[Flask]

General Information & Licensing

Code Repository	https://github.com/pallets/flask
License Type	BSD-3-Clause License
License Description	• The BSD-3-Clause license applies to all files in the Flask repository and source distribution. This includes Flask's source code, the examples, and tests, as well as the documentation. • 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?	Hyukjoo, Changho, Brian, Timothy

[app = flask.Flask(__name__)]

Purpose

The Flask class passes the parameter __name__ which allows the class to find resources on the file system. Flask uses folders such as templates and statics to store and reference html and css files. Therefore, passing in __name__ allows the class to find files in the directory in reference to the file where the class is initiated.

This class is initiated in the 4th line of the server.py file as the route function in the class will be used to route directories for the website.

Magic

The Flask class can be found at https://github.com/pallets/flask/blob/main/src/flask/app.py from line 98 to the end of the file. The class can take parameters but are in the form of the Scaffold class; which can be found at https://github.com/pallets/flask/blob/ea93a52d7d94ba093bbce4680c622cc4fc9771d8/src/flask/scaffold.py#L751 from lines 62 to the end of the file. When only 1 parameter is passed, __name__, this class uses the passed file directory to find the other folder; such as templates and statics for uses stated above.

Beyond this the class itself does not have any specific functionality until they are called.

Chain: https://github.com/pallets/flask/blob/main/src/flask/app.py -> https://github.com/pallets/flask/blob/ea93a52d7d94ba093bbce4680c622cc4fc9771d8/src/flask/scaffold.py#L751

[@app.route()]

Purpose

The route function is called from the Flask class as a decorator and is used to route users to specific directories. Then the decorated function will be called.

This function is called several times in server.py for each directory that is created. When a user goes to a directory that does not exist a route function is linked to all non existing directories. Here a not found message will be displayed.

Magic

The .route() function is called from a Flask class instance and is always in the form of a decorator. The code can be found at https://github.com/pallets/flask/blob/main/src/flask/app.py from lines 1037 to 1094. This function takes the input string, which indicates the path, and optional POST/GET methods - if not stated it will default to get. In line 1047 endpoint is called which was a function call that is passed as a parameter that can be found at https://github.com/pallets/flask/blob/ea93a52d7d94ba093bbce4680c622cc4fc9771d8/src/flask/wrappers.py#L62 from line 62 to 74. This file imports https://github.com/pallets/werkzeug/blob/main/src/werkzeug/serving.py which uses the socket library and returns the HttpRequest(more details on the werkzeug import in the app.run() function Magic). The endpoint function parses the returned HttpRequest which will then be used to check with the .route() parameter to determine if the appropriate function should be run.

Chain: https://github.com/pallets/flask/blob/main/src/flask/app.py -> https://github.com/pallets/flask/blob/ea93a52d7d94ba093bbce4680c622cc4fc9771d8/src/flask/wrappers.py#L62 -> https://github.com/pallets/werkzeug/blob/main/src/werkzeug/serving.py

[flask.render_template()]

Purpose

The render_template function is called from the Flask class and is used to render an html template. This function looks for the html file in the templates folder. This is possible because the class was initiated with the directory of the called file.

This function is called every time it is necessary to render a pre-made html.

Magic

The function can be found at https://github.com/pallets/flask/blob/ea93a52d7d94ba093bbce4680c622cc4fc9771d8/src/flask/templating.py from lines 133 to 151.

The render_template function takes the name of an html file and looks for it in the templates folder. This can be done thanks to the pathing as explained in the Flask(__name__) class. This function calls the _render() function which is in line 124 of the same file. This function calls the send() function at https://github.com/pallets/flask/blob/ea93a52d7d94ba093bbce4680c622cc4fc9771d8/src/flask/signals.py#L25 from lines 25 to 37. This function takes the rendered html and sends it to the server with the proper formatting.

