

Writing your first Django app, part 1

Checking that Django is installed.

```
PS C:\Users\clari> py -m django --version
5.1.1
```

Creating a project.

```
PS C:\Users\clari> mkdir mysite

Directorio: C:\Users\clari

Mode                LastWriteTime         Length Name
----                -
d-----          25/09/2024   4:07 p. m.             mysite

PS C:\Users\clari> cd mysite
PS C:\Users\clari\mysite> django-admin startproject mysite
```

Checking that the project was correctly created.

```
PS C:\Users\clari\mysite> cd mysite
PS C:\Users\clari\mysite\mysite> ls

Directorio: C:\Users\clari\mysite\mysite

Mode                LastWriteTime         Length Name
----                -
d-----          25/09/2024   4:07 p. m.             mysite
-a-----          25/09/2024   4:07 p. m.             684 manage.py

PS C:\Users\clari\mysite\mysite> cd mysite
PS C:\Users\clari\mysite\mysite\mysite> ls

Directorio: C:\Users\clari\mysite\mysite\mysite

Mode                LastWriteTime         Length Name
----                -
-a-----          25/09/2024   4:07 p. m.             405 asgi.py
-a-----          25/09/2024   4:07 p. m.            3344 settings.py
-a-----          25/09/2024   4:07 p. m.             784 urls.py
-a-----          25/09/2024   4:07 p. m.             405 wsgi.py
-a-----          25/09/2024   4:07 p. m.              0 __init__.py
```

Running the local server.

```

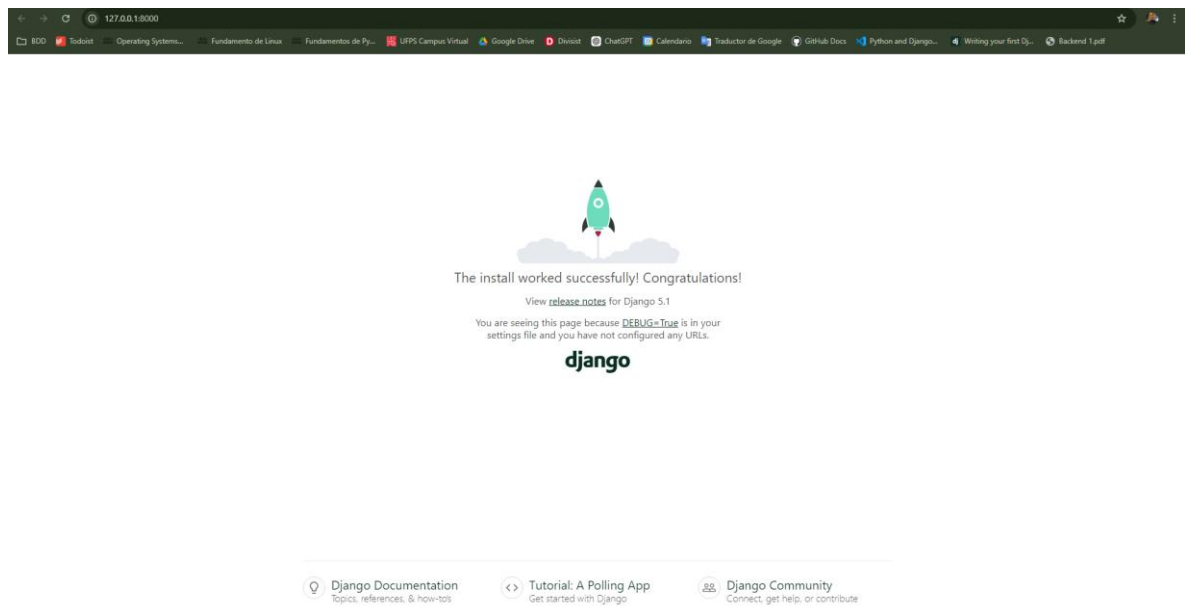
PS C:\Users\clari\mysite\mysite> py manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin,
auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
September 25, 2024 - 16:12:39
Django version 5.1.1, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

[25/Sep/2024 16:12:45] "GET / HTTP/1.1" 200 12068
Not Found: /favicon.ico
[25/Sep/2024 16:12:45] "GET /favicon.ico HTTP/1.1" 404 2208

```



Creating the Polls app.

```

PS C:\Users\clari\mysite\mysite> py manage.py startapp polls
PS C:\Users\clari\mysite\mysite> cd polls
PS C:\Users\clari\mysite\mysite\polls> ls

Directorio: C:\Users\clari\mysite\mysite\polls

Mode                LastWriteTime         Length Name
----                -
d-----          25/09/2024   4:15 p. m.      migrations
-a----          25/09/2024   4:15 p. m.         66 admin.py
-a----          25/09/2024   4:15 p. m.        148 apps.py
-a----          25/09/2024   4:15 p. m.         60 models.py
-a----          25/09/2024   4:15 p. m.         63 tests.py
-a----          25/09/2024   4:15 p. m.         66 views.py
-a----          25/09/2024   4:15 p. m.          0 __init__.py

