Open-Source Report

[Flask]

General Information & Licensing

Code Repository	https://github.com/pallets/flask
License Type	BSD
License Description	Do whatever you want with it as long as the copyright in Flask sticks around, the conditions are not modified, and the disclaimer is present.
License Restrictions	Redistribution and use in source and binary forms of the software as well as documentation, with or without modification, are permitted provided that the following conditions are met: ● 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. ● The names of the contributors may not be used to endorse or promote products derived from this software without specific prior written permission.(https://flask.palletsprojects.com/en/0.12.x/license/#flask-license)

Magic

How does this technology do what it does?

Flask wraps Werkzeug to access WSGI (Web Server Gateway Interface) features such as headers, cookies, and form data. Therefore, the source code for obtaining the values mentioned is not found in Flask, but in Werkzeug.

Werkzeug (WZ), when parsing request headers, first splits into a multithread process, one of which is the “process_request_thread”. This thread ends up calling on socketserver.py to create a BaseRequestHandler, which then uses .handle() as we’ve seen in the homeworks. Within this handle method is a method called “handle_one_request” which ends up handling each request the server receives.

Within this request handler is a call to http.client.parse_headers. This is where the magic happens. http is a Python library that provides all sorts of useful tools like BaseHTTPRequestHandler. So essentially, WZ ends up relying on Python to handle some of the heavy lifting for its server code.

Going into the parse_headers method, we find three key lines:

```
headers = _read_headers(fp)
hstring = b''.join(headers).decode('iso-8859-1')
return email.parser.Parser(_class=_class).parsestr(hstring)
```

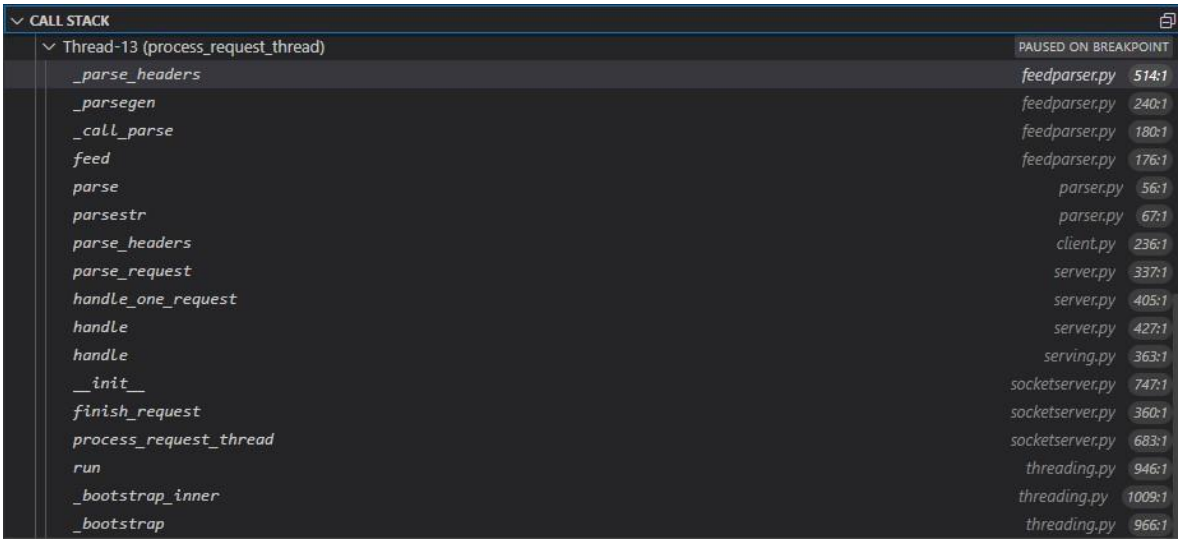
The first uses _read_headers to find lines where headers are expected, put them into a bytestring, and add each line’s bytestring to a list. This gets passed to the second line, which creates one long decoded string of headers to then be passed to the third line.

This third line calls email.parser.Parser.parsestr. parsestr ends up calling a few other helper functions before finally getting to email.feedparser.FeedParser._parse_headers in which we actually find the line: “line.find(‘:’)” which, according to the comments right above it “separates field name from value”. That would be some of Flask’s header parsing in the works.

The parsed headers are put into tuples which then go into WZ’s Headers class, which gets used throughout Flask.

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.

Here is the full call stack. Keep in mind, the bottom is the beginning of one of WZ’s threads, and the deeper parts of the stack are higher up:



The specific line mentioned earlier, which finds colons as part of the header parsing, is located [on line 514 of feedparser.py in the official](#) [cpython repository.](#)

Going back through the attached stack trace, here are the permalinks for each of those calls (in the same order as pictured, including the previously linked 514 of feedparser.py). Note that some of the code has been updated and thus the specific calls made are on different lines. These differences have been noted for the relevant calls.

- _parse_headers - feedparser.py:514
- _parsegen – feedparser.py:240
- _call_parse - feedparser.py:180
- feed - feedparser.py:176
- parse – parser.py:56 (line 53 on GitHub, as permalinked)
- parsestr – parser.py:67 (line 64 on GitHub, as permalinked)
- parse_headers – client.py:236
- parse_request – server.py:337 (line 342 on GitHub, as permalinked)
- handle_one_request – server.py:405 (line 410 on GitHub, as permalinked)
- handle – server.py:427 (line 432 on GitHub, as permalinked)
- handle – serving.py:363 (line 361 on GitHub, as permalinked)
- __init__ - socketserver.py:747 (line 754 on GitHub, as permalinked)
- finish_request - socketserver.py:360
- process_request_thread - socketserver.py:683 (line 690 on GitHub, as permalinked)
- run – threading.py:946 (line 989 on GitHub, as permalinked)
- _bootstrap_inner - threading.py:1009 (line 1052 on GitHub, as permalinked)
- _bootstrap - threading.py:966 (line 1009 on GitHub, as permalinked)