Open-Source Report

Proof of knowing your stuff in CSE312

Guidelines

Provided below is a template you must use to write your reports for your project.

Here are some things to note when working on your report, specifically about the **General Information & Licensing** section for each technology.

- **Code Repository**: Please link the code and not the documentation. If you'd like to refer to the documentation in the **Magic** section, you're more than welcome to, but we need to see the code you're referring to as well.
- **License Type**: Three letter acronym is fine.
- **License Description**: No need for the entire license here, just what separates it from the rest.
- **License Restrictions**: What can you *not* do as a result of using this technology in your project? Some licenses prevent you from using the project for commercial use, for example.

Also, feel free to extend the cell of any section if you feel you need more room.

If there's anything we can clarify, please don't hesitate to reach out! You can reach us using the methods outlined on the course website or see us during our office hours.

Flask

General Information & Licensing

Code Repository	https://github.com/pallets/flask
License Type	BSD 3-Clause
License Description	• Permissive, can use without restriction • Redistributions must meet a few extra conditions • The copyright holder is not liable for anything
License Restrictions	• Cannot use name of copyright holders or names of contributors for endorsing or promoting our product without written permission • Redistribution requires listing a copyright notice and list of conditions

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? Please explain this in detail, starting from after the TCP socket is created
- 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 will likely grow beyond the page

In order to run the server and start receiving headers, our server calls app.run(). This is located here in app.py:
https://github.com/pallets/flask/blob/e1e4e82096efbf25aa3c65b706aec60f1b00dec7/src/flask/app.py#L786

Within app.run(), run_simple() is then called:
https://github.com/pallets/flask/blob/e1e4e82096efbf25aa3c65b706aec60f1b00dec7/src/flask/app.py#L902
Which is imported from werkzeug.serving.run_simple() here:
https://github.com/pallets/flask/blob/e1e4e82096efbf25aa3c65b706aec60f1b00dec7/src/flask/app.py#L899
This is located in another repository called werkzeug:
https://github.com/pallets/werkzeug
Where within that repository werkzeug.serving.run_simple() is defined here:
https://github.com/pallets/werkzeug/blob/main/src/werkzeug/serving.py#L938

run_simple() calls srv.serve_forever() here:
https://github.com/pallets/werkzeug/blob/45c1e774eb77a91b6de2c1923c77d7c7aceaf946/src/werkzeug/serving.py#L1100
Where the srv object that serve_forever() was called on is created with make_server here:
https://github.com/pallets/werkzeug/blob/45c1e774eb77a91b6de2c1923c77d7c7aceaf946/src/werkzeug/serving.py#L1068
And make_server is defined here:
https://github.com/pallets/werkzeug/blob/45c1e774eb77a91b6de2c1923c77d7c7aceaf946/src/werkzeug/serving.py#L884
make_server will return a BaseWSGIServer:
https://github.com/pallets/werkzeug/blob/45c1e774eb77a91b6de2c1923c77d7c7aceaf946/src/werkzeug/serving.py#L924
Defined here:
https://github.com/pallets/werkzeug/blob/45c1e774eb77a91b6de2c1923c77d7c7aceaf946/src/werkzeug/serving.py#L682
Which contains server_forever():
https://github.com/pallets/werkzeug/blob/45c1e774eb77a91b6de2c1923c77d7c7aceaf946/

src/werkzeug/serving.py#L795
Though it calls super().server_forever():
https://github.com/pallets/werkzeug/blob/45c1e774eb77a91b6de2c1923c77d7c7aceaf946/src/werkzeug/serving.py#L797
The BaseWSGIServer inherits from the HTTPServer class:
https://github.com/pallets/werkzeug/blob/45c1e774eb77a91b6de2c1923c77d7c7aceaf946/src/werkzeug/serving.py#L682
Imported here from python:
https://github.com/pallets/werkzeug/blob/45c1e774eb77a91b6de2c1923c77d7c7aceaf946/src/werkzeug/serving.py#L26
Python repository:
https://github.com/python/cpython/
Within the python repository it is defined here:
https://github.com/python/cpython/blob/main/Lib/http/server.py#L130

