# Django 3: Forms & Class-Based Views

**IN608: Intermediate Application Development Concepts** 

Kaiako: Tom Clark & Grayson Orr

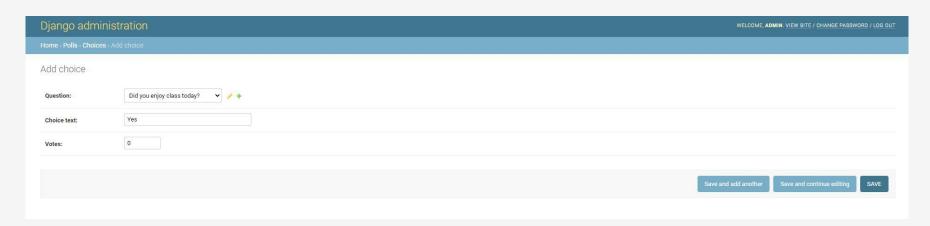
### **Last Session's Content**

- View
- Template

## Today's Content

- Forms
- Class-based views

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



• polls/urls.py

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

app_name = 'polls'

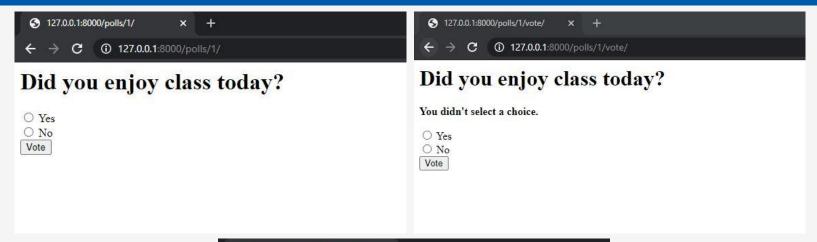
urlpatterns = [
    path('', views.index, name='index'), # /polls/
    path('<int:question_id>/', views.detail, name='detail'), # /polls/2/
    path('<int:question_id>/results/', views.results, name='results'), # /polls/2/results/
    path('<int:question_id>/vote/', views.vote, name='vote'), # /polls/2/vote/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

- 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
  - o 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

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

- polls/views.py
- What is happening here?
  - request.POST is a dictionary-like object which lets you access submitted data by its key name
  - o 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
  - o 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,)))
```





- 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 (2)
- Resource: <a href="https://docs.djangoproject.com/en/3.0/topics/class-based-views">https://docs.djangoproject.com/en/3.0/topics/class-based-views</a>

• 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/
]
```

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
- Open the file 09-practical.pdf and work on the tasks described