Django 3: Forms & Class-Based Views

IN608: Intermediate Application Development Concepts

Kaiako: Tom Clark & Grayson Orr

Last Session’s Content

- View
- Template

Today’s Content

- Forms
- Class-based views

Forms

Forms

- Create appropriate `Choice` objects for each `Question` object in Django admin site

Forms

- polls/urls.py

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

app_name = 'polls'

urlpatterns = [
    path('', views.index, name='index'), # /polls/
    path('<int:question_id>/', views.detail, name='detail'), # /polls/2/
    path('<int:question_id>/results/', views.results, name='results'), # /polls/2/results/
    path('<int:question_id>/vote/', views.vote, name='vote'), # /polls/2/vote/
]
```

Forms

- Update `polls/templates/polls/detail.html`
- What is happening here?
  - Displays a radio button for each question choice
  - The value of each radio button is the question choice's ID
  - The form's action & method is set
  - `forloop.counter` indicates how many times the `for` tag has gone through its loop
  - We will talk more about `csrf_token` later in the course

```
<h1>{{ question }}</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 }}</label><br>
    {% endfor %}
    <input type="submit" value="Vote">
</form>
```

Forms

- In the `polls/templates/polls/` directory, create an `.html` file called `results`

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

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

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

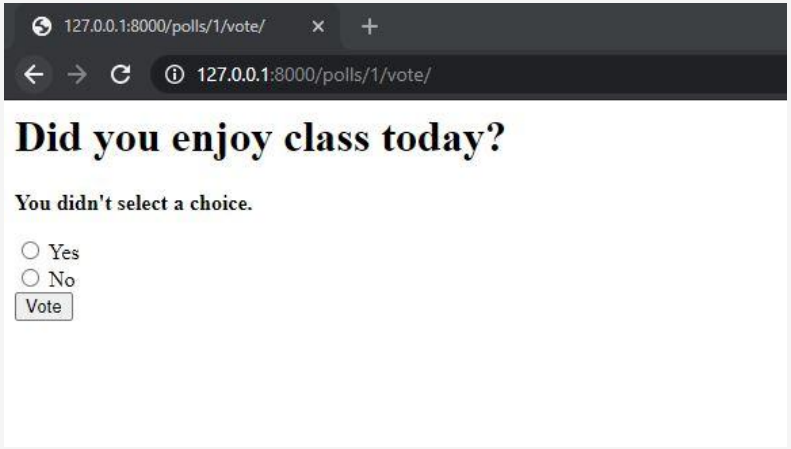
Forms

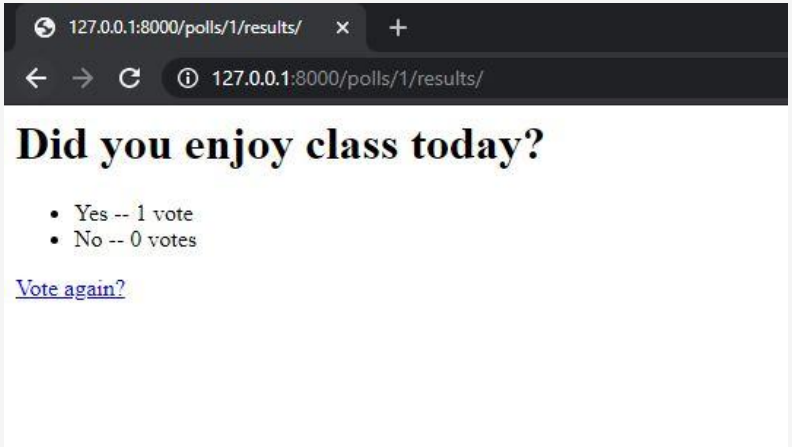
- `polls/views.py`
- What is happening here?
  - `request.POST` is a dictionary-like object which lets you access submitted data by its key name
  - `request.POST['choice']` will raise `KeyError` exception if `choice` is not provided in POST data
  - Returns an `HttpResponseRedirect` preventing data from being posted twice if a user clicks the browser's go back button
  - `reverse()` - avoids having to hardcode a URL in the view function

```
from django.http import HttpResponseRedirect
from django.shortcuts import get_object_or_404, render
from django.urls import reverse
from .models import Question, Choice

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

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

Forms


127.0.0.1:8000/polls/1/
127.0.0.1:8000/polls/1/
Did you enjoy class today?
Yes
No
Vote


127.0.0.1:8000/polls/1/vote/
127.0.0.1:8000/polls/1/vote/
Did you enjoy class today?
You didn't select a choice.
Yes
No
Vote


127.0.0.1:8000/polls/1/results/
127.0.0.1:8000/polls/1/results/
Did you enjoy class today?
Yes -- 1 vote
No -- 0 votes
Vote again?

Class-Based Views

Class-Based Views

- A way to implement views as Python objects instead of functions
- Reusable components using inheritance & mixins (multiple inheritance)
- We are going to refactor some code 😢
- Resource: https://docs.djangoproject.com/en/3.0/topics/class-based-views

Class-Based Views

- polls/urls.py

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

app_name = 'polls'

urlpatterns = [
    path('', views.IndexView.as_view(), name='index'), # /polls/
    path('<int:pk>/', views.DetailView.as_view(), name='detail'), # /polls/2/
    path('<int:pk>/results/', views.ResultsView.as_view(), name='results'), # /polls/2/results/
    path('<int:question_id>/vote/', views.vote, name='vote'), # /polls/2/vote/
]
```

Class-Based Views

- polls/views.py

```
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 Question, Choice

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')

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):
    question = get_object_or_404(Question, pk=question_id)
    try:
        selected_choice = question.choice_set.get(pk=request.POST['choice'])
    except (KeyError, Choice.DoesNotExist):
        context = {
            'question': question,
            'error_message': 'You didn\'t select a choice.',
        }
        return render(request, 'polls/detail.html', context)
    else:
        selected_choice.votes += 1
        selected_choice.save()
        return HttpResponseRedirect(reverse('polls:results', args=(question.id,)))
```

Practical

Programming Activity

- Checkout to master - `git checkout master`
- Create a new branch called 09-practical - `git checkout -b 09-practical`
- Copy 09-practical.ipynb from the course materials repository into your practicals repository
- Open up the Anaconda Prompt (it should be install on all lab computers) & `cd` to your practicals repository
- Run the following command: `jupyter notebook`