```

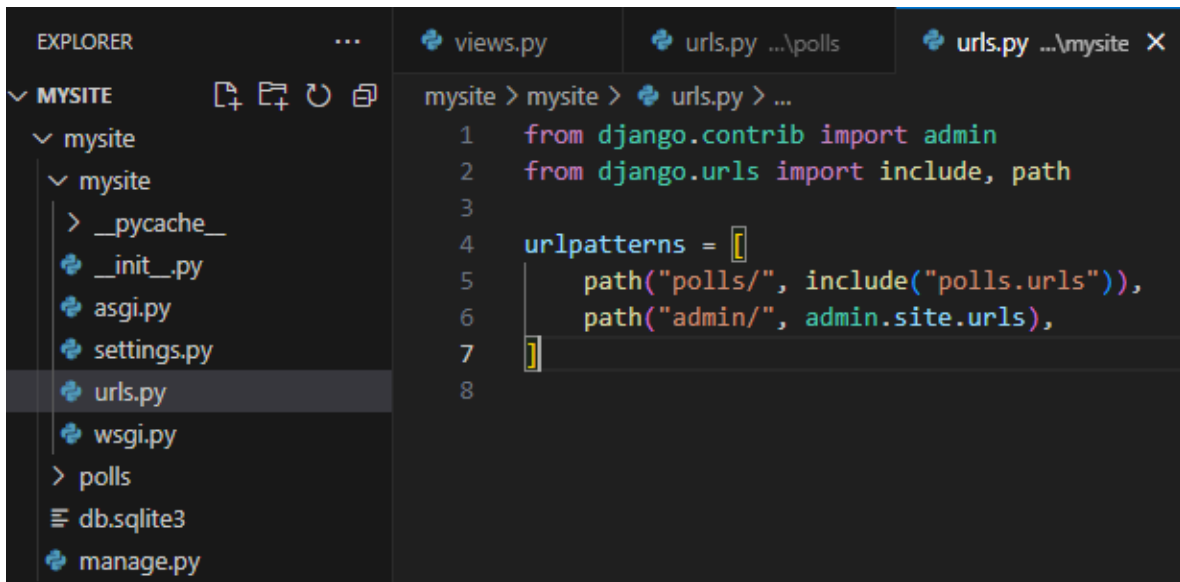
Write your first view.

This screenshot shows the Visual Studio Code editor with the Explorer sidebar on the left and the Editor pane on the right. The Explorer sidebar shows a project named 'mysite' with a subdirectory 'polls'. Inside 'polls', there are files for migrations, __init__.py, admin.py, apps.py, models.py, tests.py, and views.py. The 'views.py' file is selected. The Editor pane shows the code for the 'index' view in 'views.py'. The code imports 'HttpResponse' from 'django.http' and defines a function 'index(request)' that returns an 'HttpResponse' with the text 'Hello, world. You're at the polls index.'.

```
mysite > polls > views.py > index
1  from django.http import HttpResponse
2
3
4  def index(request):
5      return HttpResponse("Hello, world. You're at the polls index.")
6
```

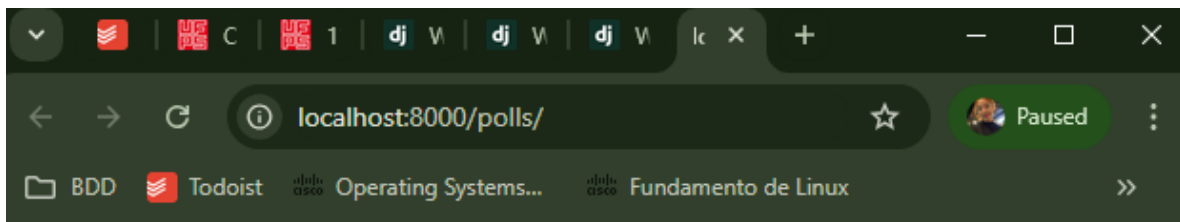
This screenshot shows the Visual Studio Code editor with the Explorer sidebar on the left and the Editor pane on the right. The Explorer sidebar shows the same project structure as the first screenshot, but now 'urls.py' is selected instead of 'views.py'. The Editor pane shows the code for the 'urls.py' file. The code imports 'path' from 'django.urls' and imports 'views' from the current directory. It then defines a list 'urlpatterns' containing a single path pattern: 'path("", views.index, name="index")'.

