

# Лабораторная работа 1

## Часть 1

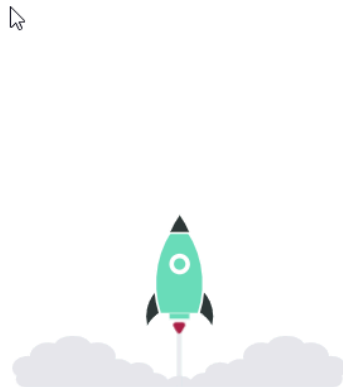
Установка фреймворка

```
$ pip install django
Collecting django
  Downloading Django-5.1.4-py3-none-any.whl.metadata (4.2 kB)
Collecting asgiref<4,>=3.8.1 (from django)
  Using cached asgiref-3.8.1-py3-none-any.whl.metadata (9.3 kB)
Collecting sqlparse>=0.3.1 (from django)
  Downloading sqlparse-0.5.3-py3-none-any.whl.metadata (3.9 kB)
Collecting tzdata (from django)
  Using cached tzdata-2024.2-py2.py3-none-any.whl.metadata (1.4 kB)
Downloading Django-5.1.4-py3-none-any.whl (8.3 MB)
6.0/8.3 MB 9.6 MB/s eta 0:00:01
```

Инсталируем проект

```
$ django-admin startproject mysite
```

Командой `python manage.py runserver` запускаем приложение



The install worked successfully! Congratulations!

View [release notes](#) for Django 5.1

You are seeing this page because `DEBUG=True` is in your settings file and you have not configured any URLs.

**django**

Создаем приложение опросов

```
$ python manage.py startapp polls
```

Создаем представление (views)

```
1 from django.http import HttpResponse
2
3
4 def index(request):
5     return HttpResponse("Hello, world. You're at the polls index.")
```

Теперь настраиваем главный urls.py всего приложения и urls.py polls

```
1 from django.urls import path
2
3 from . import views
4
5 urlpatterns = [
6     path("", views.index, name="index"),
7 ]
```

```
1 from django.contrib import admin
2 from django.urls import include, path
3
4 urlpatterns = [
5     path("polls/", include("polls.urls")),
6     path("admin/", admin.site.urls),
7 ]
```

## Часть 2

Чтобы данные с бд отображались надо применить миграции

```
$ 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...
```

Создаем собственную модель, после чего сможем создавать объекты в бд

```
1 from django.db import models
2
3
4 class Question(models.Model):
5     question_text = models.CharField(max_length=200)
6     pub_date = models.DateTimeField("date published")
7
8
9 class Choice(models.Model):
10     question = models.ForeignKey(Question, on_delete=models.CASCADE)
11     choice_text = models.CharField(max_length=200)
12     votes = models.IntegerField(default=0)
```

Создаем миграции приложения polls

```
$ python manage.py makemigrations polls
Migrations for 'polls':
  polls\migrations\0001_initial.py
    + Create model Question
    + Create model Choice
```

Можно посмотреть на эти миграции

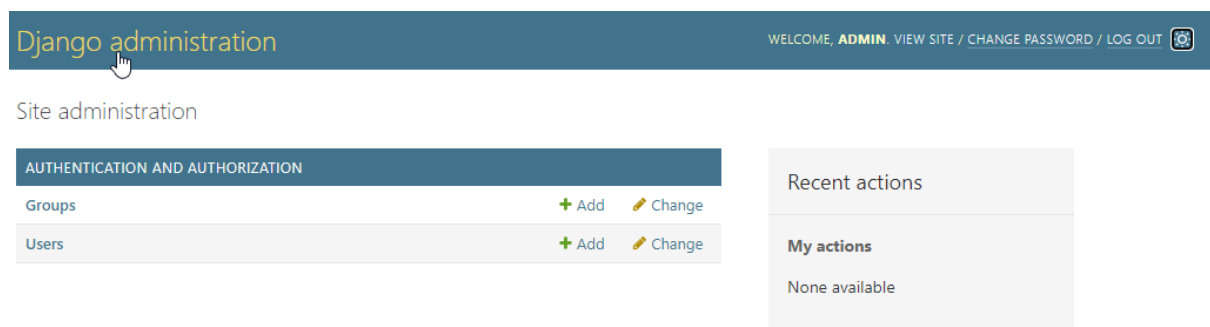
```
$ python manage.py sqlmigrate polls 0001
BEGIN;
--
-- Create model Question
--
CREATE TABLE "polls_question" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "question_text"
--
-- Create model Choice
--
CREATE TABLE "polls_choice" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "choice_text" var
CREATE INDEX "polls_choice_question_id_c5b4b260" ON "polls_choice" ("question_id");
COMMIT;
(venv)
```

