
بسم الله

PART 3

HOSSEIN FORGHANI

MAKTAB SHARIF

Contents

- Creating forms
- Implementing “vote” page of our “polls” example using forms
- Generic views
- Using generic views in our “polls” example

Django Form

Write a minimal form

because the act of submitting this form will alter data server-side

To protect you against Cross Site Request Forgeries

The value of "choice" will be the selected choice id

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>
```

Write a real version of vote view

polls/views.py

```python
from django.http import HttpResponse, HttpResponseRedirect
from django.shortcuts import get_object_or_404, render
from django.urls import reverse

from .models import Choice, Question
# ...
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 += 1
        selected_choice.save()
        return HttpResponseRedirect(reverse('polls:results', args=(question.id,)))
```

dictionary-like object that lets you access submitted data by key name, also request.GET

To avoid having to hardcode a URL in the view function

You should always redirect after successfully dealing with POST data

“results” view

- After somebody votes in a question, the vote() view redirects to the results page for the question
- Let’s write that view:

polls/views.py

```
from django.shortcuts import get_object_or_404, render


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

“results” template

- Now, create a polls/results.html template:

polls/templates/polls/results.html

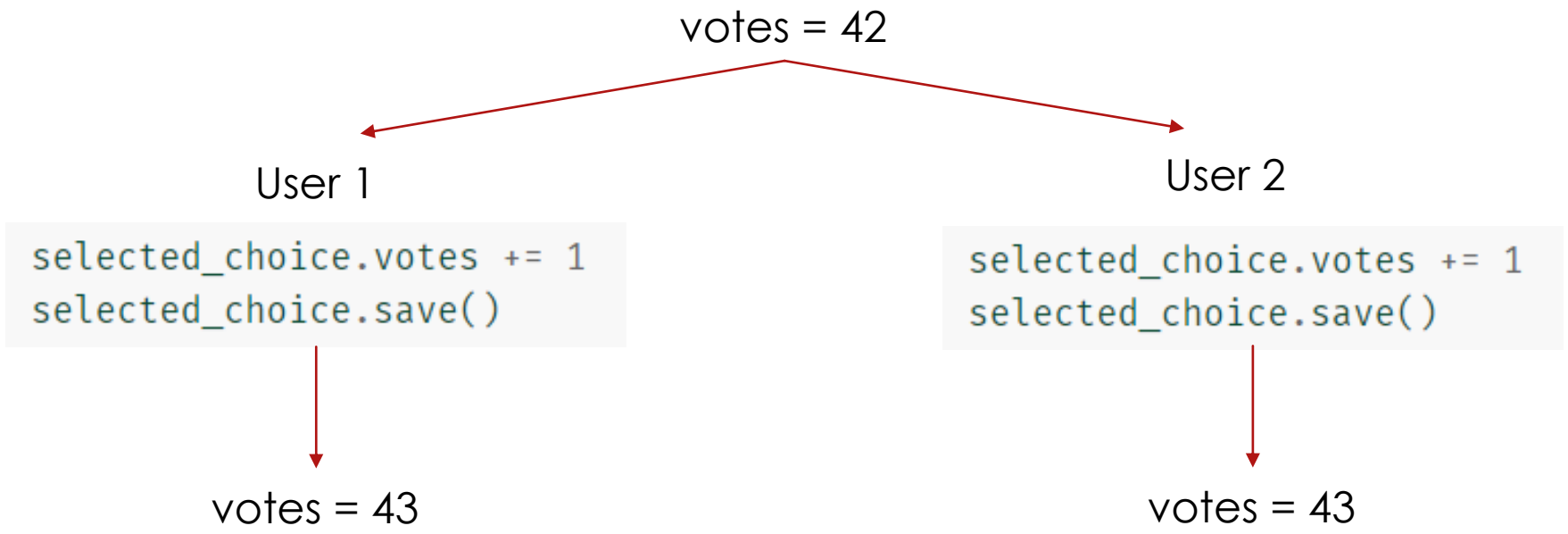
```
<h1>{{ question.question_text }}</h1>

<ul>
{% for choice in question.choice_set.all %}
    <li>{{ choice.choice_text }} -- {{ choice.votes }}
vote{{ choice.votes|pluralize }}</li>
{% endfor %}
</ul>

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

Race Condition

- Remind this code in vote view:



But must be 44 !

Avoid Race Condition

- Avoid race condition using F()

