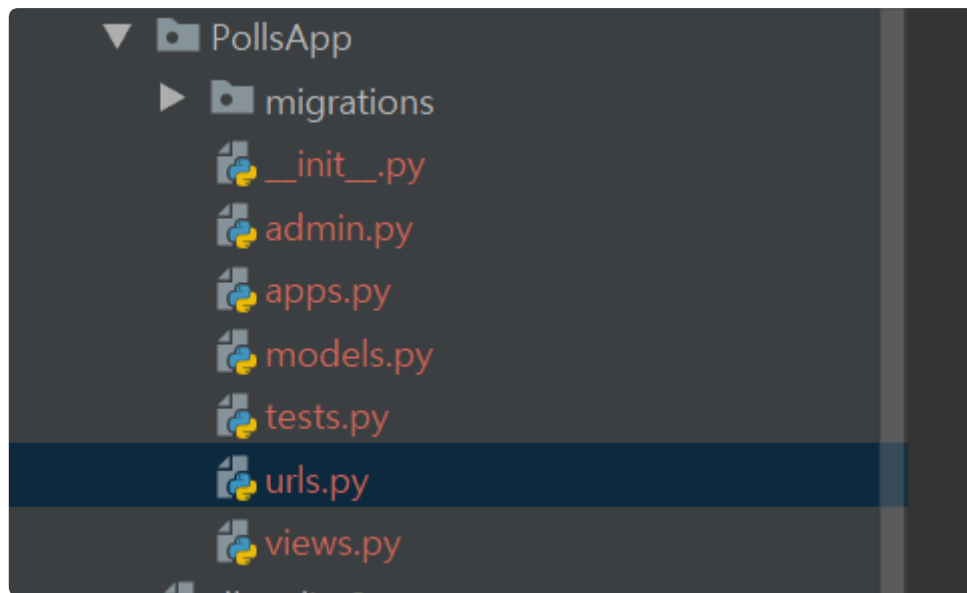


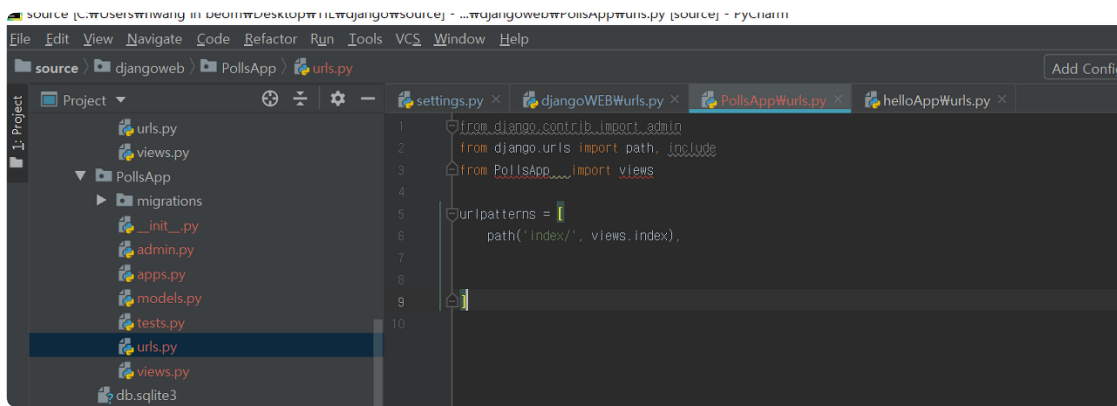
해당 프로젝트 setting 에서 app을 사용한다고 명시해줘야 한다.

그 다음에는 url 상에서 어디로 인식을 시켜주기 위해 path를 추가 해준다. 이렇게 해야 이 url로 들어오는 것을 polls App으로 가게끔 한다.



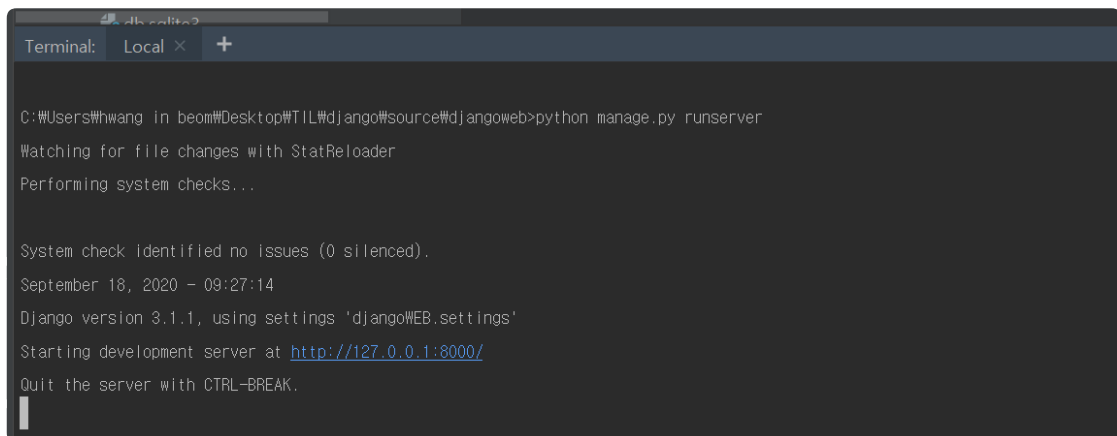
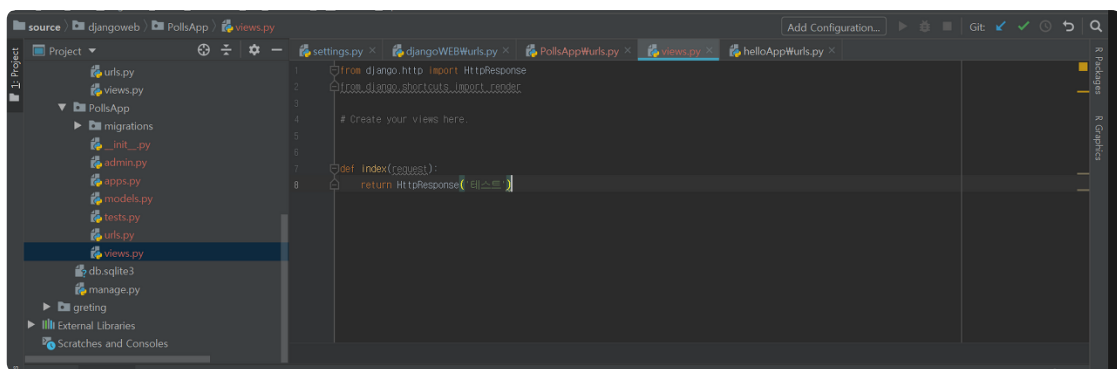
이 보낸 url을 받기위해 polls 에 urls를 생성한다.



