Homework 4: Extending grumblr

Due date: October 4, 2018

This homework is one in a series of homeworks in which you will build an increasingly sophisticated nano-blogging site. **grumblr** will eventually be a featureful, interactive web application including user registration and authentication, email integration for user verification, photo upload, and quasi-real-time updates.

For this assignment you will extend the application you built in the previous assignment, adding features such as the ability to edit and enrich your profile, follow other users, upload images, and send email. As in previous assignments, we recommend reading this document in its entirety before starting 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.
- Gain familiarity with features common to modern web application frameworks, including:
    - developing modular, reusable view templates with template inheritance,
    - using form classes to encapsulate input validation,
    - image upload (or file upload in general), and
    - sending email as part of an application.

Specification

This section describes the enhancements you will make to **grumblr**. To begin your assignment, start by copying your project from the previous homework to the `homework/4` directory.

Refactoring your previous solution

After you have copied your previous solution, refactor your application to eliminate code duplication in your Django templates, using Django template inheritance. After you have eliminated code duplication in your templates, refactor your application to validate all user input using Django Forms. This means that you will not *manually* validate request parameters as in the early course demos, but you will instead use the Django-provided validation functions such as `Form.is_valid()`[1]. Neither of these refactoring steps should change the behavior of your application, only details of its implementation.

[1] See https://docs.djangoproject.com/en/2.1/ref/forms/api/#using-forms-to-validate-data for some examples.

New **grumblr** features

After you refactor your application as described above, add the following features:

- Profiles for logged-in users should include at least the following information:
  - first name
  - last name
  - age
  - short bio (420 characters or less)
- Logged-in users can edit their profile information and change their password.
- Logged-in users can upload a profile image, which is displayed on their profile page and next to their posts as a small image. The image upload form should be located on the profile-editing page.
- Logged-in users may choose to 'follow' and 'unfollow' another user.
- Logged-in users may view a 'follower stream', displaying all posts from the users that the current user is following (in reverse-chronological order).
- New users must register with, and confirm, their e-mail address.
- Users can reset their password by a link sent to their registered e-mail address.

Implementation hints

You might end up adding something like the following pages:

1. An **edit profile** page; displays a form that allows the current user to edit their first name, last name, age, and short bio, as well as upload a profile image.
2. A **follower stream** page; lists posts from all users that the current user follows.

You might want to re-familiarize yourself with the following resources:

- 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 meet these requirements:

- You must meet **all** requirements specified in the previous assignment, including:
  - 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 2.1.x and Python 3.
  - 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.

- All tasks and features in the specification must be easily accomplishable using the user interface for your social network.
- Cite all external resources used and any additional notes you would like to convey to your grader in the `README.md` file. It is helpful if you provide your Python version in the `README.md` file.

Grading criteria

For substantial credit your solution must clearly demonstrate the learning goals for this assignment, which are described above in the introduction.

Committing your work (10 pts)

As with the previous homeworks, we will evaluate your version control use. Good version control use typically means (1) incremental, modular commits with (2) descriptive and useful commit messages.

Fulfilling our specification (30 pts)

Your submission must follow all specifications and requirements introduced in the previous sections.

Form-based validation (20 pts)

As with previous assignments, any client request (with or without using your user interface) must not crash your application. You must correctly use Django Forms for all input validation.

Appropriate usage of Django technologies (60 pts)

You must demonstrate effective use of the introduced technologies of this assignment, including:

- Template inheritance and reverse URL resolution
- Image upload and display
- Sending email as part of a web application

User interface design (0 pts)

Turning in your work

Your submission must be turned in via Git and should consist of a Django application in the `homework/4` directory. Name your project **webapps** and the application **grumblr**. The directory structure might resemble:

```
[YOUR-ANDREW-ID]/homework/4/
     webapps/
        settings.py
        urls.py
        [etc.]
     grumblr/
        static/
            grumblr/
        templates/
            grumblr/
        models.py
        views.py
        [etc.]
     manage.py
     db.sqlite3
     README.md
```