Chain:
https://github.com/pallets/flask/blob/ea93a52d7d94ba093bbce4680c622cc4fc9771d8/src/flask/templating.py ->
https://github.com/pallets/flask/blob/ea93a52d7d94ba093bbce4680c622cc4fc9771d8/src/flask/templating.py ->
https://github.com/pallets/flask/blob/ea93a52d7d94ba093bbce4680c622cc4fc9771d8/src/flask/signals.py#L25

[flask.session()]

Purpose

flask.session is an extension for Flask that supports server-side sessions to your application. The session is the time between the clients logging in to the server and logging out of the server. The data that is required to be saved in the session is stored in a temporary directory on the server. Some uses of flask.session are: remember user when they log in, store user specific website settings (theme), store E-Commerce site user items in cart

flask.session is used everywhere throughout the code. The purpose of flask.session is to access the saved data. So whenever the data is needed in the code is where flask.session is used.

Magic ⋆★｡˚⋆˚☽˚｡⋆｡˚★彡⋆༄

We worked with Flask to route the directories. flask.session is used when accessing data is required. flask.session has many different builtin configuration values. Some of which are: SESSION_COOKIE_NAME, SESSION_TYPE, SESSION_PERMANENT, and many more... After the TCP socket is created and after the user is directed to the homepage the code uses flask.session to access the data stored whenever clients log in and out of the server. Session allows programmers to store information specific to a user from one request to the next. This is implemented on top of cookies and signs the cookies cryptographically. In other words, the user could look at the contents of the programmers cookies but they are unable to modify the code, unless the user knows the secret key - which are in bytes - used for signing. Accessing these key and value pairs work like a dictionary. You would use the same functionalities as a dictionary to access the key and values.

Citation: https://flask-session.readthedocs.io/en/latest/

Link: https://github.com/jaykwonproject/CSE312/blob/registration_fuctionality/server.py
flask.session is used on lines 35 and 36. In both of these lines flask.session is used to assign a user who has just logged in to the boolean True and to assign a new key and value pair, which are ‘user’->username, into the data

[flask.session.get()]

Purpose

flask.session.get() retrieves data the programmer is requesting for in the data stored in flask.session.

flask.session.get() is used on lines 8, 11, and 14.

Magic

As explained before, flask.session is where data is stored by signing cookies cryptographically. flask.session.get() is used to retrieve data that is requested by the programmer. flask.session.get() iterates through the data to find the key in the parameter. Once the key is found then the function returns the object from the data. If the key isn't found then the function will return NULL.

Citation: https://flask-session.readthedocs.io/en/latest/

Link: https://github.com/jaykwonproject/CSE312/blob/registration_fuctionality/server.py
flask.session.get() is used on lines 8, 11, and 14. In all of these lines flask.session.get() are used to retrieve data from flask.session. In this specific context the code is determining whether the user is logged in or not. If the user is not logged in then the code will require the user to sign in or make an account. If the user is logged in then the homepage will stay as it is.

[flask.redirect()]

Purpose

flask.redirect() returns a response object and redirects the user to another target location with a specified status code.

flask.redirect() is used when the user logs out or if the user is creating a new account. Once the user enters a username the code will redirect to another route to check if the username is available, if the user is already logged in, or the code will redirect the user to the homepage after the user logs out.

Magic

After the homepage is connected we need to go to the login or register pages to use flask.redirect(). flask.redirect() has 3 parameters: flask.redirect(location, code, Response). However, in our code we only included the location so far. flask.redirect() returns a response object (a WSGI) application that redirects the client to the target location (url - flask.url_for() will be explained later). Supported codes are 301, 302, 303, 305, 307, and 308. For example: when a user is logging into his/her account and the login is incorrect flask will redirect the user to the login page asking the user to login again. If the login is correct then the user is redirected to the homepage. Another example is when the user is on the register page. When the user is registering a new account the user must pick a username that doesn't already exist and the user's password must be entered twice. If the user keeps messing this part up then flask will continuously redirect the user to the register page. But if the user successfully made an account then the user is redirected to the login page.