И применяем новые миграции

```
$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, polls, sessions
Running migrations:
  Applying polls.0001_initial... OK
(venv)
```

Создадим админа, чтобы посмотреть объекты в бд

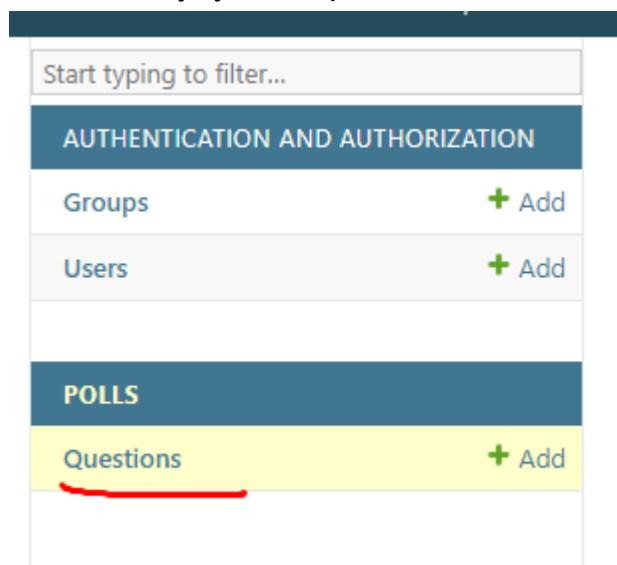
```
$ python manage.py createsuperuser
```



Для отображения в админке необходимо зарегистрировать модель

```
1 from django.contrib import admin
2
3 from .models import Question
4
5 admin.site.register(Question)
```

Объекты будут отображаться так:



Select question to change

Action:   0 of 1 selected

<input type="checkbox"/>	QUESTION
<input type="checkbox"/>	Question object (1)

1 question

## Часть 3

Будем оптимизировать представления

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

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

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

также обновляем urls.py приложения polls

```
from . import views

urlpatterns = [
    # ex: /polls/
    path("", views.index, name="index"),
    # ex: /polls/5/
    path("<int:question_id>/", views.detail, name="detail"),
    # ex: /polls/5/results/
    path("<int:question_id>/results/", views.results, name="results"),
    # ex: /polls/5/vote/
    path("<int:question_id>/vote/", views.vote, name="vote"),
]
```

```

Ctrl+L to chat, Ctrl+K to generate
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(output)

```

Так работает нейминг приложения для записи в шаблонах:

```

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"),
]

```

## Часть 4

Изменим представление, отвечающее за голосование

```

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 = F("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,)))

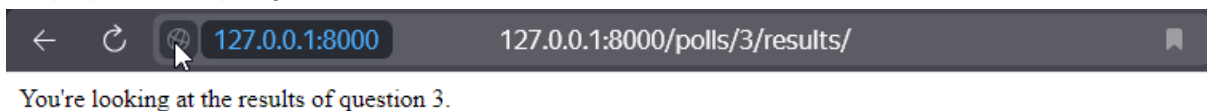
```

перейдя по ссылке, увидим такой результат: (детальная информация)

← ↻ 🌐 127.0.0.1:8000 127.0.0.1:8000/polls/3/

You're looking at question 3.

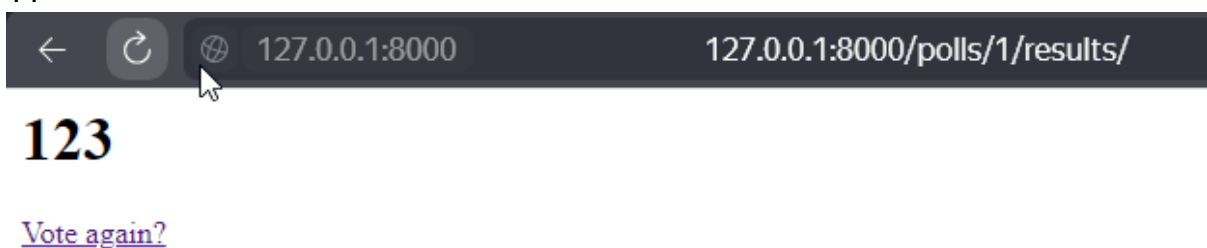
информация результата



Обновим представление отображения результата

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

и теперь у нас такое отображение. меняя index.html можно настраивать фронтенд



## Часть 5

В предыдущей версии была ошибка в атрибуте класса, изменим его:

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

```
$ \python manage.py shell
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun 6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> import datetime
>>> from django.utils import timezone
>>> from polls.models import Question
>>> future_question = Question(pub_date=timezone.now() + datetime.timedelta(days=30))
>>> future_question.was_published_recently()
False
>>>
```

```
$ python manage.py test polls
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.
-----
Ran 1 test in 0.001s

OK
Destroying test database for alias 'default'...
(venv)
```

```
def test_was_published_recently_with_old_question(self):
    """
    was_published_recently() returns False for questions whose pub_date
    is older than 1 day.
    """
    time = timezone.now() - datetime.timedelta(days=1, seconds=1)
    old_question = Question(pub_date=time)
    self.assertIs(old_question.was_published_recently(), False)

Ctrl+L to chat, Ctrl+K to generate
def test_was_published_recently_with_recent_question(self):
    """
    was_published_recently() returns True for questions whose pub_date
    is within the last day.
    """
    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)
```

```
$ python manage.py test polls
Found 3 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
...
-----
Ran 3 tests in 0.001s

OK
Destroying test database for alias 'default'...
(venv)
```

```
class IndexView(generic.ListView):
    template_name = "polls/index.html"
    context_object_name = "latest_question_list"

    def get_queryset(self):
        """Return the last five published questions."""
        return Question.objects.order_by("-pub_date")[:5]
```



```

from django.utils import timezone
from django.urls import reverse
    Ctrl+L to chat, Ctrl+K to generate
from .models import Question

def create_question(question_text, days):
    """
    Create a question with the given `question_text` and published the
    given number of `days` offset to now (negative for questions published
    in the past, positive for questions that have yet to be published).
    """
    time = timezone.now() + datetime.timedelta(days=days)
    return Question.objects.create(question_text=question_text, pub_date=time)

class QuestionIndexViewTests(TestCase):
    def test_no_questions(self):
        """
        If no questions exist, an appropriate message is displayed.
        """
        response = self.client.get(reverse("polls:index"))
        self.assertEqual(response.status_code, 200)
        self.assertContains(response, "No polls are available.")
        self.assertQuerySetEqual(response.context["latest_question_list"], [])

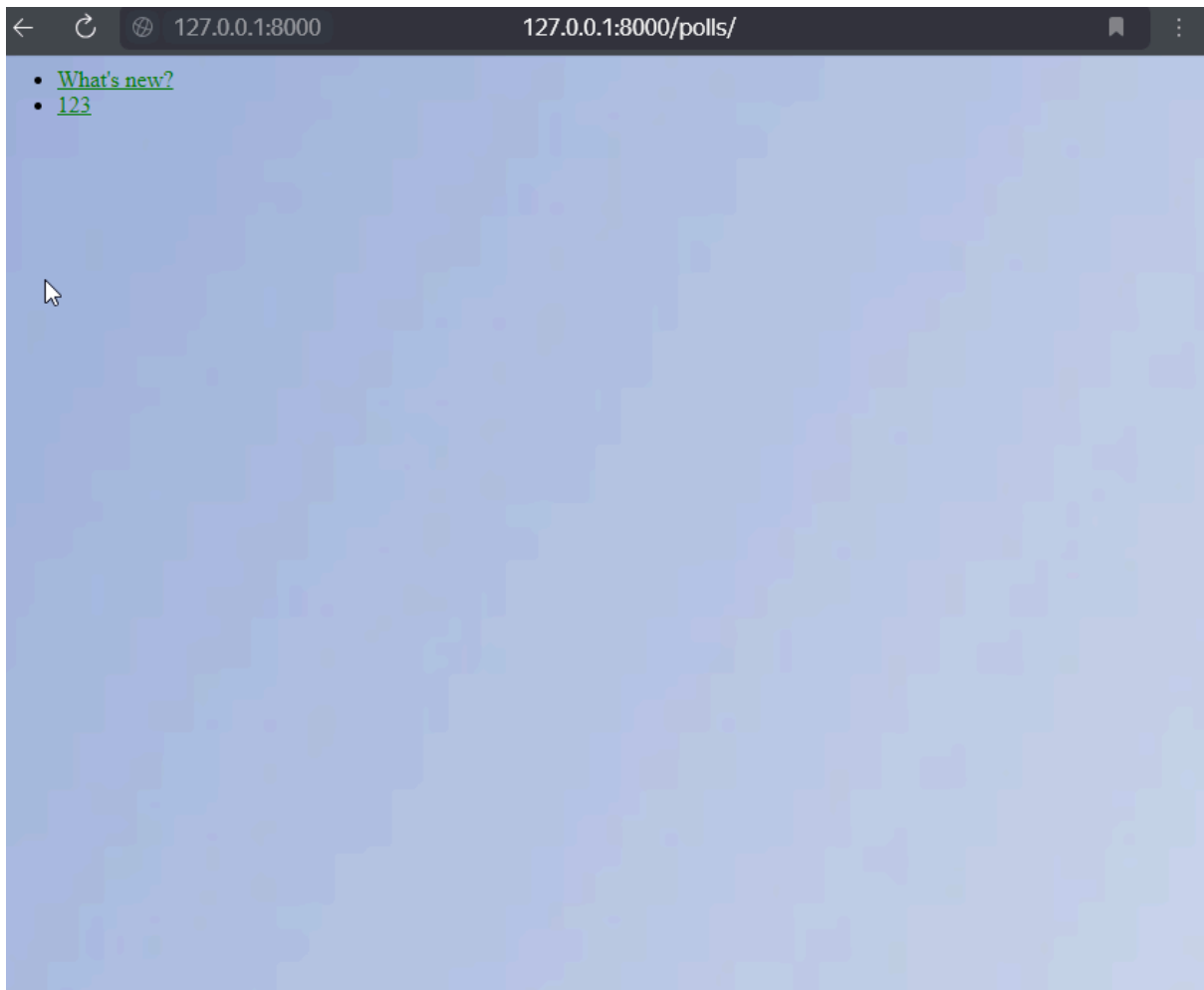
    def test_past_question(self):