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



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a project named 'mysite' with a subdirectory 'mysite'. Inside 'mysite', there are files: '__pycache__', '__init__.py', 'asgi.py', 'settings.py', 'urls.py' (selected), 'wsgi.py', and a 'polls' directory. The main editor area shows the content of 'urls.py' with the following code:

```
mysite > mysite > urls.py > ...
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  ]
8
```



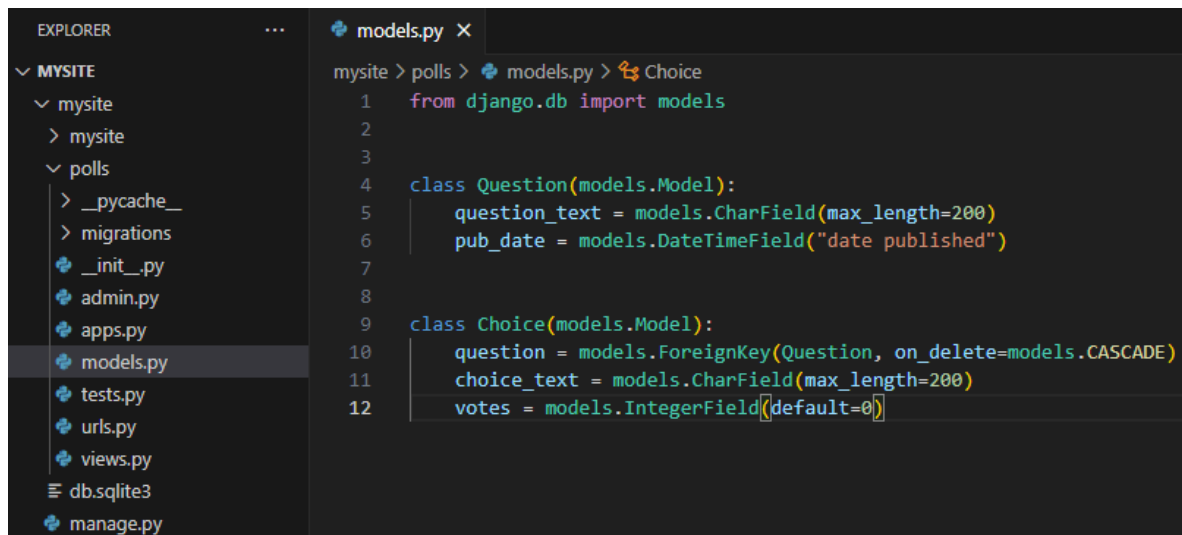
Hello, world. You're at the polls index.

Database setup.

We need to create the tables in the database before we can use them. To do that, we need to run the following command:

```
PS C:\Users\clari\mysite\mysite> py 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.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
```

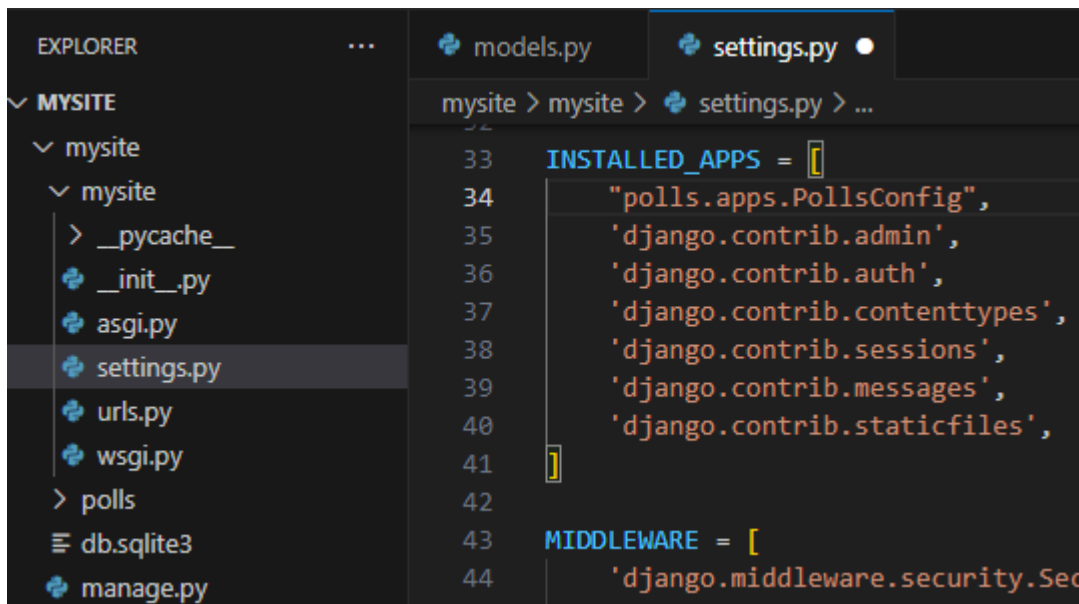
Creating models.