HTTPServer inherits from socketserver.TCPServer:
https://github.com/python/cpython/blob/main/Lib/socketserver.py#L390
Which inherits from BaseServer:
https://github.com/python/cpython/blob/main/Lib/socketserver.py#L153
Where server_forever() is defined:
https://github.com/python/cpython/blob/main/Lib/socketserver.py#L216
Where a request is handled by calling self.handle_request_noblock():
https://github.com/python/cpython/blob/main/Lib/socketserver.py#L238
Within handle_request_no_block():
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/socketserver.py#L303
The request is processed:
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/socketserver.py#L316
And within process_request:
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/socketserver.py#L341
finish_request is called:
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/socketserver.py#L347
And within finish_request:
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/socketserver.py#L358
self.RequestHandlerClass is instantiated:
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/socketserver.py#L360
Where in the initialization of BaseServer:
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/socketserver.py#L201
self.RequestHandlerClass is set:
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/socketserver.py#L204

Which means going back to BaseWSGIServer:
https://github.com/pallets/werkzeug/blob/45c1e774eb77a91b6de2c1923c77d7c7aceaf946/src/werkzeug/serving.py#L682
Where in the initialization method the second argument as per the HTTPServer and BaseServer initialization is the handler object:
https://github.com/pallets/werkzeug/blob/45c1e774eb77a91b6de2c1923c77d7c7aceaf946/src/werkzeug/serving.py#L734

handler is set to WSGIRequestHandler:
https://github.com/pallets/werkzeug/blob/45c1e774eb77a91b6de2c1923c77d7c7aceaf946/src/werkzeug/serving.py#L704
Which is defined here:
https://github.com/pallets/werkzeug/blob/45c1e774eb77a91b6de2c1923c77d7c7aceaf946/src/werkzeug/serving.py#L149
WSGIRequestHandler inherits from BaseHTTPRequestHandler:
https://github.com/pallets/werkzeug/blob/45c1e774eb77a91b6de2c1923c77d7c7aceaf946/src/werkzeug/serving.py#L25
Which is defined here (in the python repository):
https://github.com/python/cpython/blob/main/Lib/http/server.py#L146
Which inherits from StreamRequestHandler:
https://github.com/python/cpython/blob/main/Lib/socketserver.py#L776
Which inherits from BaseRequestHandler:
https://github.com/python/cpython/blob/main/Lib/socketserver.py#L730
Where the __init__ method is called:
https://github.com/python/cpython/blob/main/Lib/socketserver.py#L748
And within the __init__ method the handle() function is called:
https://github.com/python/cpython/blob/main/Lib/socketserver.py#L754

Going back to WSGIRequestHandler the handle() method is defined:
https://github.com/pallets/werkzeug/blob/45c1e774eb77a91b6de2c1923c77d7c7aceaf946/src/werkzeug/serving.py#L388
Where super().handle() is called, which is BaseHTTPRequestHandler.handle():
https://github.com/python/cpython/blob/main/Lib/http/server.py#L428
Where self.handle_one_request() is called:
https://github.com/python/cpython/blob/main/Lib/http/server.py#L432
Which is defined here:
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/http/server.py#L391
And which contains code that resembles our homework code as it is reading lines.
It calls self.parse_request():
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/http/server.py#L410
Which is defined here:
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/http/server.py#L267
Which contains code that really looks like our homework code as it checks that it starts with HTTP/ among other things to see that it is an actual HTTP request. It then does the rest of the request line, and a lot of error checking, where it can then move on to the headers.

self.headers is set here by calling http.client.parse_headers():
Additionally, after this call it is clear the headers have been parsed because it's using self.headers with the header name as a key to find out what the headers are such as Connection: keep-alive.
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/http/server.py#L342
Which is defined here:
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/http/client.py#L224
Within this method the headers are read:
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/http/client.py#L234
Which is defined here, clearly reading from a file pointer somewhat like our homework

code in order to get each individual line, which would be an individual header:
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/http/client.py#L206
It checks for the end of the headers here, the CRLF CRLF, though since it already read the line it would only be one other CRLF after having already read a CRLF:
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/http/client.py#L220
It then decodes the headers from a bytestring and uses that decoded string to call email.parser.Parser.parsestr():
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/http/client.py#L236
Which is defined here:
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/email/parser.py#L56
It then calls self.parse():
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/email/parser.py#L64
Which is defined here:
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/email/parser.py#L41
parse() calls feed with the data it read from the file pointer:
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/email/parser.py#L53
Which is defined here, and this pushes the data onto a structure that can be accessed later:
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/email/feedparser.py#L171
It then calls self._call_parse():
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/email/feedparser.py#L174
Which is defined here:
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/email/feedparser.py#L176
It then calls self._parse():
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/email/feedparser.py#L178
Which is defined here in the __init__() function of FeedParser:
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/email/feedparser.py#L162
It is defined as self.parsegen().__next__, which is defined here:
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/email/feedparser.py#L216
It then appends the headers that matches its checks which it says are according to the RFC to the headers variable and calls _parse_headers() on that variable:
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/email/feedparser.py#L238
Which is defined here:
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/email/feedparser.py#L469
After doing some checks, it splits the line on a colon like in our homework code, first finding the colon to split on an index, also like our homework):
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/email/feedparser.py#L512
It then sets lastheader to the header name, such as “Connection”, and sets lastvalue to the entire line as in “Connection: keep-alive”.