Citation: https://flask.palletsprojects.com/en/2.0.x/api/

Link: https://github.com/jaykwonproject/CSE312/blob/registration_fuctionality/server.py
flask.redirect() is used on lines 33, 37, 54, 58, and 64.

[flask.url_for()]

Purpose

flask.url_for() is accompanied with flask.redirect(). flask.url_for() determines where the route should be sent next.

flask.url_for() is used wherever flask.redirect() is used. flask.redirect() needs to use flask.url_for() to send the user to the appropriate webpage.

Magic

Continuing from where we left off at flask.redirect() in the code. flask.url_for() is the parameter of flask.redirect() and it returns a string of the URL. flask.url_for() generates a URL to the given endpoint with the method provided. Variable arguments that are unknown to the target endpoint are appended to the generated URL as query arguments. If the value of a query argument is None, the whole pair is skipped. Flask has a hook to intercept URL build errors through Flask.url_build_error_handlers. flask.url_for() results in a BuildError when the current app doesn't have a URL. When it does, the current_app calls url_build_error_handlers if it is None, which returns a string of the URL.

Citation: https://flask.palletsprojects.com/en/2.0.x/api/

[flask.request.form()]

Purpose

This function is used to get the data from the html form that contains user input. For example in the login feature. The user will input their username and password and the function will get the corresponding data.

In our code this is called depending on the functionality but is almost always used under functions that need to accept user input.

Magic

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

- How does this technology do what it does for you in the **Purpose** section of this report? Please explain this in detail, starting from after the TCP socket is created. Remember, to be allowed to use a technology in your project, you must be able to know how it works.
- Where is the specific code that does what you use the tech for? You ***must*** provide a link to the specific file in the repository for your tech with a line number or number range.
    - If there is more than one step in the chain of calls *(hint: there will be)*, you must provide links for the entire chain of calls from your code, to the library code that actually accomplishes the task for you.
    - Example: If you use an object of type HttpRequest in your code which contains the headers of the request, you must show exactly how that object parsed the original headers from the TCP socket. This will often involve tracing through multiple libraries and you must show the entire trace through all these libraries with links to all the involved code.

*This section may grow beyond the page for many features.

The request function can be found from https://github.com/pallets/flask/blob/main/src/flask/wrappers.py where it calls the form function from https://github.com/pallets/werkzeug/blob/main/src/werkzeug/wrappers/request.py at lines 413 to 429. In the form function the load_form_data() function is called which is in the same file lines 251 to 277. This function parses the form data from lines 264 to 271. This parsed data is then used by load_form_data() which is then used by request.form().

Chain: https://github.com/pallets/flask/blob/main/src/flask/wrappers.py -> https://github.com/pallets/werkzeug/blob/main/src/werkzeug/wrappers/request.py

[app.run()]

Purpose

App is an instance of the Flask class that was initiated at the start of the program. This function allows the application to be run in a local environment. This function deals with creating the TCP socket connection.

The function is called under the __name__ == __main__ statement at the end of the program. This allows the application to be run and, therefore, allows all the other functionalities to be accessed.

Magic

The code for the function can be found at https://github.com/pallets/flask/blob/main/src/flask/app.py from lines 805 to 924.

For the purpose of running the application lines 917 to 925 are used. In line 917 run_simple is imported from werkzeug.serving and it is run in line 920; passing the host and port as parameters. Werkzeug is a WSGI web application library that allows web servers to forward requests to python web applications. The code for run_simple can be found at https://github.com/pallets/werkzeug/blob/main/src/werkzeug/serving.py from lines 798 to 989. The function uses the socket library from python to create a socket connection that can be seen from line 942 where it calls the server_forever on the server that was made from lines 929 to 939.

Chain: https://github.com/pallets/flask/blob/main/src/flask/app.py -> https://github.com/pallets/werkzeug/blob/main/src/werkzeug/serving.py