```
# selected_choice.votes += 1
selected_choice.votes = F('votes') + 1
selected_choice.save()
```

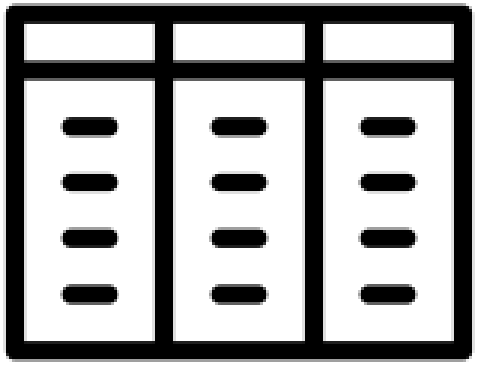
- Read more about F(): https://docs.djangoproject.com/en/3.1/ref/models/expressions/#f-expressions

Generic Views

LESS CODE IS BETTER

Similar Views

- Some similar views:

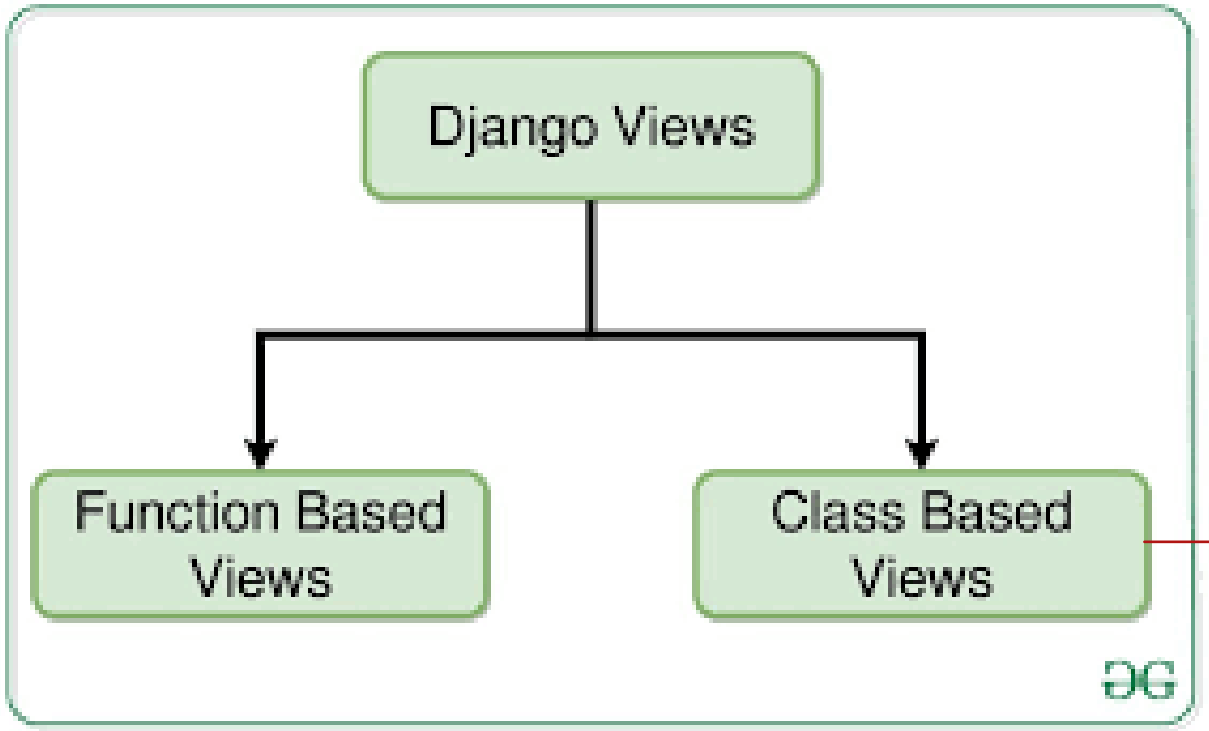


Lists some data

Displays details of an object of a model

Generic Views

- Django provides a shortcut, called the “generic views” system



polls/views.py

```python
from django.http import HttpResponseRedirect
from django.shortcuts import get_object_or_404, render
from django.urls import reverse
from django.views import generic

from .models import Choice, Question


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]


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


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


def vote(request, question_id):
    ... # same as above, no changes needed.
```

A class extending generic.ListView instead of index function

Specify template to render

Specify context object name of iterable objects

Classes extending generic.DetailView instead of datail and results functions

Specify the model from which we want to get an object

Specify template to render

Some views cannot be generic!

polls/urls.py

```
from django.urls import path

from . import views

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

Reference to the generic view

Notice it must be exactly:
`<int:pk>`

References

- https://docs.djangoproject.com/en/3.1/intro/tutorial04/
- https://docs.djangoproject.com/en/3.1/topics/class-based-views/

Any Question?