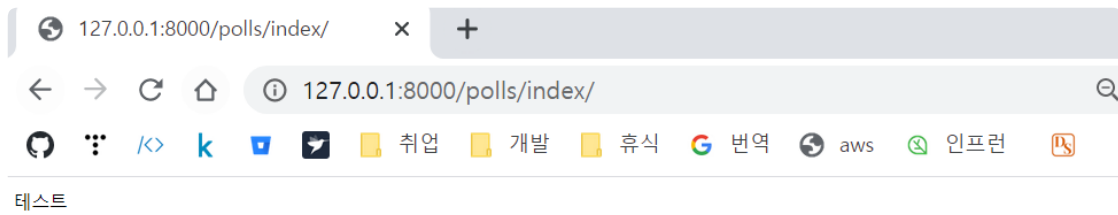


그 다음 urls를 세팅해준다. from 부분에 views를 가져온다고 명시해주고
urlpatterns 부분에 받을 url을 만들어준다.

그다음 view 부분을 설정해준다. 간단하게 되는지를 먼저 확인해준다.



python manage.py runserver 로 멀쩡히 되는지 확인한다.



멀쩡히 된다.

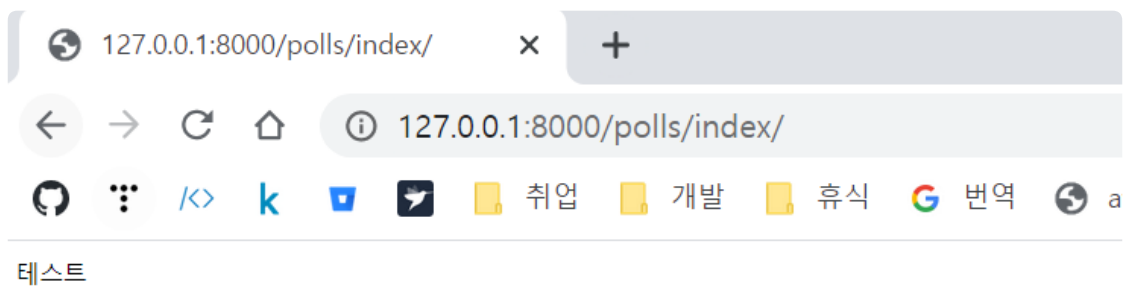
이 url은 project - polls 의 url을 타고 간다.