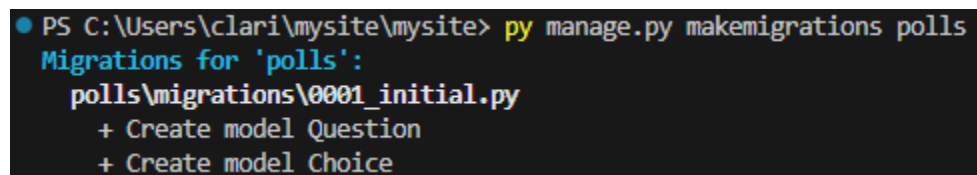
```
models.py
mysite > polls > models.py > Choice
1  from django.db import models
2
3
4  class Question(models.Model):
5      question_text = models.CharField(max_length=200)
6      pub_date = models.DateTimeField("date published")
7
8
9  class Choice(models.Model):
10     question = models.ForeignKey(Question,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

Activating models.

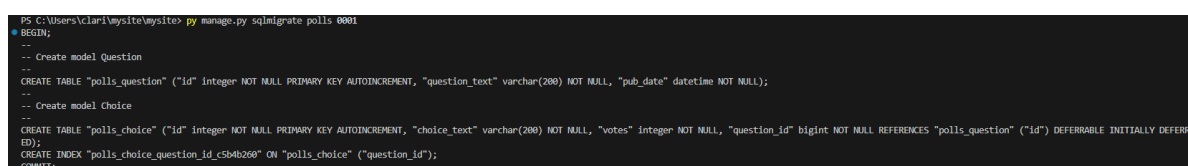
We need to tell our project that the **polls** app is installed.



```
settings.py
mysite > mysite > settings.py > ...
33  INSTALLED_APPS = [
34      "polls.apps.PollsConfig",
35      'django.contrib.admin',
36      'django.contrib.auth',
37      'django.contrib.contenttypes',
38      'django.contrib.sessions',
39      'django.contrib.messages',
40      'django.contrib.staticfiles',
41  ]
42
43  MIDDLEWARE = [
44      'django.middleware.security.SecurityMiddleware',
45      'django.contrib.sessions.middleware.SessionMiddleware',
46      'django.middleware.common.CommonMiddleware',
47      'django.middleware.csrf.CsrfViewMiddleware',
48      'django.contrib.auth.middleware.AuthenticationMiddleware',
49      'django.contrib.messages.middleware.MessageMiddleware',
50  ]
```



```
PS C:\Users\clari\mysite\mysite> py manage.py makemigrations polls
Migrations for 'polls':
  polls\migrations\0001_initial.py
    + Create model Question
    + Create model Choice
```



```
PS C:\Users\clari\mysite\mysite> py manage.py sqlmigrate polls 0001
BEGIN;
-- Create model Question
CREATE TABLE "polls_question" ("id" Integer NOT NULL PRIMARY KEY AUTOINCREMENT, "question_text" varchar(200) NOT NULL, "pub_date" datetime NOT NULL);
-- Create model Choice
CREATE TABLE "polls_choice" ("id" Integer NOT NULL PRIMARY KEY AUTOINCREMENT, "choice_text" varchar(200) NOT NULL, "votes" Integer NOT NULL, "question_id" bigint NOT NULL REFERENCES "polls_question" ("id") DEFERRABLE INITIALLY DEFERRED);
CREATE INDEX "polls_choice_question_id_c54b2690" ON "polls_choice" ("question_id");
COMMIT;
```

```

● PS C:\Users\clari\mysite\mysite> py manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, polls, sessions
Running migrations:
  Applying polls.0001 initial... OK

```

Remember the three-step guide to making model changes:

Change your models (in models.py).

Run python manage.py makemigrations to create migrations for those changes.

Run python manage.py migrate to apply those changes to the database.

Playing with the API

```

○ PS C:\Users\clari\mysite\mysite> py manage.py shell
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from polls.models import Choice, Question
>>> Question.objects.all()
<QuerySet []>
>>> from django.utils import timezone
>>> q = Question(question_text="What's new?", pub_date=timezone.now())
>>> q.save()
>>> q.id
1
>>> q.question_text
'What's new?'
>>> q.pub_date
datetime.datetime(2024, 9, 25, 23, 24, 57, 620562, tzinfo=datetime.timezone.utc)
>>> q.question_text = "What's up?"
>>> q.save()
>>> Question.objects.all()
<QuerySet [<Question: Question object (1)>]>

```

<Question: Question object (1)> isn't a helpful representation of this object. Let's add an `__str__()` method to both `Question` and `Choice`.

```
EXPLORER
mysite
mysite
  > __pycache__
  > __init__.py
  > asgi.py
  > settings.py
  > urls.py
  > wsgi.py
polls
  > __pycache__
  > migrations
  > __init__.py
  > admin.py
  > apps.py
  > models.py
  > tests.py
  > urls.py
  > views.py
  > db.sqlite3
  > manage.py

models.py
mysite > polls > models.py > ...
1 import datetime
2 from django.db import models
3 from django.utils import timezone
4
5 class Question(models.Model):
6     question_text = models.CharField(max_length=200)
7     pub_date = models.DateTimeField("date published")
8     def __str__(self):
9         return self.question_text
10    def was_published_recently(self):
11        return self.pub_date >= timezone.now() - datetime.timedelta(days=1)
12
13
14 class Choice(models.Model):
15     question = models.ForeignKey(Question, on_delete=models.CASCADE)
16     choice_text = models.CharField(max_length=200)
17     votes = models.IntegerField(default=0)
18     def __str__(self):
19         return self.choice_text
20
```

