Homework 5: Social Network Models

Due date: February 19, 2020 at 11:59pm

This homework is the second in a series of homeworks in which you will build an increasingly sophisticated nano-blogging site.

For this assignment, you will create the data models used to store posts and supplementary user profile information. You will use these data models to implement the ability for users to create & edit their profiles, make new blog posts, view blog posts, and follow other users. Again, we recommend reading this document in its entirety before starting on your homework.

The learning goals for this assignment are to:

- Demonstrate mastery of many learning goals from the previous assignment, including all of the learning goals noted from earlier in the semester.
- Gain experience with sophisticated data management, including data models with complex relationships and the use of an ORM to execute queries using those relationships.
- Achieve integration of data models with:
    - modular, reusable views with inheritance,
    - form classes to encapsulate input validation, and
    - image (or, in general, file) upload.

Specification

This section describes the enhancements you will make to your social network application. To begin your assignment, start by copying over your project from the last homework over to the `hw5` folder (see *Turning in your work*, below, for the directory structure).

Effective use of Django Models and Forms

Make the following enhancements to your templates for your social network:

1. Create a data model for posts called `Post`. Ensure your model has all the needed fields and declare these fields to be of an appropriate field type. (E.g., dates should not be stored in a character field.)
2. Create a data model for the enhanced profile called `Profile`. Specifically, you need to create a model to store the user bio and profile image. This model will also need a field to reference the Django authentication module's `User` model (so you will know which user the extended profile data is for).

3. Add a field to the enhance profile model, above, to keep track of the follower relationships. (Use an appropriate model field to keep track of who is following whom.)

4. Create form or model-form objects to process posts and profile creation/edits. Be sure to validate input parameters using these form (or model-form) objects.

5. Create routing and actions (in `urls.py` and `views.py`) to implement:
   - post creation,
   - global stream viewing,
   - profile creation/editing,
   - profile viewing,
   - following,
   - unfollowing, and
   - follower stream viewing.

   Display posts in reverse-chronological order (newest ones first). Display times in using `TIME_ZONE = 'US/Eastern'`

6. Eliminate all hard-coded URLs in your Python and templating code by using reverse URL resolution. Particularly, this means usage of the `{% url 'location' %}` template tag as well as the `reverse()` function in Python.

Implementation hints

You can update your templated views from `hw4`, as necessary to fix any problems you encounter, but you should generally keep the design of your site the same. Certainly, the id attributes specified in the `hw4` spec should still be present on all the pages.

You must now make posts and profiles work, as described in `hw4`. In particular, clicking on the name of the user that created a post should now show that user's profile page and permit the logged-in user to follow the posting user (or unfollow this user, if the logged-in user is already following). However, when viewing the currently logged-in user's profile, you must show, in addition to the current bio and photo, a form to allow the bio and photo to be edited (or created). You must also show, on the logged-in user's profile page, a list of the other users that the logged-in user is following. That list includes links to the followed users' profile pages.

Your UI should still show the fields/buttons to permit comments to be made on posts, but you do not need to (i.e., should not) make this work, beyond not crashing if someone tries to make a comment. You do not need to show any sample comments for this homework.

You will want to take a look at the following guides on the Django reference:
- Relationship fields (i.e. `ForeignKey`, `OneToOneField`, `ManyToManyField`)
- Making queries ("Retrieving specific objects with filters" and "Lookups that span relationships" might be very useful)

Requirements

Your application must also follow these requirements:

- You must meet all requirements specified in the previous assignment, including in particular:
  - The empty URL (i.e. http://localhost:8000) must route to the first page of your application.
  - Your application should not use any hard-coded absolute paths.
  - Your application should run with **Django 3.0**.
  - Your application should use the default Django database configuration based on a SQLite database file (named `db.sqlite3`) in your project directory.
  - Your application should not crash as a result of any input sent to the server-side or because of any actions the user performs.

- Your application should use template inheritance, reverse URL resolution, as well as complete validation of client-submitted data with Django Forms.

- You should not commit into your repo your database file, migrations, or any user-uploaded images. Update your `.gitignore` file accordingly (to prevent committing uploaded images). It's OK to commit default profile images or other images used by your site (which should be in be static folder). The TAs (or grading script) will "`makemigrations`" and "`migrate`" before running your homework. Your image (media) folder should be someplace under the `hw5` folder (just as in the course example) and that directory should be specified in your `.gitignore`.

- Cite all external resources used and any additional notes in the `README.md` file.

Committing your work

As with the previous homeworks, we will be evaluating your version control usage. Keep in mind that good version control usage typically means (1) incremental, modular commits with (2) descriptive and useful commit messages.

Validation

As with previous assignments, any client request (achievable or not by your user interface) must not be able to crash the application. Additionally, we are looking to see if you correctly use Django Forms for input validation.

Coverage of technologies

You must demonstrate effective use of the introduced technologies of this assignment.

- ORM usage (model design, data manipulation, and data retrieval)
- image upload and display
- (and form objects, which you may not have used to prepare dummy data in `hw4`)

Turning in your work

Your submission should be turned in via Git and should consist of a Django application in the `hw5` directory. Name your project **webapps** and the application **socialnetwork**. The directory structure will look somewhat like the following (some files/directories omitted):

```
[YOUR-ANDREW-ID]/hw5/
    |-- webapps/
        |-- settings.py
        |-- urls.py
        |-- [etc.]
    |-- socialnetwork/
        |-- static/
        |-- templates/
        |-- forms.py
        |-- models.py
        |-- views.py
        |-- [etc.]
    |-- manage.py
    |-- README.md
```