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('hello/', include('helloApp.urls')),  
    path('polls/', include('PollsApp.urls')),  
]
```

그래서

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

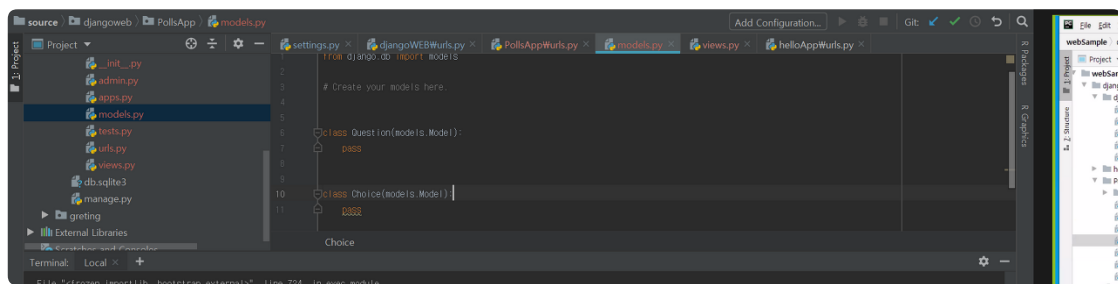
로 타고 들어간다.



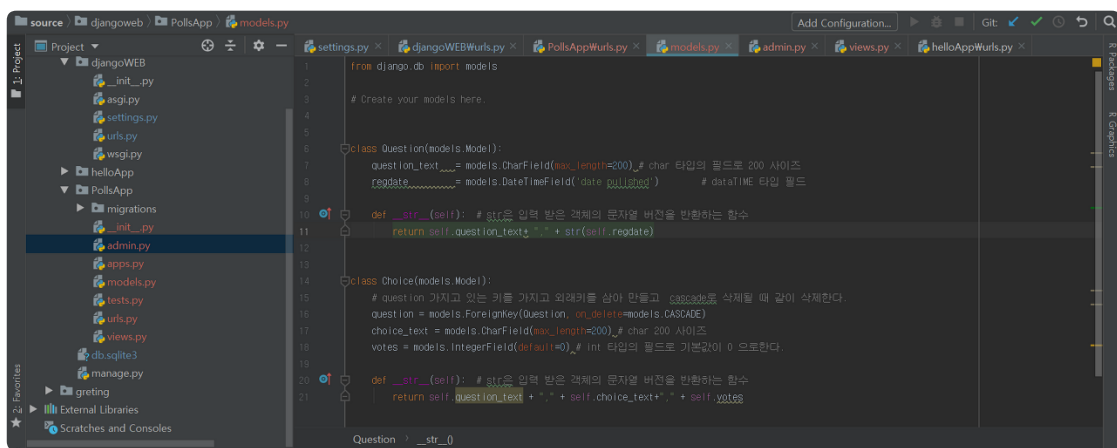
정상적으로 되는걸 볼 수 있다.

ORM 생성

이제 orm을 생성해보자

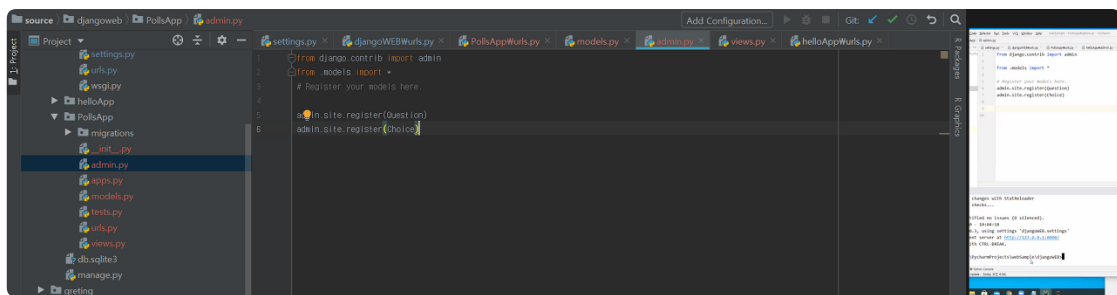


model 부분에 클래스를 만든다. 이건 table을 만드는 작업이라고 보면 된다.



이렇게 클래스 2개를 만든다.

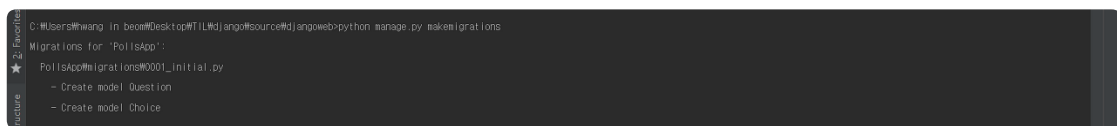
이렇게 만든 모델을 admin에 추가한다.



이렇게 admin에 추가하고

이제 migrate로 현재의 상태를 맞춰준다.

맨처음 migrate할 포인트를
python manage.py makemigrations



이제 migrate 한다.

python manage.py migrate

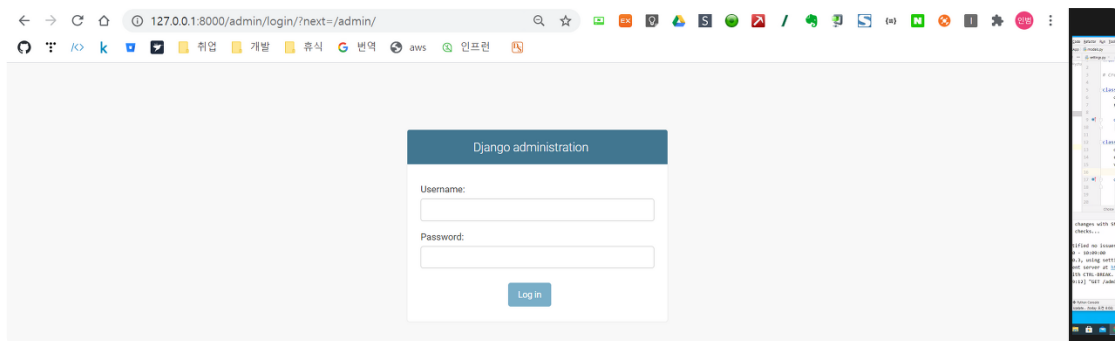
```
C:\Users\hwang> python manage.py migrate
Operations to perform:
  Apply all migrations: PollsApp, admin, auth, contenttypes, helloApp, sessions
Running migrations:
  Applying PollsApp.0001_initial... OK
```