```
>>> from polls.models import Choice, Question
>>> Question.objects.all()
<QuerySet [Question: What's up?]>
>>> Question.objects.filter(id=1)
<QuerySet [Question: What's up?]>
>>> Question.objects.filter(question_text__startswith="What")
<QuerySet [Question: What's up?]>
>>> from django.utils import timezone
>>> current_year = timezone.now().year
>>> Question.objects.get(pub_date__year=current_year)
<Question: What's up?>
>>> Question.objects.get(id=2)
Traceback (most recent call last):
  File "<console>", line 1, in <module>
  File "C:\Users\clari\AppData\Local\Programs\Python\Python312\Lib\site-packages\django\db\models\manager.py", line 87, in manager_method
    return getattr(self.get_queryset(), name)(*args, **kwargs)
  File "C:\Users\clari\AppData\Local\Programs\Python\Python312\Lib\site-packages\django\db\models\query.py", line 649, in get
    raise self.model.DoesNotExist(
polls.models.Question.DoesNotExist: Question matching query does not exist.
```

```
>>> Question.objects.get(pk=1)
<Question: What's up?>
>>> q = Question.objects.get(pk=1)
>>> q.was_published_recently()
True
>>> q = Question.objects.get(pk=1)
>>> q.choice_set.all()
<QuerySet []>
>>> q.choice_set.create(choice_text="Not much", votes=0)
<Choice: Not much>
>>> q.choice_set.create(choice_text="The sky", votes=0)
<Choice: The sky>
>>> c = q.choice_set.create(choice_text="Just hacking again", votes=0)
>>> c.question
<Question: What's up?>
>>> q.choice_set.all()
<QuerySet [Choice: Not much, Choice: The sky, Choice: Just hacking again]>
>>> q.choice_set.count()
3
```

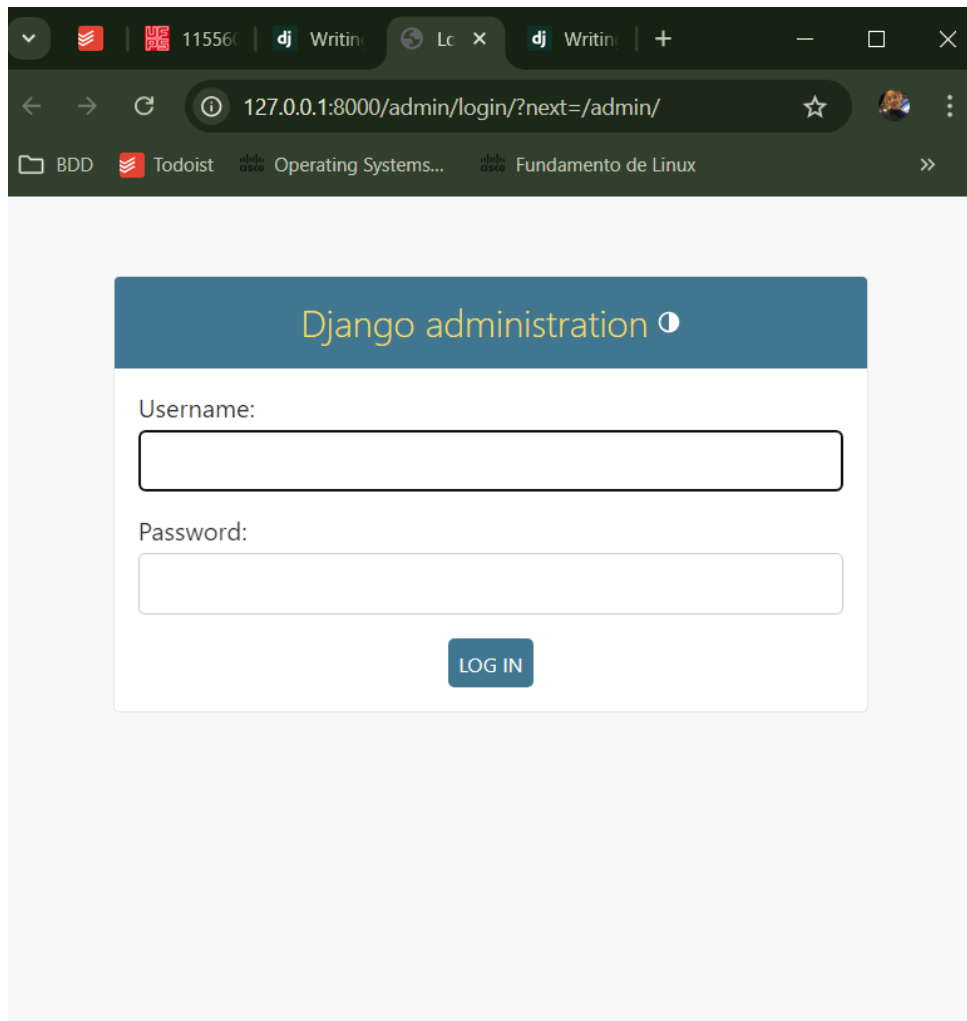
```
>>> Choice.objects.filter(question__pub_date__year=current_year)
<QuerySet [<Choice: Not much>, <Choice: The sky>, <Choice: Just hacking again>]>
>>> c = q.choice_set.filter(choice_text__startswith="Just hacking")
>>> c.delete()
(1, {'polls.Choice': 1})