```

в test.py создали тесты и запустили их. все работает исправно

## Часть 6

Изменим index.html, добавив статику



```
mysite > polls > templates > polls > <> index.html
{% load static %}

<link rel="stylesheet" href="{% static 'polls/style.css' %}">
{% if latest_question_list %}
<ul>
  {% for question in latest_question_list %}
    <li><a href="{% url 'polls:detail' pk=question.pk %}">{{ question.question_text }}</a></li>
  {% endfor %}
</ul>
{% else %}
<p>No polls are available.</p>
{% endif %}
```

```
li a {
  color: green;
}
body {
  background: white url("images/background.png") no-repeat;
}
```

Ctrl+L to chat, Ctrl+K to generate

## Часть 7


Админку можно кастомизировать, можно настроить отображение полей объекта так:


```
1 from django.contrib import admin
2
3 from .models import Question
4
5
6 class QuestionAdmin(admin.ModelAdmin):
7     fields = ["pub_date", "question_text"]
8
9
10 admin.site.register(Question, QuestionAdmin)
```

Change question

### What's new?

Date published:

Date: 2025-01-04 Today | 

Time: 19:06:51 Now | 

Note: You are 3 hours ahead of server time.

Question text:

What's new?

А можно сделать так:

```
from django.contrib import admin

from .models import Question

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

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

Change question

### What's new?

Question text:


What's new?

#### Date information

Date published:

Date:

2025-01-04

Today | 

Time:

19:06:51

Now | 

Note: You are 3 hours ahead of server time.

зарегистрируем модель выбора

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

Add choice

Question:



Choice text:



Votes:

0

SAVE

Save and add another

Save and continue editing

настроим его отображение в admin.py

```
from .models import Question, Choice

from django.contrib import admin

from .models import Choice, Question

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

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

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

теперь админка выглядит так:

Add question

Question text:

► Date information

#### CHOICES

Choice: #1



Choice text:

Votes:

Choice: #2



Choice text:

Votes:

Choice: #3



Choice text:



Votes:

+ Add another Choice

Add question

Question text:

▼ Date information

Date published: Date:  Today |   
Time:  Now |   
Note: You are 3 hours ahead of server time.

CHOICES

Choice: #1 

Choice text:

Votes:

Choice: #2 


Choice text:

Votes:

Choice: #3 

Choice text:

Votes:

 Add another Choice

SAVE

Можно настроить отображение определенных полей, а также добавить фильтрацию по определенным полям

```
inlines = [ChoiceInline]
list_display = ["question_text", "pub_date", "was_published_recently"]
```

<input type="checkbox"/>	QUESTION TEXT	DATE PUBLISHED	WAS PUBLISHED RECENTLY
<input type="checkbox"/>	What's new?	Jan. 4, 2025, 7:06 p.m.	True
<input type="checkbox"/>	123	Jan. 4, 2025, 6:51 p.m.	True

```
list_display = ["question_text"]
list_filter = ["pub_date"]
```

ction:   0 of 2 selected

	QUESTION TEXT	DATE PUBLISHED	WAS PUBLISHED RECENTLY
]	123	Jan. 4, 2025, 6:51 p.m.	True
]	What's new?	Jan. 4, 2025, 7:06 p.m.	True