Assignment: Portfolio Site

Objectives

Build on what you have learned through the Django Tutorial and create a portfolio website to showcase your work. The objective is to build a portfolio you could show a prospective employer or to show off your work to your friends.

A Portfolio app is not much more than an image gallery app. Start a new Django project for this assignment.

Learn how to use ImageFields and static files for your app.

Learn how to use images in your templates. Learn how to use thumbnail filters.

What your program must have

1. A portfolio is made up of one or more case studies
2. The main portfolio page shows thumbnail images of your case studies in a grid format
3. A case study detail page with one or more images and a description of the project you’re showing off
4. Use the Django Admin to load case studies into your portfolio
5. The portfolio site needs to look awesome. Please use stylesheets to lay things out nicely.

Extra Credit Ideas

- Arrange your case studies into categories or tags
- JavaScript Slideshow showcasing your case studies (in addition to the grid layout of thumbnails)
- Please come up with your own ideas on what you would like to see for your own portfolio site.

Hints

- To being a new django project, activate your virtual environment and change to the directory you want to do your work in, and type `django-admin startproject myproject` where myproject is the name of the project you want to start
- Follow these steps to building out your program:
  1. First edit your settings.py
  2. Create your models.
  3. Make sure you add your app to INSTALLED_APPS in settings.py
  4. Run manage.py makemigrations and then migrate.
  5. Give yourself a superuser account for the admin: manage.py createsuperuser
  6. Edit your urls.py for the app you’re writing
  7. Work on your views while setting up your templates
  8. Edit your project’s master urls.py to include the urls for your app
  9. manage.py runserver to test your site

- For images, use the models.ImageField. See https://docs.djangoproject.com/en/1.8/ref/models/fields/#filefield and https://docs.djangoproject.com/en/1.8/ref/models/fields/#imagefield for how to set this up. Note: you’ll need to edit your settings.py to accommodate file uploads. See settings relating to media path.

- Check in your code often! The steps are usually:
  - git add
  - git commit -m “a message describing your commit”
  - git push

- Install pillow, the Python Imaging Library by typing the following after you’ve activated your virtual environment:
  
  `pip install -I Pillow`

- Serving static media is simple with a few lines of code. In settings.py:
and add these two lines to your list of Context Preprocessors list under the TEMPLATES section:

```
"django.core.context_processors.media",
"django.core.context_processors.static",
```

and when referring to static media in your template, prefix it with `{{ MEDIA_URL }}`.

.. and in your urls.py, follow the suggestion from this page: https://docs.djangoproject.com/en/1.8/howto/static-files/#serving-files-uploaded-by-a-user-during-development

- Install sorl-thumbnail. This provides template tags that will allow you to easily thumbnail images on the fly.
  - `pip install sorl-thumbnail`
  - Add sorl.thumbnail to INSTALLED_APPS in settings
  - To make troubleshooting easier, add the line THUMBNAIL_DEBUG = True to the bottom of your settings.py file
  - `python manage.py migrate` to set up the key value store in the database for things that sorl thumbnail needs
  - Use the example on this page to thumbnail your images: https://github.com/mariocesar/sorl-thumbnail
  - With sorl.thumbnail you don't need `{{ MEDIA_URL }}` in your templates any more.
  - Use Twitter Bootstrap CSS framework to get a good looking site with minimal effort (http://getbootstrap.com)