

# Django 2: View & Template

**IN608: Intermediate Application Development Concepts**

**Kaiako: Tom Clark & Grayson Orr**

# Last Session's Content

- Django
  - Overview
  - Pipenv
  - Installation
  - Creating a project
  - Development server
  - Creating an app
- MVC vs. MVT
- Route
- Model
- Admin Site

# Today's Content

- View
- Template

**View**

# View

- Type of web page in your app which serves a function & has a template
- Each view is represented by a Python function or method (class-based views)
- A view is chosen by examining the requested URL
  - Django uses a URLconf to get from a URL to a view
  - A URLconf maps URL patterns to views
- Resource: <https://docs.djangoproject.com/en/3.0/topics/http/views/#writing-views>

# View

- `polls/urls.py`
- When someone request a page, for example, `/polls/2/`, Django will load the `mvt.urls` Python module - pointed to by the `ROOT_URLCONF` setting
- It finds the variable name `urlpatterns` & traverses the patterns in order
- After finding a match at `polls/`, it strips off the matching text `polls/` & sends `2/` to the `polls.url` URLconf for further processing
- Results in a call to the `detail()` view
- Namespacing URL names
  - Larger Django projects may have more than one app
  - How does Django differentiate between them?
  - Add an `app_name` to set the application namespace

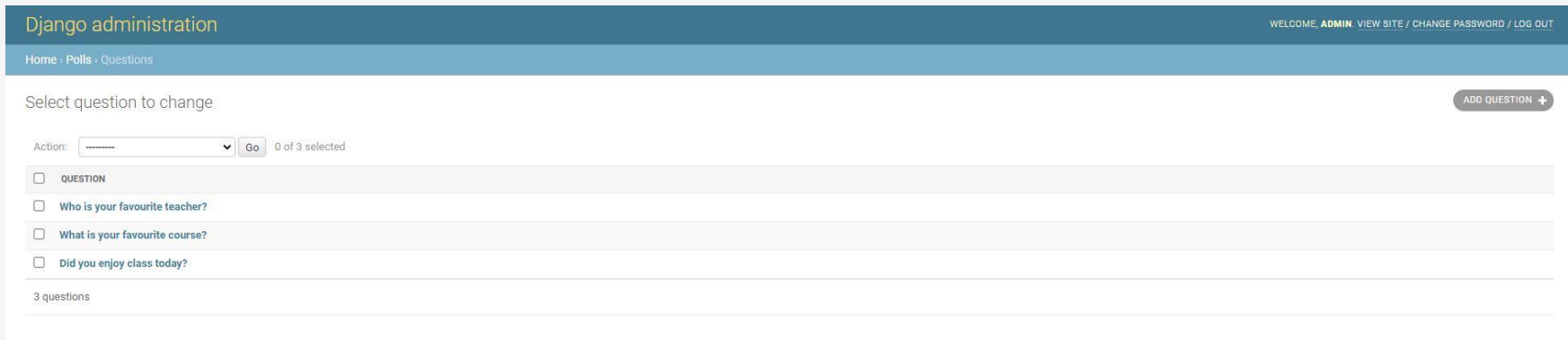
```
from django.urls import path
from . import views

app_name = 'polls'

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

# View

- Let's assume, we have created three `Question` objects in Django admin site



The screenshot displays the Django administration interface for managing 'Questions'. The top navigation bar is dark blue with the text 'Django administration' on the left and 'WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT' on the right. Below this, a breadcrumb trail reads 'Home > Polls > Questions'. The main content area has a heading 'Select question to change' and an 'ADD QUESTION +' button. An 'Action:' dropdown menu is set to '-----' with a 'Go' button and a status '0 of 3 selected'. A table lists three questions, each with a checkbox and a link to edit: 'Who is your favourite teacher?', 'What is your favourite course?', and 'Did you enjoy class today?'. At the bottom, it states '3 questions'.

Django administration

WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Polls > Questions

Select question to change

ADD QUESTION +

Action: ----- Go 0 of 3 selected

<input type="checkbox"/>	<a href="#">QUESTION</a>
<input type="checkbox"/>	<a href="#">Who is your favourite teacher?</a>
<input type="checkbox"/>	<a href="#">What is your favourite course?</a>
<input type="checkbox"/>	<a href="#">Did you enjoy class today?</a>

3 questions

# View

- `polls/views.py`
- `Question.objects.order_by('-pub_date')` - get all **Question** objects & order by `pub_date` (desc)
- `render()`
  - Combine a given template with a given context dictionary
  - Return an `HttpResponse` object with that rendered text
- `get_object_or_404()`
  - Call `get()` on a given model manager
  - Raise `Http404` instead of the model's `DoesNotExist` exception
- Resources:
  - <https://docs.djangoproject.com/en/3.0/topics/db/queries>
  - <https://docs.djangoproject.com/en/3.0/topics/http/shortcuts/#render>
  - <https://docs.djangoproject.com/en/3.0/topics/http/shortcuts/#get-object-or-404>

```
from django.shortcuts import get_object_or_404, render
from .models import Question

def index(request):
    latest_question_list = Question.objects.order_by('-pub_date')
    context = {'latest_question_list': latest_question_list}
    return render(request, 'polls/index.html', context)

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



# Template

# Template

- What is a template?
  - A text file, i.e., HTML, XML, CSV, etc.
  - Contains variables which get replaced with values when the template is evaluated
  - Contains tag which control the logic of the template
  - It should look very similar to EJS template language you saw in the Fundamentals of Web Development course
- In the `polls/` directory, create a directory called `templates/`
- In the `templates/` directory, create a directory called `polls/`
- In the `polls/` directory, create two `.html` files called `index` & `detail`
- Resource: <https://docs.djangoproject.com/en/3.0/ref/templates/language>
- App structure:

```
polls/
  __init__.py
  admin.py
  apps.py
  migrations/
    __init__.py
  models.py
  templates/
    polls/
      index.html
      detail.html
  tests.py
  views.py
```

# Template

- polls/templates/polls/index.html
- If latest\_question\_list is not empty, loop through latest\_question\_list & display each **Question** object's question\_text as a link
- Run the development server & navigate to <http://127.0.0.1:8000/polls>

```
{% if latest_question_list %}
    <ul>
    {% for question in latest_question_list %}
        <li><a href="{% url 'polls:detail' question.id %}">{{ question }}</a></li>
    {% endfor %}
    </ul>
{% else %}
    <p>No polls are available.</p>
{% endif %}
```

# Template

- `polls/templates/polls/detail.html`
- Display `Question` object's `question_text`
- Click the **What is your favourite course?** link
- `Question` object's id is 2

```
{{ question }}
```