```
C:\Users\hwang> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

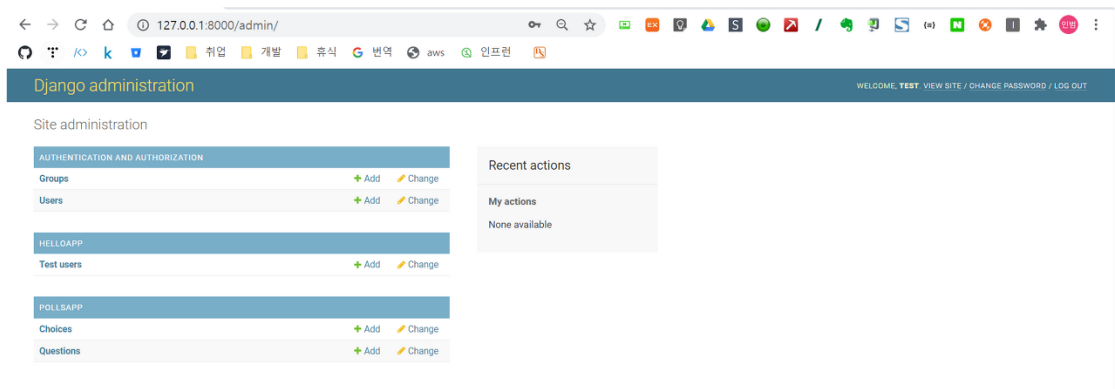
System check identified no issues (0 silenced).
September 18, 2020 - 10:07:35
Django version 3.1.1, using settings 'djangoWeb.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with Ctrl-C or BREAK
```

이제 runserver 한다.

이제 admin 으로 들어가고

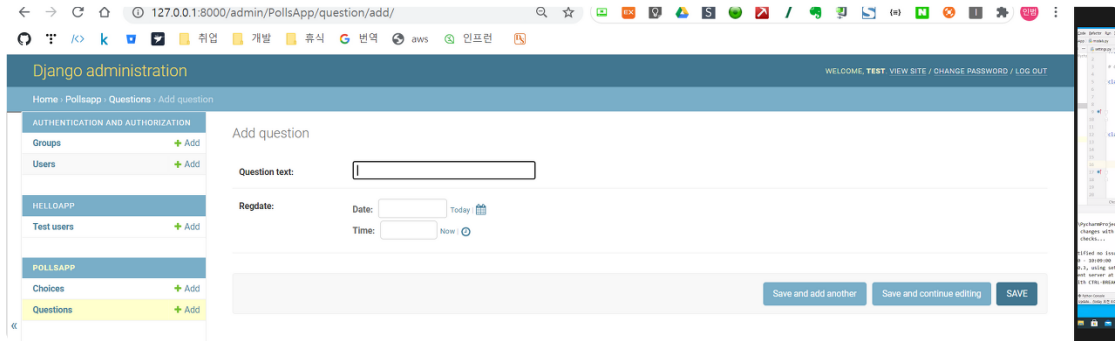


로그인하면

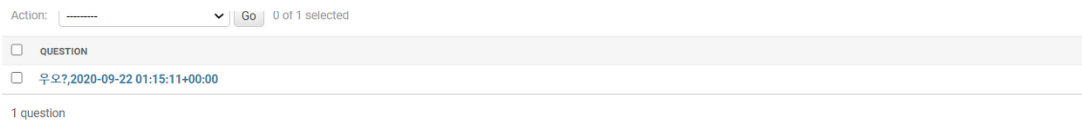


위와같이 pollsApp 이라는 것이 생기고 우리가 만들었던 두개의 테이블이 생성된 걸 볼 수있다.

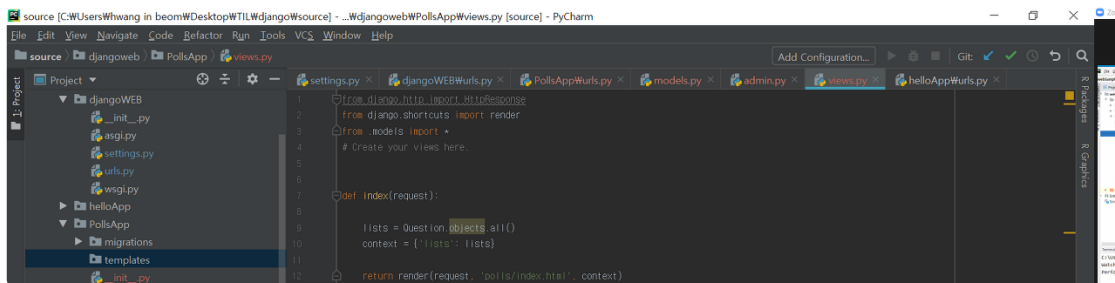
question 을 클릭하면



위와 같이 우리가 설정했던 형태를 볼 수 있다. 여기서 값을 넣어보자



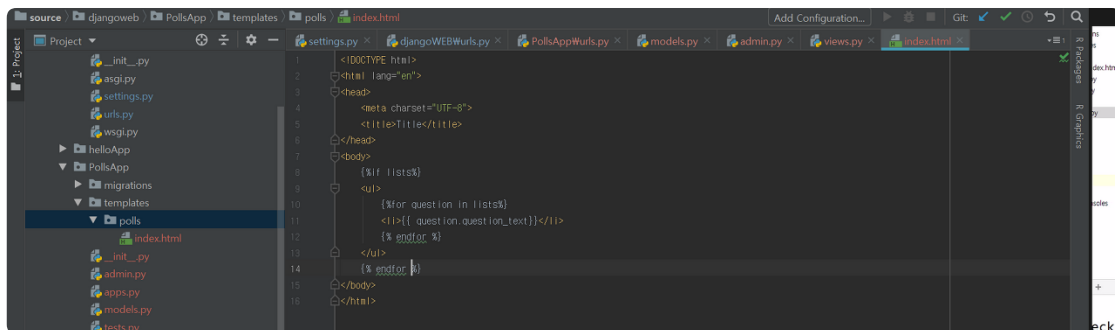
이렇게 저장이 되는걸 볼수 있다.



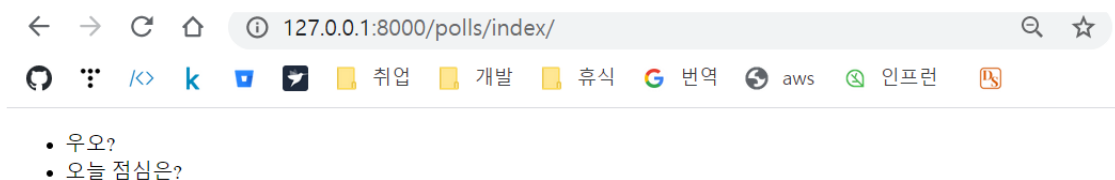
이렇게 view 부분에서 로직을 만들고 저건 모든값을 가져온다는 부분이다.



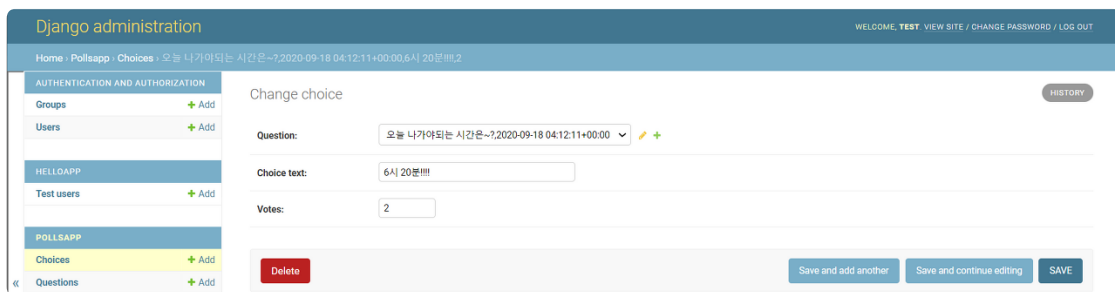
templates 라는 폴더를 만들고 그 밑에 polls 를 만든다
templates 이 폴더는 Django 에서 지정해준 폴더명으로써 이렇게 써야되고 그 밑에는 자유다.



index라는 페이지를 만드는데 동적으로 데이터를 넣게끔 만든다.



해당 url로 이동하면 아까 DB 에 넣었던 값이 이렇게 값이 나오는 걸 볼 수 있다.



똑같이 choice 도 해준다.

object에 쓰는 기본적인 CRUD

view 에서 object에 대한 기본적인 CRUD