>>> Choice.objects.filter(question__pub_date__year=current_year)
<QuerySet [<Choice: Not much>, <Choice: The sky>]>
```

Introducing the Django Admin

Creating an admin user.

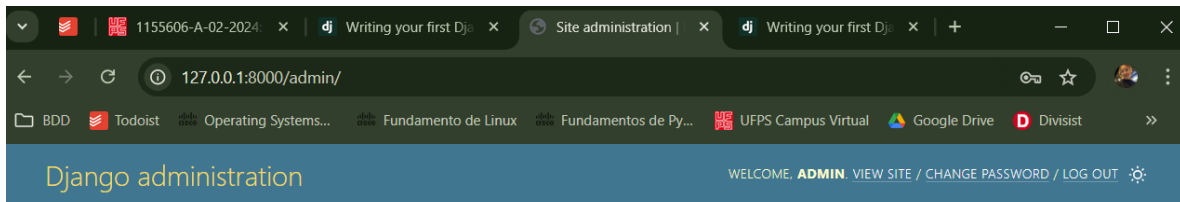
```
PS C:\Users\clari\mysite\mysite> py manage.py createsuperuser
• Username (leave blank to use 'clari'): admin
  Email address: clara.porras08@gmail.com
  Password:
  Password (again):
  Superuser created successfully.
```

Start the development server.



The screenshot shows a web browser window with the address bar displaying `127.0.0.1:8000/admin/login/?next=/admin/`. The browser's tab bar shows several tabs, including 'dj Writin...'. The main content area displays the Django administration login interface. It features a blue header with the text 'Django administration' and a small circular icon. Below the header is a white box containing the login form. The form has two input fields: 'Username:' and 'Password:'. The 'Username:' field is currently empty. Below the 'Password:' field is a 'LOG IN' button.

Enter the admin site.



Site administration

AUTHENTICATION AND AUTHORIZATION

Groups

+ Add Change

Users

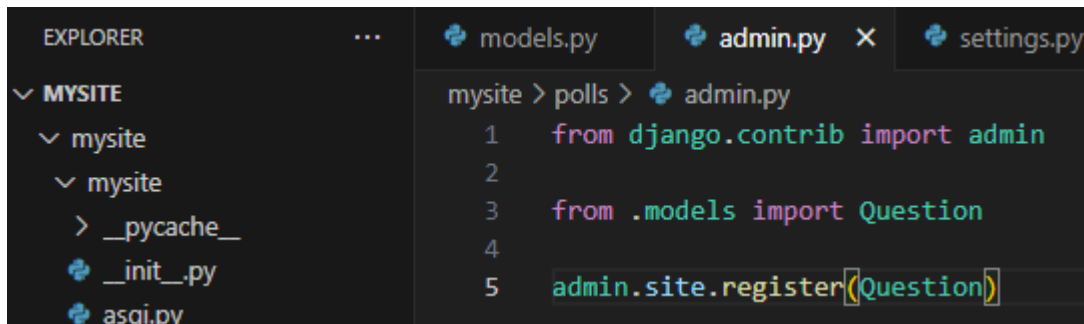
+ Add Change

Recent actions

My actions

None available

Make the poll app modifiable in the admin



Explore the free admin functionality

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups

+ Add Change

Users

+ Add Change

POLLS

Questions

+ Add Change

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups

+ Add

Users

+ Add

POLLS

Questions

+ Add

Select question to change

Action:

Go

 0 of 1 selected

☐ QUESTION

☐ What's up?

1 question

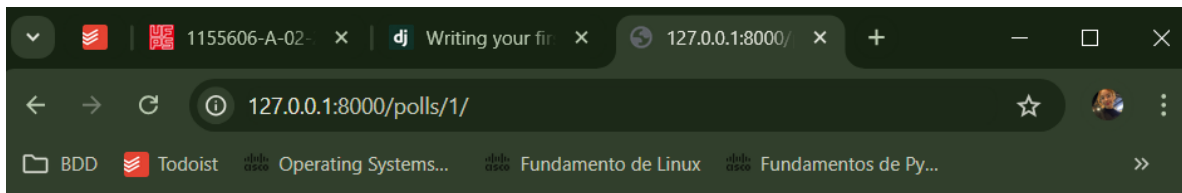
Change history: What's up?

DATE/TIME	USER	ACTION
Sept. 25, 2024, 7:37 p.m.	admin	Changed Date published.
1 entry		

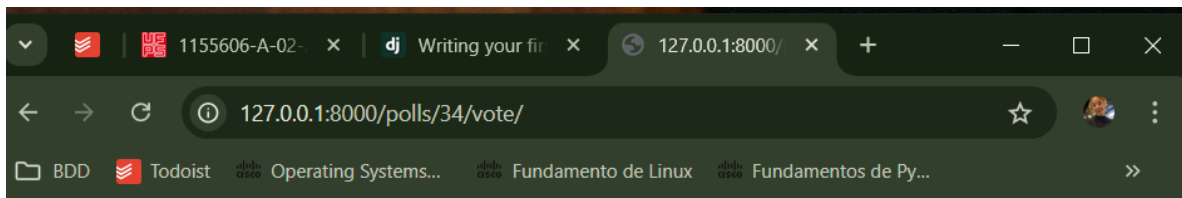
Writing more views

