FlaskWTF

General Information & Licensing

Code Repository	https://github.com/wtforms/flask-wtf/
License Type	BSD-3 Clause License
License Description	Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met.
License Restrictions	• Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. • Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. • Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
Who worked with this?	Collin

CSRFProtect Class

Purpose

- This class allows us to easily generate and validate CSRF tokens for our forms to add security to our web application.
- This tech is located on line 18 of app.py where our csrf object is defined. The CSRF tokens are generated when our html is created, which is handled by the CSRFProtect class. The tokens are generated in line 34 of login.html, line 34 of register.html and line 46 of settings.html

Magic

- This technology when the app is created globally enables CSRF protection for our flask app. From there, the CSRF object looks for {{ csrf_token() }} inside of our html templates inside our flask app to replace and generate a csrf token in place of that. These tokens are then checked when the form is sent back to the server to make sure they match. If they don't match an error is sent back instead.
- Line 168 in csrf.py (https://github.com/wtforms/flask-wtf/blob/3c91c7bda9fe1b553fc4c6f4cb217f959e967617/src/flask_wtf/csrf.py) sets up the CSRFProtect class entirely. From there, the object is looking for the {{ csrf_token }} in the html templates in our flask app. If the class object finds that, the generate_csrf_token function on line 146 is called. This function generates a CSRF token and puts it in place of the {{ csrf_token }} in our html. Then, once the form is returned to the server from the user, the protect function on line 256 is called. From there the _get_csrf_token() function on line 231 is called and the csrf tokens are validated with the validate_csrf function located on 66. If the two tokens don't match an error is sent back for our flask app to handle, otherwise the form data is accepted with no issues.

CSRFError Class

Purpose

Replace this text with some that answers the following questions for the above tech:

- This tech allows us to handle a 400 Bad Request response if the CSRF data is tampered with before being sent back to the server.
- This tech is specifically used on line 54 of app.py with the @app.errorhandler(CSRFError).

Magic

Dispel the magic of this technology. Replace this text with some that answers the following questions for the above tech:

- This technology does what is specified in our purpose section as FlasksWTF directly integrates with our Flask app upon the initial setup. Because of this, we are able to use the CSRFError flag to handle CSRFError and send a 400 Bad Request Response if this is the case. We use a flask error handler here so that we can send back a custom html page for any tampered forms.
- The code that handles this is located in line 310 located in csrf.py (https://github.com/wtforms/flask-wtf/blob/3c91c7bda9fe1b553fc4c6f4cb217f959e967617/src/flask_wtf/csrf.py) accomplishes the task at hand with the CSRFError class at the bottom. This class object will be created whenever a BadRequest is received when an invalid CSRF token is received. The error_response function in line 306 creates a CSRFError class whenever it is called upon. The protect function located in line 256 checks all CSRF form data and will call the error_response function creating a CSRFError object whenever a form error is found. This will then call the error handler we have set up in our app.py in line 54.