```
select * from table;
```

```
-> modelName.objects.all
```

```
select * from table where id = 1;
```

```
-> modelName.objects.filter(id=1)
```

```
select * from table where id = 1 and pwd = 1;
```

```
-> modelName.objects.filter(id = 1, pwd = 1)
```

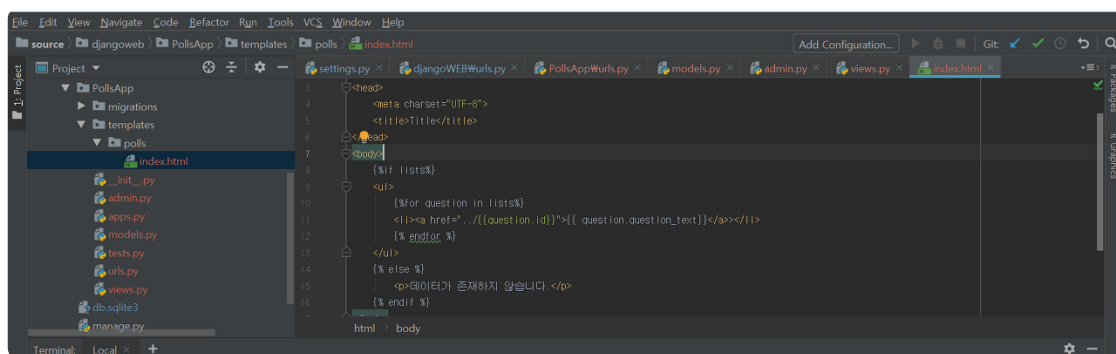
```
select * from table where id = 1 or pwd = 1;
```

```
-> modelName.objects.filter(Q(id = 1) | Q(pwd = 1))
```

```
select * from table where subject like '%공지%'
```

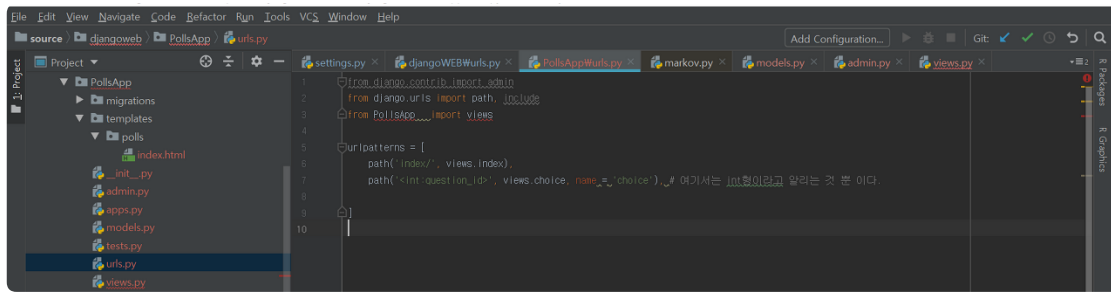
```
-> modelName.objects.filter(subject_icontains='공지')
```

```
obj.save() -- commit
```



안에 들어가는 question.id는 클래스에서 따로 지정해주지 않아도 내부에 들어가있다. id 값이기 때문이다.

주소가 바뀌는 것에따라 결과를 보여주려면 url 부분을 수정해야 한다.

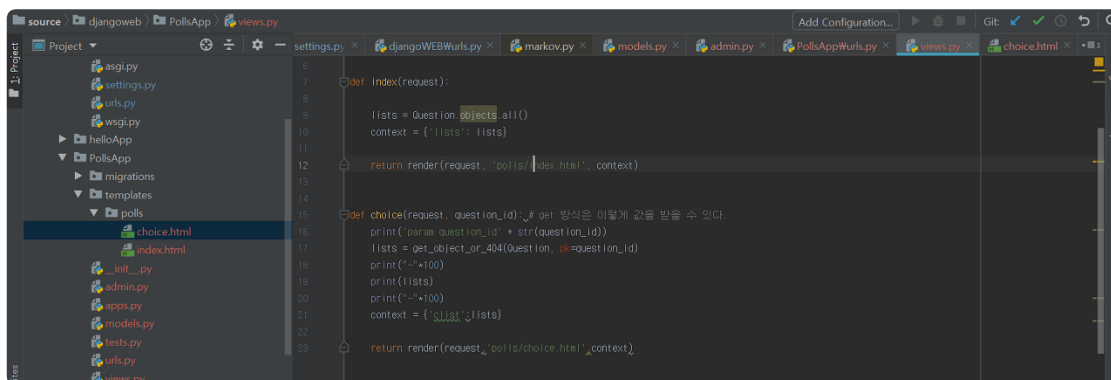