```
models.py  admin.py  settings.py  views.py  X
mysite > polls > views.py > ...
1  from django.http import HttpResponse
2
3
4  def index(request):
5      return HttpResponse("Hello, world. You're at the polls index.")
6
7
8  def detail(request, question_id):
9      return HttpResponse("You're looking at question %s." % question_id)
10
11
12 def results(request, question_id):
13     response = "You're looking at the results of question %s."
14     return HttpResponse(response % question_id)
15
16
17 def vote(request, question_id):
18     return HttpResponse("You're voting on question %s." % question_id)
```

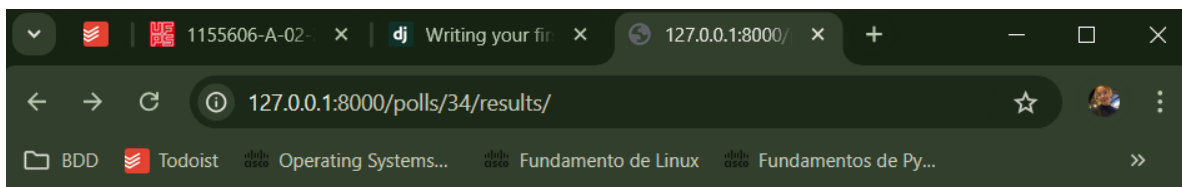
```
models.py  admin.py  settings.py  views.py  urls.py  X
mysite > polls > urls.py > ...
1  from django.urls import path
2
3  from . import views
4
5  urlpatterns = [
6      # ex: /polls/
7      path("", views.index, name="index"),
8      # ex: /polls/5/
9      path("<int:question_id>/", views.detail, name="detail"),
10     # ex: /polls/5/results/
11     path("<int:question_id>/results/", views.results, name="results"),
12     # ex: /polls/5/vote/
13     path("<int:question_id>/vote/", views.vote, name="vote"),
14 ]
```



You're looking at question 1.

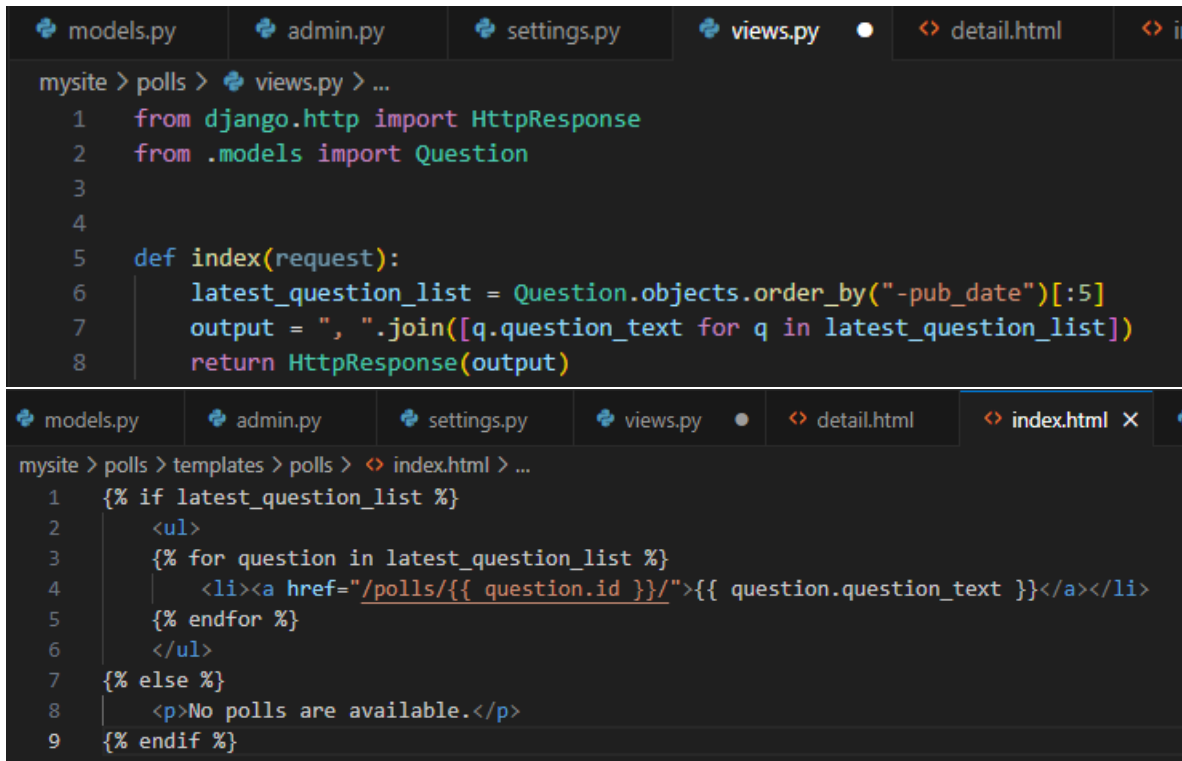


You're voting on question 34.



You're looking at the results of question 34.

Write views that actually do something



Update our index view in polls/views.py to use the template.

