Introduction

For my second coursework I have created a simple Twitter like application. The application allows for the user to register using their e-mail address and provide a username and password. This allows them to post to the website and all messages are displayed on the index page for the entire internet to view. There is no limit to the characters therefore could be used as a public viewed blog or users could limit themselves to the 140 characters twitter enforces.

The tools used to build the application were Python Flask, Jinja2, HTML and CSS.

The libraries used were Flask-Login, Peewee and the extension WTForms

Design

The application uses two libraries Flask login and Peewee. Both libraries add the fundamental functionality to the application. Flask Login allows to create and store the active user's ID which gives the logging in/out functionality. By using this library I could also restrict what logged out users could view i.e. the post message page, in its place there is a highlighted "Register" link.

The application uses pewee to define the user and post class

WTForms allowed me to validate the user's information before sending it to the database. It can detect if a valid e-mail format is being used, the correct characters for usernames (A-Za-z0-9_) and if the password had been inserted correctly twice by the user. It also stops any data from being submitted if a form is missing data. WTForms is also used on the "Post Message" page, a message cannot be displayed without any input. If the user tries to click Post! Without any data it will display an error.

Peewee allowed me to created classes for the data that was to be inserted into the database and to save the data input to the forms to the database.

Enhancements

In order to make the application twitter-like it would need profiles pages and a way to search for relevant posts. Profiles would contain a profile picture that would be used as an avatar beside each of their listed post. This would be used to differentiate from other users rather than relying on a text based username.

A search function would also prove to be very useful as it would allow for user's to search for real time events.

The functionality of the application is limited but can be easily built on.

Critical evaluation

One of the main issues I encountered was the installation of the bcrypt library which resulted in the passwords being passed and validated in plain text. I could not install the library using either "easy_install" or via pip as the package could not find an installation of gcc. Multiple attempts were made to install flask-bcrypt, bcrypt or py-bcrypt. One solution I came across was to install the python

development tools but I couldn't find this required package either. This resulted in giving up and having to do with plain text passwords and continue with developing the application.

Personal Evaluation

While developing the first coursework I decided to use Bootstrap for the first time – this allowed me to give the application an aesthetically pleasing design without having to code the CSS from scratch. This intrigued me and led to researching some of the libraries that are available for flask. Implementing libraries saves a lot of time and each of the libraries are heavily documented on their website.

This coursework has taught me how important it is to research available libraries before attempting to code the project from scratch.

I found developing with python-flask enjoyable and would definitely consider using it for future projects. The large amount of libraries available can give an application basic features with little effort, allowing you to spend more time developing the unique features within the application

Resources

Flask Login: https://flask-login.readthedocs.org/en/latest/

Peewee: http://docs.peewee-orm.com/en/latest/

Introduction to Peewee: http://www.blog.pythonlibrary.org/2014/07/17/an-intro-to-peewee-another-python-orm/

Advanced Web Technologies Workbook

CSS Template: http://templated.co/privy