CSCI 2133

Rapid Programming Techniques for Innovation

Lab 4:

Python/Django 4 There is a Template for Everyone

Lab Instructor: Colin Conrad

Faculty of Computer Science

Dalhousie University

Working with Django: The Basics

- Today we are going to start bringing together the work we have done the previous three weeks. We will focus on making the front end logic work on our core Django app.
- We will be working directly with the week4 folder, so please get that ready in Git
- There will be a LOT of copy-paste this week. Please be patient.

Step 1: Pull from Github

- Get set by using `git pull`
- Configure the folder to get it to say ``‘‘hello world’’``
- Visit `http://127.0.0.1:8000/` in your browser
- Don’t forget to `python manage.py migrate`

Step 2: Preparing Views

- Open `profileGrabber/views.py`.
- Make it look as follows.

```
from django.shortcuts import render, get_object_c

from .models import SavedUser

def index(request):
    context = {}
    return render(request, ’profileGrabber/index.

def discover(request):
    profile = "cd_conrad"
    influencer = "NO"
    context = {
        ’profile’: profile,
        ’influencer’: influencer,
    }

    return render(request, ’profileGrabber/discov

def collection(request):
    saved_users = SavedUser.objects.all()

    context = {
        ’saved_users’: saved_users,
    }
    return render(request, 'profileGrabber/colle
```

```
from __future__ import unicode_literals

from django.apps import AppConfig


class ProfilegrabberConfig(AppConfig):
    name = 'profileGrabber'
```

```
INSTALLED_APPS = [
    'profileGrabber.apps.ProfilegrabberConfig',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

Step 3: Configure the Templates

- Modify the html views that you created in week 2 so Django can read them.
- Pay attention to the special tags that tell the framework to talk to the views file.

```
profiler
-profileGrabber
--templates
---discover.html
---index.html
...

from django.conf.urls import url
from . import views

urlpatterns = [
    url(r'^$', views.index, name='index'),
    url('discover.html', views.discover, name='di
]
```

Step 4: Make the Questions

- Pay attention to the special tags that tell the framework to talk to the views file.
- You will have a blank profile feature, if successful. This is because we have not yet configured the api (next week!).

```
<div class="row">
  <div class="col m3">
    <h4>{{ profile.name }}</h4>
    <span style="font-weight:bold;">{{ profile.sc
    <span>{{ profile.location }}</span>
  </div>
  <div class="col m3">
    <h4>Influencer: {{ influencer }}</h4>
    <!-- Modal Trigger -->
    <a class="waves-effect waves-light btn green'

    <!-- Modal Structure -->
    <div id="modal1" class="modal">
      <div class="modal-content">
        <h4>Influencers</h4>
        <p>Influencers are people who start trend
                Twitter. Typically these users ha

followers, and a large number of people actually

post. We can create any number of algorithms to c
```

```
but we kept it simple for this tutorial: people w

followers.</p>

        </div>
        <div class="modal-footer">
          <a href="#!" class=" modal-action modal-c
        </div>
      </div>
    </div>
    <div class="col m3">
      <h4>Followers: {{ profile.followers_count }}<
      <span style="font-weight:bold;">Following:</s
      <span>{{ profile.friends_count }}</span>
    </div>
    <div class="col m3">
      <form action="" method="post">
        {% csrf_token %}
        <input type="hidden" name="push_source" val
        <input type="hidden" name="push_handle" val
        <button class="btn-floating btn-large waves
            <i class="material-icons">add</i>
        </button>
      </form>
    </div>
  </div>
  <div class="row">
    <div class="col m12">
      <blockquote>
        <p>{{ profile.description }}</p>
        <footer>{{ profile.screen_name }}<cite titl
      </blockquote>
    </div>
  </div>
```

Step 5: Configure Static

- The static folder is used by Django to read custom static code such as JavaScript or images
- It helps keep things organized and is usually contained in the app root folder.

Step 5: Configure Reusable Templates

- Templates are designed to make you code less, not more.
- Creating `template.html` will allow us to reuse code from the document headers and footers, reducing work.
- For simplicity, I provided you with a simple template file. Rename `template.txt` to `template.html` and take a look in your editor.

```
{% extends "profileGrabber/template.html" %}

{% block content %}
```

```
...[the non-header content]


{% endblock %}
```