```
1 from django.conf.urls import url
2 from django.urls import path, include
3 from PollsApp import views
4
5 urlpatterns = [
6     path('index/', views.index),
7     path('<int:question_id>', views.choice, name='choice'), # 여기서는 int형이라고 말하는 것 뿐 이다.
8 ]
9
10
```

url에 따른 view 부분을 작성한다 .



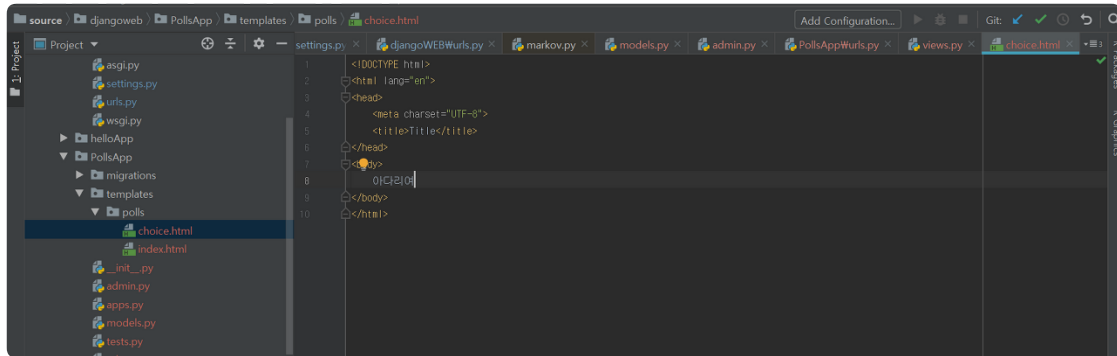
```
7 def index(request):
8
9     lists = Question.objects.all()
10    context = {'lists': lists}
11
12    return render(request, 'polls/index.html', context)
13
14 def choice(request, question_id): # get 방식은 이렇게 값을 받을 수 있다.
15    print(str(question_id))
16
```



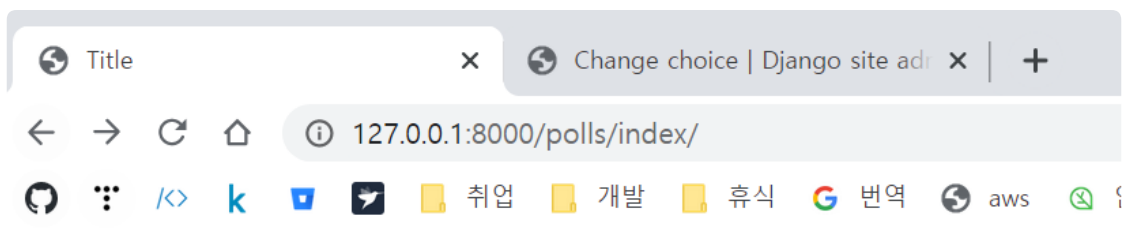
```
6
7 def index(request):
8
9     lists = Question.objects.all()
10    context = {'lists': lists}
11
12    return render(request, 'polls/index.html', context)
13
14
15 def choice(request, question_id): # get 방식은 이렇게 값을 받을 수 있다.
16    print((param question_id + str(question_id))
17    lists = get_object_or_404(Question, pk=question_id)
18    print("-->+100)
19    print(lists)
20    print("-->+100)
21    context = {'lists': lists}
22
23    return render(request, 'polls/choice.html', context)
24
```

작성 완료

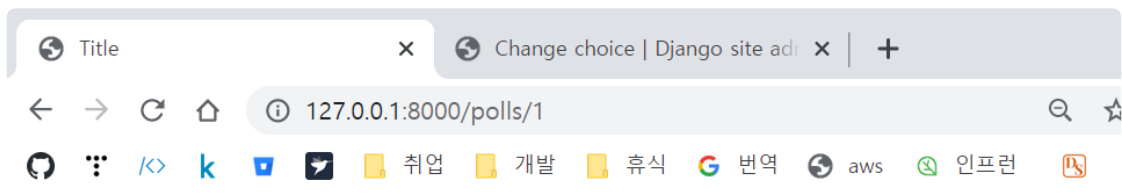
여기까지 진행이 됐다면 choice.html을 만들어보자



일단은 페이지가 돌아가는지 확인하기 위해 여기까지 만들고 확인해본다.



해당값을 클릭했을때



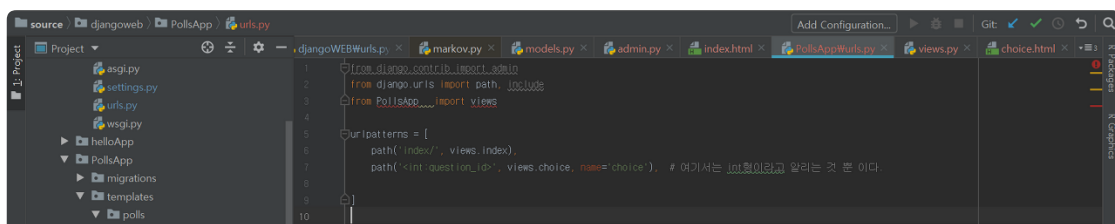
지금은 이렇게 나온다.

정리해보면



```
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
  {% if lists %}
  <ul>
    {% for question in lists %}
    <li><a href="{% url 'polls:index' question.id %}">{{ question.question_text }}</a></li>
    {% endfor %}
  </ul>
  {% else %}
  <p>데이터가 존재하지 않습니다.</p>
  {% endif %}
</body>
</html>
```

여기서 question.id 로 인덱스를 해서 보내는데 이 url이



```
1 from django.conf.urls import include
2 from django.urls import path, include
3 from PollsApp import views
4
5 urlpatterns = [
6     path('index/', views.index),
7     path('<int:question_id>', views.choice, name='choice'), # 여기서는 [이동이랑] 일라는 것 뿐 이다.
8 ]
```

이경로로 이동해서 해당 path로 이동하게 되고

이 path에 해당 하는 view로 이동하게 되고

```

7 def index(request):
8
9     lists = Question.objects.all()
10    context = {'lists': lists}
11
12    return render(request, 'polls/index.html', context)
13
14
15 def choice(request, question_id): # get 방식은 이렇게 값을 받을 수 있다.
16     print('param question_id' + str(question_id))
17     lists = get_object_or_404(Question, pk=question_id) # 객체가 존재하지 않을 때 404를 띄우고 아닐 경우 값을 받아와 넘긴다.
18     # pk 값이 인덱스 값인 question을 가지고 와서 lists에 넣는다
19     print("-"*100)
20     print(lists)
21     print("-"*100)
22     context = {'clist': lists}
23
24     return render(request, 'polls/choice.html', context)

```

이 view로 이동해서 choice 라는 함수로 이동하게 된다. 여기서 request 와 question_id 를 매개변수로 삼는데 request 는 사용자의 요청이고 question_id 는 path에서 받았던 int 값이 된다.

이 값을 통해 choice에서 pk로 삼아 Question 객체를 가져오고 그 값이 존재하면 값을 lists에 넣고 없을 경우에 page 404 에러를 발생시킨다.

이 값을 context에 넣어 리턴값으로 render 해 choice.html로 다시 보내 주게 된다.