https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/email/feedparser.py#L523-L524
It does the for loop again, this time with lastheader and lastvalue set to something.
It then calls self.policy.header_source_parse(lastvalue):
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/email/feedparser.py#L485-L487
Which is defined here (Compat32 is the default self.policy):
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/email/_policybase.py#L293-L303
Which contains code which is the same as our homework code, splitting on colon, stripping the potential extra space after the colon, and stripping the CRLF on the end of the string that delimits the headers. It then returns them as a tuple.
This tuple is then appended as a tuple inside of the Message class (_cur is set as an instance of class Message):
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/email/message.py#L510
Calling the get method of the Message class allows you to access it in the same way it was accessing it in parse_request as in here:
https://github.com/python/cpython/blob/7b95d23591f605fc05d4820f83fef8fbf1552729/Lib/http/server.py#L358
It returns all the way back and starts running the above code, which, since it accesses the headers from self.headers, means that it has parsed the headers of the BaseHTTPRequestHandler class which was inherited by the WSGIRequestHandler class

This next part is somewhat confusing and I'm pretty sure unnecessary at this point:

In order to access headers sent in a request, you need to use flask.request.headers. The global flask.request object is defined here defined here:
https://github.com/pallets/flask/blob/main/src/flask/globals.py#L65

app.py imports this here:
https://github.com/pallets/flask/blob/main/src/flask/app.py#L43

The request object is the Request class as in this line in app.py:
https://github.com/pallets/flask/blob/main/src/flask/app.py#L204
Which links to this definition of the Request class:
https://github.com/pallets/flask/blob/e1e4e82096efbf25aa3c65b706aec60f1b00dec7/src/flask/wrappers.py#L15

That definition uses the werkzeug Request class, imported from werkzeug.wrappers.Request seen here:
https://github.com/pallets/flask/blob/e1e4e82096efbf25aa3c65b706aec60f1b00dec7/src/flask/wrappers.py#L4

That brings us to werkzeug:
https://github.com/pallets/werkzeug

In werkzeug.wrappers the Request class is defined again:
https://github.com/pallets/werkzeug/blob/main/src/werkzeug/wrappers/request.py#L30
This time with headers:
https://github.com/pallets/werkzeug/blob/main/src/werkzeug/wrappers/request.py#L122

headers however is an instance of the EnvironHeaders class imported here:
https://github.com/pallets/werkzeug/blob/45c1e774eb77a91b6de2c1923c77d7c7aceaf946/src/werkzeug/wrappers/request.py#L9
And another import here before getting to the definition:
https://github.com/pallets/werkzeug/blob/45c1e774eb77a91b6de2c1923c77d7c7aceaf946/src/werkzeug/datastructures/__init__.py#L12
And finally defined here:
https://github.com/pallets/werkzeug/blob/45c1e774eb77a91b6de2c1923c77d7c7aceaf946/src/werkzeug/datastructures/headers.py#L518

This EnvironHeaders class acts like a dictionary for headers to be accessed. The __getitem__ function is defined here:
https://github.com/pallets/werkzeug/blob/45c1e774eb77a91b6de2c1923c77d7c7aceaf946/src/werkzeug/datastructures/headers.py#L536
The value returned is from the self.environ environment…

The __getattr__ method in BaseWSGIServer calls run_wsgi:
https://github.com/pallets/werkzeug/blob/45c1e774eb77a91b6de2c1923c77d7c7aceaf946/src/werkzeug/serving.py#L410
Which calls make_environ()
https://github.com/pallets/werkzeug/blob/45c1e774eb77a91b6de2c1923c77d7c7aceaf946/src/werkzeug/serving.py#L245
It uses the self.header in make_environ() to fill out the key value pairs accessible in environ
https://github.com/pallets/werkzeug/blob/45c1e774eb77a91b6de2c1923c77d7c7aceaf946/src/werkzeug/serving.py#L204-L218