Open-Source Report

Proof of knowing your stuff in CSE312

[Flask - Parsing HTTP Headers]

General Information & Licensing

Code Repository	https://github.com/pallets/flask
License Type	BSD 3-Clause License
License Description	This license is similar to the BSD 2-Clause License which contains two clauses. The first clause is that redistribution of the source code must contain a specific copyright notice, a list of conditions and a specific disclaimer. The second clause requires that redistribution that is in binary form must include the same three items mentioned above in the documentation. There is a third clause that comes in with the BSD 3-Clause License which the framework Flask is under. This clause is that 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. The permissions that are allowed with Flask under this license are that it is approved for commercial and/or private use, and is able to be modified and/or distributed. There are limitations with liability and warranty. Essentially, the requirements of this license when copying, modifying or distributing any code is that the full text of the license must be included, along with the original copyright notice (which is actually included in the license). The license must include the year and owner in the form "Copyright (c) $YEAR $OWNER, All rights reserved".
License Restrictions	The BSD 3-Clause License has very minimal restrictions, which is what makes it easier to use and understand since it can be fairly used commercially and privately. A restriction that is does include concerns using the names of contributors to the repository for endorsement when advertising without permission. This is the third clause that was mentioned above, where people using code under the BSD 3-Clause license are unable to include the name of the copyright holder and/or the contributors to endorse any work that was produced from using the open source code under this license. For example, a project created from using Flask cannot be endorsed by the contributors of Flask without special permission.

Magic ⋆★˚₊‧⁺˖⋆ ☽ ˚⌒☄.˚★⚞✧༄

Flask is a framework that is widely used in the development of web applications. Flask is implemented using Werkzeug (a comprehensive WSGI library) and Jinja2 (a templating engine). It is simple to use when compared to other frameworks such as Django, and it provides programmers with back-end components such as tools, libraries and functionalities that are able to efficiently host web servers responding to HTTP requests.

It is important to be able to correctly parse HTTP requests because it allows the clients and the server to communicate
through different types of requests such as GET and POST, and send information that is important to us through headers, such
as Cookies.

1. Our program starts in app.py where the following line below calls the run() function:
   if __name__=="__main__":
   app.run(debug=True,host='0.0.0.0')
   As mentioned in the documentation, the run() function will run the application on a local development server. This function appears in app.py on line 1067:
https://github.com/pallets/flask/blob/066a35dd322f689ec07d7c0e82b19eacadac3c6b/src/flask/app.py#L1067

   However, to begin, we assign the variable app to a flask object implements a WSGI application and acts as the central object, which is accomplished by the line:
   app = Flask(__name__).
   In the provided repository for Flask, this can be found in the app.py file on line 92:
https://github.com/pallets/flask/blob/d0bf462866289ad8bfe29b6e4e1e0f531003ab34/src/flask/app.py#LL92C1-L92C1

2. When we call the run() function on the flask object, a parameter that is included is options. This parameter, as mentioned in the comments are the "the options to be forwarded to the underlying Werkzeug server". Because of Werkzeug, we are able to import a function run_simple: from werkzeug.serving import run_simple. This function is available in Werkzeug library's serving.py on line 945:
https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L907

3. The run_simple() function mentioned above includes a parameter: request_handler: type[WSGIRequestHandler] | None = None.
   This can be found on line 920:
https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L920

   The class WSGIRequestHandler is defined with a parameter BaseHTTPRequestHandler on line 148 of serving.py:
https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L148

   To understand how Flask parses HTTP headers, we will take a closer look at the BaseHTTPRequestHandler which can be found on line 145 of python's server.py file:
https://github.com/python/cpython/blob/bf26bdf6ac04878fc720e78422991aaedb9808a1/Lib/http/server.py#L145

4. The BaseHTTPRequestHandler class includes a function named parse_request()

which parses the HTTP requests provided.
This can be found in the same server.py file mentioned above on line 266:
https://github.com/python/cpython/blob/bf26bdf6ac04878fc720e78422991aaedb9808a1/Lib/http/server.py#L266

To go into detail about this function, we can start with line 281 where the request is split by a '\r\n':
https://github.com/python/cpython/blob/bf26bdf6ac04878fc720e78422991aaedb9808a1/Lib/http/server.py#L281

Following this, on line 292 onwards to 302, the HTTP protocol version number is determined:
https://github.com/python/cpython/blob/bf26bdf6ac04878fc720e78422991aaedb9808a1/Lib/http/server.py#L302

It then calls a function parse_headers() in python's client.py file on line 224:
https://github.com/python/cpython/blob/1012dc1b4367e05b92d67ea6925a39d50dce31b7/Lib/http/client.py#L224

Which then calls the read_headers() function which (as described by the comments) reads potential header lines into a list on line 206:
https://github.com/python/cpython/blob/1012dc1b4367e05b92d67ea6925a39d50dce31b7/Lib/http/client.py#L206

The headers are parsed and set on line 341 on python's server.py file:
https://github.com/python/cpython/blob/bf26bdf6ac04878fc720e78422991aaedb9808a1/Lib/http/server.py#L341

This function handles parsing HTTP requests in a similar way to what we did in our homework because it uses common string parsing methods (split, rstrip on a '\r\n', etc...) to isolate important information from the requests such as the headers.