```
models.py  admin.py  settings.py  views.py  X  detail.html  <
mysite > polls > views.py > ...
1  from django.http import HttpResponse
2  from django.shortcuts import render
3  from django.template import loader
4  from .models import Question
5
6
7  def index(request):
8      latest_question_list = Question.objects.order_by("-pub_date")[:5]
9      template = loader.get_template("polls/index.html")
10     context = {
11         "latest_question_list": latest_question_list,
12     }
13     return HttpResponse(template.render(context, request))
```

A shortcut: render().

```
models.py  admin.py  settings.py  views.py  X  detail.html
mysite > polls > views.py > ...
1  from django.http import HttpResponse
2  from django.shortcuts import render
3  from .models import Question
4
5
6  def index(request):
7      latest_question_list = Question.objects.order_by("-pub_date")[:5]
8      context = {"latest_question_list": latest_question_list}
9      return render(request, "polls/index.html", context)
```

Raising a 404 error.

```
models.py  admin.py  settings.py  views.py  X  detail.html  <
mysite > polls > views.py > detail
1  from django.http import HttpResponse, Http404
2  from django.shortcuts import render
3  from .models import Question
4
5
6  def index(request):
7      latest_question_list = Question.objects.order_by("-pub_date")[:5]
8      context = {"latest_question_list": latest_question_list}
9      return render(request, "polls/index.html", context)
10
11
12  def detail(request, question_id):
13      try:
14          question = Question.objects.get(pk=question_id)
15      except Question.DoesNotExist:
16          raise Http404("Question does not exist")
17      return render(request, "polls/detail.html", {"question": question})
18
```

```
settings.py  views.py  detail.html  X
mysite > polls > templates > polls > detail.html
1  {{ question }}
```

A shortcut: `get_object_or_404()`.

```
settings.py  views.py  X  detail.html
mysite > polls > views.py > detail
1  from django.http import HttpResponse
2  from django.shortcuts import render, get_object_or_404
3  from .models import Question
4
5
6  def index(request):
7      latest_question_list = Question.objects.order_by("-pub_date")[:5]
8      context = {"latest_question_list": latest_question_list}
9      return render(request, "polls/index.html", context)
10
11
12  def detail(request, question_id):
13      question = get_object_or_404(Question, pk=question_id)
14      return render(request, "polls/detail.html", {"question": question})
15
```

Use the template system.

```
mysite > polls > templates > polls > detail.html > ul
1 <h1>{{ question.question_text }}</h1>
2 <ul>
3 {% for choice in question.choice_set.all %}
4     <li>{{ choice.choice_text }}</li>
5 {% endfor %}
6 </ul>
```

Removing hardcoded URLs in templates.

```
mysite > polls > templates > polls > index.html > ul
1 {% if latest_question_list %}
2     <ul>
3     {% for question in latest_question_list %}
4     <li><a href="{% url 'detail' question.id %}">{{ question.question_text }}</a></li>
5     {% endfor %}
6     </ul>
7 {% else %}
8     <p>No polls are available.</p>
9 {% endif %}
```

Namespacing URL names.

```
mysite > polls > urls.py > ...
1 from django.urls import path
2
3 from . import views
4 app_name = "polls"
5 urlpatterns = [
6     # ex: /polls/
7     path("", views.index, name="index"),
8     # ex: /polls/5/
9     path("<int:question_id>/", views.detail, name="detail"),
10    # ex: /polls/5/results/
11    path("<int:question_id>/results/", views.results, name="results"),
12    # ex: /polls/5/vote/
13    path("<int:question_id>/vote/", views.vote, name="vote"),
14 ]
```

settings.py views.py detail.html index.html X urls.py

mysite > polls > templates > polls > index.html > ...

```
1 {% if latest_question_list %}
2     <ul>
3         {% for question in latest_question_list %}
4         <li><a href="{% url 'polls:detail' question.id %}">{{ question.question_text }}</a></li>
5         {% endfor %}
6     </ul>
7 {% else %}
8     <p>No polls are available.</p>
9 {% endif %}
```