```

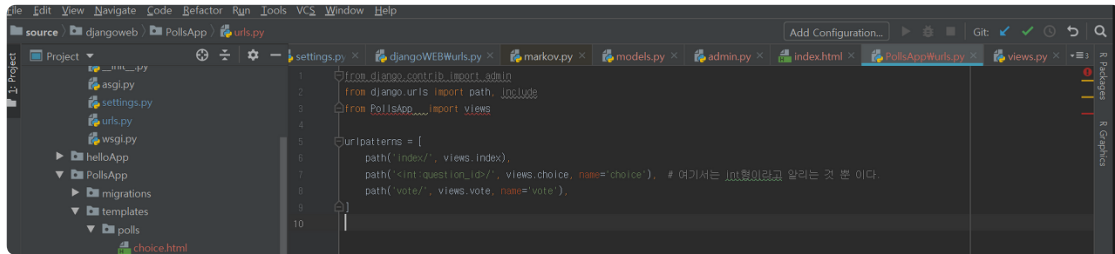
3
4 <meta charset="UTF-8">
5 <title>Title</title>
6
7 <body>
8     <h1>{{ clist.question_text }}</h1>
9
10    <form method="post" action="{% url 'vote' %}">
11        <input type="hidden" name="question" value="{{ clist.id }}"> <!-- 이 형식으로 hidden으로 형태는 안보이게 하되 값을 넣어서 보내준다. -->
12        {% csrf_token %} <!-- django 에서 post를 쓸때는 csrf 토큰을 사용한다. 보안을 위해서 이다. -->
13        {% for choice in clist.choice_set.all %} <!-- xxx_set.all 은 왼쪽에서 사용하는 코드로 모든 객체들을 가져온다. -->
14            <input type="radio"
15                name="choice"
16                value="{{ choice.id }}"
17            >
18            <label>
19                {{ choice.choice_text }}
20            </label><br/>
21            {% endfor %}
22            <input type="submit" value="VOTE">
23        </form>
24    </body>

```

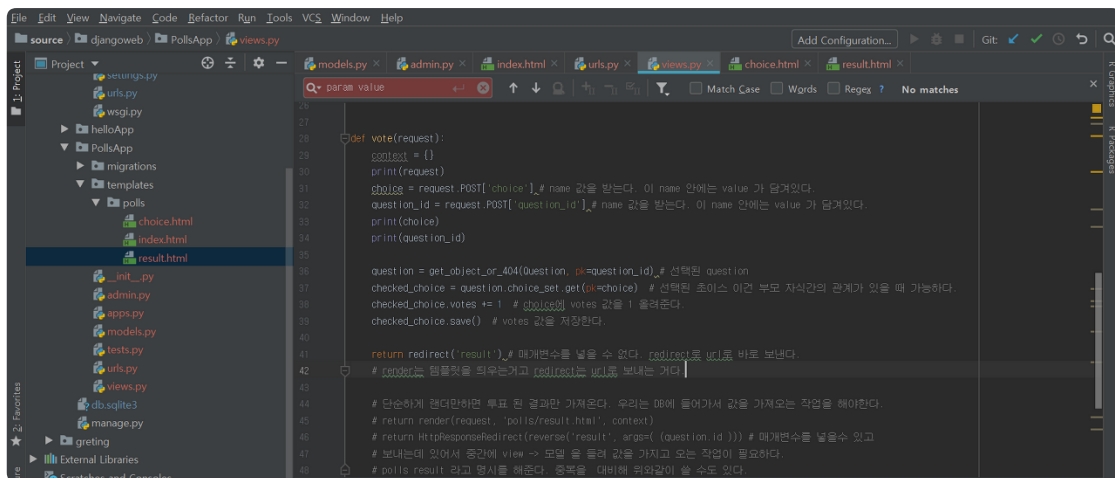
