Open-Source Technology Use Report

Flask Login

General Information & Licensing

Code Repository	https://github.com/maxcountryman/flask-login.git
License Type	MIT License
License Description	• Fairly “permissive license requiring preservation of copyright and license notices” • otherwise allowing usage and distribution without any limitations • Following permissions are allowed: - Commercial use - Modification - Distribution - Private use
License Restrictions	• Does not provide any warranty or liability • Licensed works, modifications, and larger works may be distributed under different terms and without source code.
Who worked with this?	Ahsan

LoginManager (class)

Purpose

This class allows for behind the scenes session management of users by storing their session information in a User object. This library also allows us to make certain routes on the server only accessible if a user is currently logged in.

flask_server.py – lines 20 to 22:

- this is where the LoginManager class is instantiated which will be used to keep track of session management

flask_server.py – lines 91 to 110:

- this is where more information about the current user is stored, with methods that LoginManager uses to verify if a certain user is authenticated, if they are active, and their username given their session information

flask_server.py – lines 114 to 115:

- this is a callback function that the library uses to return a User object which is used by LoginManager, with the username of that user attached

Magic ⋆★˚｡°.☽˚⁀➷｡˚★彡⋆ ༄

- The way that this library works is by using a User class, that is defined in *flask_server.py*. When a user first registers through the HTML form on the /register route, the *load_user* function is called with their username, which creates a User object (as described in the purpose section above).
- This function takes one parameter, which is the username that the user inputted as a string.
- The result of this is stored as a variable called *user*, which is then used to actually login, which will be described in more detail in the next section

login_user (method)

Purpose

This function sets authentication cookies for the user, which allows them to access pages on the app that are only for logged in users.

flask_server.py – line 68:

- this is where the login_user method is called. The first parameter is the user object that was created in the previous section

Magic ⋆★.˚ ⋆｡˚☽˚｡⋆ ⭒˚｡⋆★ ⋆ ༄

- The way that this library works when you look at the bigger picture is by using session cookies. When a user first registers through the HTML form on the /register route, the *load_user* function is called with their username, which creates a User object (as described in the LoginManager section above).
- Then, the *login_user* function is called with the aforementioned User object that was just created. This function is located in the *utils.py* file at lines 169 to 217.
    - https://github.com/maxcountryman/flask-login/blob/ecd3b59339175e575ba598eb5c5fd3330e0ff73b/src/flask_login/utils.py#L169
    - the second parameter of this function is a Boolean called *remember*, which when set to True, will remember the user after their session expires
    - the third parameter of this function is a *datetime.delta* object called *duration*, which specifies the amount of time before the authenticated session cookie expires
- The *login_user* function calls the *_get_user* function on line 216 in the *utils.py* file
    - https://github.com/maxcountryman/flask-login/blob/ecd3b59339175e575ba598eb5c5fd3330e0ff73b/src/flask_login/utils.py#L216
- The *_get_user* function calls the *_load_user* function on line 383 in the *utils.py* file
    - https://github.com/maxcountryman/flask-login/blob/ecd3b59339175e575ba598eb5c5fd3330e0ff73b/src/flask_login/utils.py#L383
- The *_load_user* function is located in the *login_manager.py* file. On line 369, it sets the cookie
    - https://github.com/maxcountryman/flask-login/blob/ecd3b59339175e575ba598eb5c5fd3330e0ff73b/src/flask_login/login_manager.py#L369
- On line 484, the cookie is actually set
    - https://github.com/maxcountryman/flask-login/blob/ecd3b59339175e575ba598eb5c5fd3330e0ff73b/src/flask_login/login_manager.py#L484

logout_user (method)

Purpose

This function deletes the authentication cookie for the user, which disables access to the pages of the app where being logged in is required

flask_server.py – line 87:
- this is where the logout_user method is called.

Magic ⋆★˚｡⋆˚☽˚⁀➷｡˚★彡⋆ ༄

- The way this works is by first calling *logout_user* which is located in utils.py on line 220
  - https://github.com/maxcountryman/flask-login/blob/ecd3b59339175e575ba598eb5c5fd3330e0ff73b/src/flask_login/utils.py#L220
- This function first gets the cookie on line 237
  - https://github.com/maxcountryman/flask-login/blob/ecd3b59339175e575ba598eb5c5fd3330e0ff73b/src/flask_login/utils.py#L237
- Then, it clears the cookie (if it was set) on line 239
  - https://github.com/maxcountryman/flask-login/blob/ecd3b59339175e575ba598eb5c5fd3330e0ff73b/src/flask_login/utils.py#L239
- Finally, it sends an HTTP request without that cookie on line 243
  - https://github.com/maxcountryman/flask-login/blob/ecd3b59339175e575ba598eb5c5fd3330e0ff73b/src/flask_login/utils.py#L243

@flask_login.login_required (method)

Purpose

This annotation allows us to make certain routes/pages on our app secure, by checking if a user is logged in using the authentication cookies that were being used as part of the *flask_login* method

flask_server.py – line 24:

- this is one example of making our home page (/ route) secure by only allowing logged in/authenticated users to access it

Magic ⋆★˚｡⋆˚☽˚⁀➷｡˚★彡⋆ ༄

- The way that this works is by calling the *login_required* method in utils.py on line 259
  - https://github.com/maxcountryman/flask-login/blob/ecd3b59339175e575ba598eb5c5fd3330e0ff73b/src/flask_login/utils.py#L259
- This function then calls the *is_authenticated* function that was defined as part of the LoginManager class, which was written by us, on line 297.
  - https://github.com/maxcountryman/flask-login/blob/ecd3b59339175e575ba598eb5c5fd3330e0ff73b/src/flask_login/utils.py#L297
- This returns either True or False depending on if the user is logged in (in which case the user object would exist as part of LoginManager, or if the user is not logged in, in which case the user object would not exist. This is on line 105 of *flask_server.py*
  - https://github.com/rdstrach/CSE312_Group_Project/blob/c942a36ce3c93ffaaa39a48910b38e9ca6c0523e/flask_server.py#L105