Open-Source Report: Parsing HTTP Headers

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	Flask Repository
License Type	BSD 3-Clause "New" or "Revised" License via source
License Description	• All code is free for commercial and private use • Code is free to be modified • Code is free to distribute • No citations or credits are required when publishing code that uses code from this repo

License Restrictions	• Flask is not liable if we use their code and it breaks anything • Flask does not provide a warranty for their code

Magic

General Explanation

After a TCP socket is created, app.py is responsible for processing incoming requests. When a client sends a request to a Flask app, the request is first received by the WSGI server, which is responsible for passing the request on to the Flask application. The Flask application then processes the request and sends back a response to the WSGI server, which then sends the response back to the client.

When Flask receives an HTTP request from a client, it accesses the Request object declared within the wrappers module of the Werkzeug library through the request variable. Upon receiving an HTTP request from the client, Flask internally calls a function with two parameters, environ and start_response, to handle the request received from the WSGI server. The first parameter of this function, environ, is a dictionary containing information about the request. Flask converts this dictionary into a Werkzeug Request object and assigns it to the request variable. Therefore, the Request object can be accessed within the Flask application through the request variable.

Step by step

In our app.py file, you can find we declared Flask into a variable “app”. Using **Ctrl +** Click(**Command +** Click), you can access the Flask class file. Through the line, you can access the Request class. The Request object contains parsed information from the HTTP request. In the Request class, you can find load form data method. This method parses form data or file upload data based on the value of the Content-Type header and converts it into a dictionary using the MultiDict class of the Werkzeug library.

Where “MultiDict” is defined in werkzeug/structures.py
Where “Request” is defined in werkzeug/wrappers/request.py
Where “Request” is defined in Flask/wrappers.py
Where “_load_form_data” is defined in Request class in Flask/wrappers.py