choice.html을 위와 같이 만들어

해당 값을 선택하는 것에 따라 submit을해 post방식으로 해당 action으로 이어지게끔 한다.

이때 넘어가는 값은 value 값이 넘어간다.

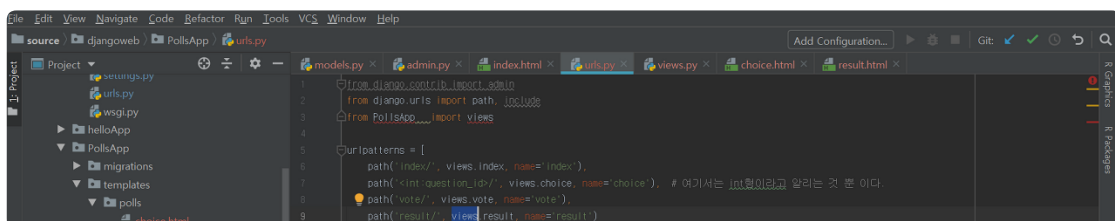


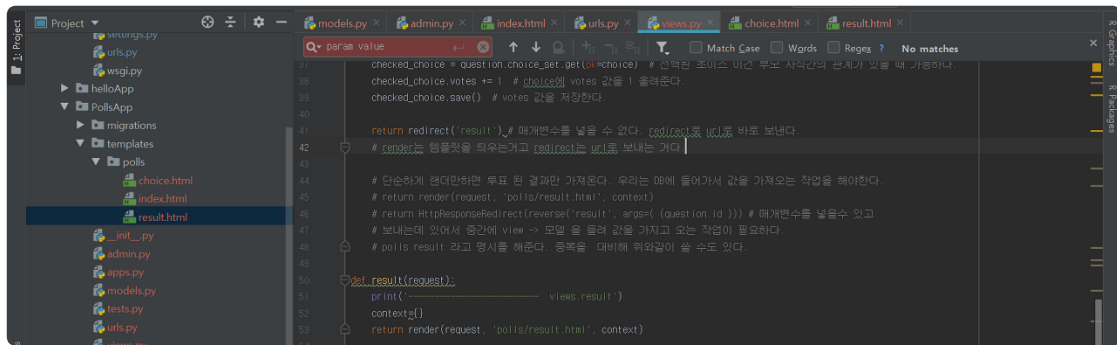
url의 vote 부분을 만들고 vote 의 view 부분을 다시 만든다.



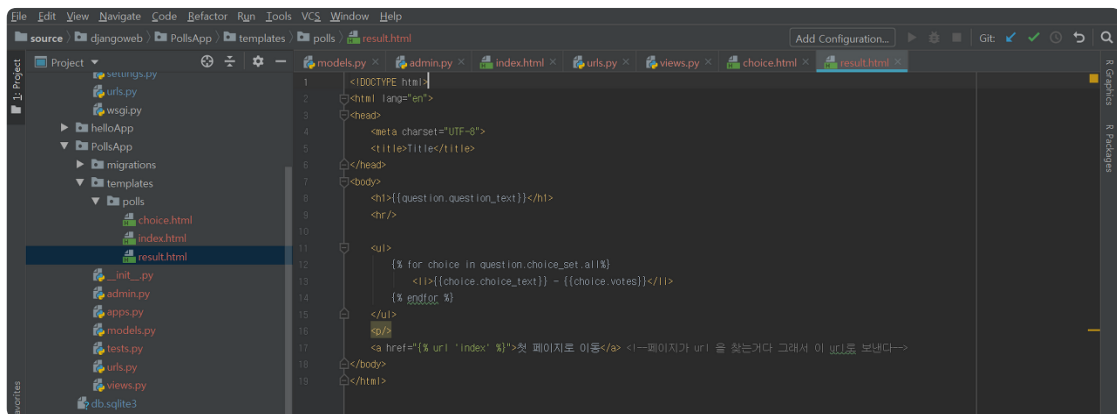
vote의 view 부분이다.

그리고 그거에 맞춰 urls를 만들어준다.



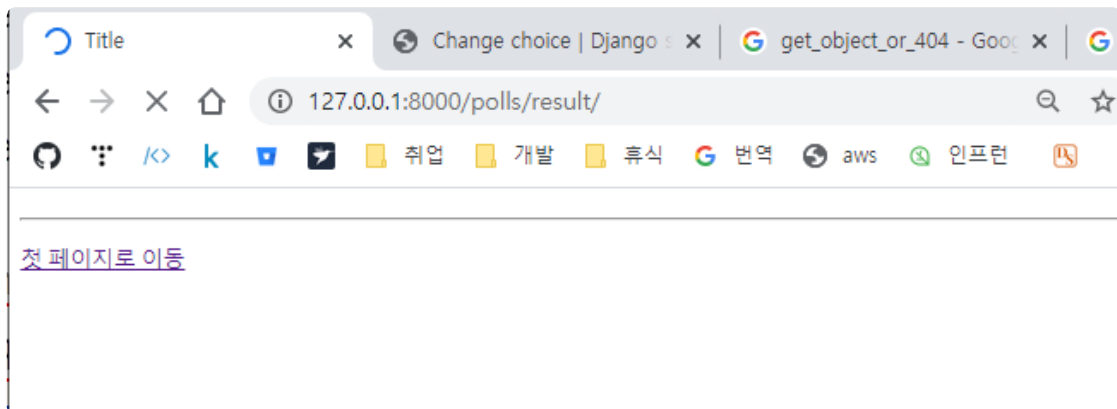


다시 view 부분에 result를 만들어주고

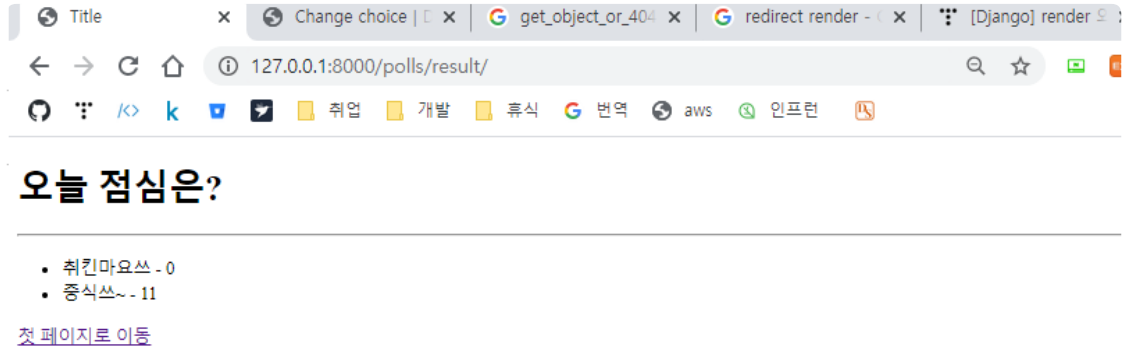
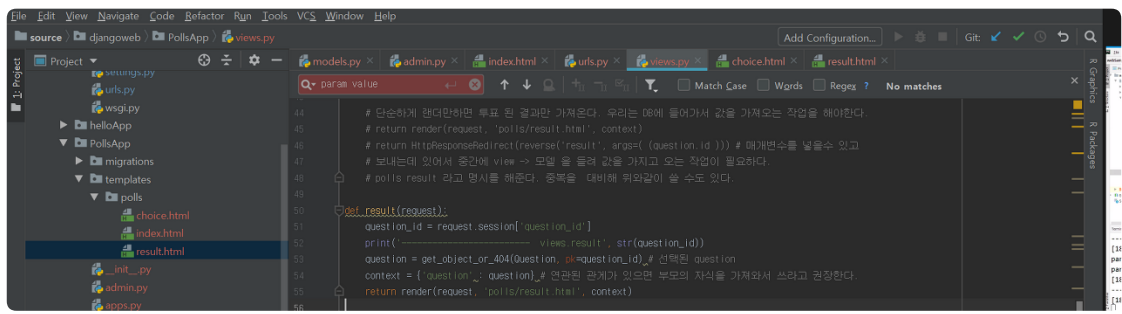


result.html을 만들어주면 연결된다.

투표했을때 아래와 같이 나온다.



result의 view 부분을 다시 수정하면



이렇게 된다.

'Django' 카테고리의 다른 글

[Django] #6 - Django 를 통한 사용자 등록 구현하기

[Django] #5 - Django 를 통한 static 사용하기

[Django] #4 - Django ORM을 통한 데이터 관리 예제

[Django] #3 - Django ORM을 통한 데이터 관리

[Django] #2 - Django 개발 환경 세팅 및 서버 실행까지

[Django] #1 - Django 란?

django ORM

Django ORM 예제

ORM

ORM 예제



꾸까꾸

혼자 끄적끄적하는 블로그 입니다.