Flask is a lightweight python web application framework that allows developers to focus on the web application istself by simplifying the backend. Using the flask framework will help our team in a variety of different ways.

Flask will handle HTTP exceptions for us as well. It will also make the implentation of authentication smoother and simpler with its use encoding of sessions.

It will parse incoming request information for us as well as build response requests for us. This will make serving content to the user much easier.

Flask also allows us to serve static files with little work. This will help for images/css files. (jinja library found in Reports)

Flask uses a previous library called Werkzeug to define its requests and responses as well as its parsing. (description for Werkzeug found in Reports)

We will store the user information on login in a session. This information will be used to send information between sockets and display who is online.

The sessions will also make logout a breeze as we can just pop the user info from the session on logout.

Another thing that flask does for us is both handles and creates TCP connections using Werkzeug. (description for Werkzeug found in Reports)

Flask URL:

LINK URL binder:
https://github.com/pallets/flask/blob/3c9b85469ef4abda1e996a5ec0e7bef0f1a0b394/src/flask/app.py#L1032

LINK view function:
https://github.com/pallets/flask/blob/3c9b85469ef4abda1e996a5ec0e7bef0f1a0b394/src/flask/views.py#L69

Flask will build the url path by using the information given by the user in the decorator. @app.route('/').

This decorator uses add_url_rule(). This binds a url to the function that will decide what occurs when the user accesses that path.

The url rule is parsed and returned into a generator from the Werkzeug library.

Flask also uses url parsing from the Werkzeug library (report for Werkzeug library located in reports)

HTTP Requests:

Link request object:
https://github.com/pallets/flask/blob/3c9b85469ef4abda1e996a5ec0e7bef0f1a0b394/src/flask/wrappers.py#L14

LINK finalize_request:
https://github.com/pallets/flask/blob/29d33203d0325f006c75fc88359872bd68c8bdf5/src/flask/app.py#L1955

LINK dispatching:
https://github.com/pallets/flask/blob/3c9b85469ef4abda1e996a5ec0e7bef0f1a0b394/src/flask/app.py#L1463

https://github.com/pallets/flask/blob/3c9b85469ef4abda1e996a5ec0e7bef0f1a0b394/src/flask/app.py#L1504

LINK Method detection:
https://github.com/pallets/flask/blob/3c9b85469ef4abda1e996a5ec0e7bef0f1a0b394/src/flask/views.py#L105

When flask recieves a request from the server it will store it into a global object to be used elsewhere.

The object that flask uses is from Werkzeug, but adds a couple more attributes of its own.

The attributes that are provided by flask and werkzeug allow for http requests to be easily parsed by the user.

For example, if a user needs to access the arguements passed in by a form they can use request.args (Werkzeug property)

Using finalize_request() flask will finish the http request by converting it into a response object, conduct postprocessing, and then send the response.

Flask will also allow us to easily detect what http method is being used in the request.

Within the decorator, Flask allows us to pick and choose what http method can be used on this path by adding them as arguments in app.route('path',methods =[GET, POST])

Flask will detect what type of request was made and allow us to handle it as needed easily. (quick example: if request.method==GET: do something)

HTTP Responses:

Link Response object:
https://github.com/pallets/flask/blob/3c9b85469ef4abda1e996a5ec0e7bef0f1a0b394/src/flask/wrappers.py#L101

LINK finalize_request:
https://github.com/pallets/flask/blob/29d33203d0325f006c75fc88359872bd68c8bdf5/src/flask/app.py#L1955

LINK dispatching:
https://github.com/pallets/flask/blob/3c9b85469ef4abda1e996a5ec0e7bef0f1a0b394/src/flask/app.py#L1463

https://github.com/pallets/flask/blob/3c9b85469ef4abda1e996a5ec0e7bef0f1a0b394/src/flask/app.py#L1504

LINK make_response:
https://github.com/pallets/flask/blob/3c9b85469ef4abda1e996a5ec0e7bef0f1a0b394/src/flask/app.py#L1616\

LINK render_template:
https://github.com/pallets/flask/blob/3c9b85469ef4abda1e996a5ec0e7bef0f1a0b394/src/flask/templating.py#L130

LINK send file:
https://github.com/pallets/flask/blob/3c9b85469ef4abda1e996a5ec0e7bef0f1a0b394/src/flask/helpers.py#L490

Flask response objects are built off of the response objects from Werkzeug, but flask sets up responses to have the html mimetype by default (line 119 of object link)

Flask uses make_response to create the http response using the value returned by the user at the end of the function. This will allow us to add our own headers to the

response if needed.

Using finalize_request() flask will finish the http request by converting it into a response object, conduct postprocessing, and then send the response.

Using make_response is different then returning render_template.

Render_template() will render the provided file using the JinjaLoader library. (send the html file)

Sending files can be done using send_file which uses utilities from Werkzeug to send the file to the path.

AUTHENTICATION AND SESSIONS

https://github.com/pallets/flask/blob/master/src/flask/sessions.py

Flask uses its own sessions to store data to be used between multiple server requests. Flasks sessions are created using signed cookies.

The session object itself is stored as a dictionary object which will allow you to access your stored data using key value pairs. Sessions are only able to be used if

the flask application has a secret key set.

The session information cannot be changed if the secret key is not known this is because when Flask encodes this cookie with the secret string you provide it into a

base64 encoding.

LISCENSES

Flask is licensed under a BSD License. The copywrite was established in 2015 by Armin Ronacher and contributers.

Any modifications and redistributions of flask are allowed as long as a set of rules are followed.

These rules are as follows, "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.", and "The names of the contributors may not be

used to endorse or promote products derived from this software without specific prior written permission."