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 / Python

General Information & Licensing

Code Repository	https://github.com/miguelgrinberg/flask-socketio : Server Side https://github.com/socketio/socket.io : Client Side https://github.com/sinank/gevent-websocket/blob/5020669b0439fd49f054830c51b1aa1602b7d086/geventwebsocket/websocket.py#L44: Raw Data Parsing
License Type	BSD 3-Clause "New" or "Revised" License
License Description	• A permissive license similar to the BSD 2-Clause License, but with a 3rd clause that prohibits others from using the name of the copyright holder or its contributors to promote derived products without written consent

License Restrictions	• Liability • Warranty

Dispel the magic of this technology. Replace this text with some that answers the following questions for the above tech:

- Explain what this technology does in your project. What problems does it solve for you?
- 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

Explain what this technology does in your project. What problems does it solve for you? This technology setsup a websocket connection and allows us to use a mainstream form of communication between websockets. Flask-Socketio and gevent-websocket work in tandem to establish the websocket connection and parse frame data to the server, and to the client from the server.

Creating the server through flask is done by simply calling the Flask function, this creates a server, and then to upgrade it we call the SocketIO function. This upgrades the whole server to be able to handle websocket connections.

In line 54 of Flask-SocketIO our flask app gets transformed into a socketio server that can now handle websocket connections.

https://github.com/miguelgrinberg/Flask-SocketIO/blob/91b5ddc31bebeb6241d281252c711b160550ce01/src/flask_socketio/__init__.py#L54

Ctrl-f “websock” and on line 690 websocket is set to true
https://github.com/miguelgrinberg/Flask-SocketIO/blob/91b5ddc31bebeb6241d281252c711b160550ce01/src/flask_socketio/__init__.py#L690

And on line 701 since websocket was set to true, we set the wsgi_server
https://github.com/miguelgrinberg/Flask-SocketIO/blob/91b5ddc31bebeb6241d281252c711b160550ce01/src/flask_socketio/__init__.py#L701

Below that, theres a conditional that checks if the reloader was used, if true then we run the server forever:
https://github.com/miguelgrinberg/Flask-SocketIO/blob/91b5ddc31bebeb6241d281252c711b160550ce01/src/flask_socketio/__init__.py#L715

But if we don't use the reloader, we still serve the wsgi server forever:
https://github.com/miguelgrinberg/Flask-SocketIO/blob/91b5ddc31bebeb6241d281252c711b160550ce01/src/flask_socketio/__init__.py#L719

Web Frame Parsing:

Flask-Socketio does not parse the frame data, but instead uses the gevent-websocket library. The first thing gevent does is use its read_frame function and within this function it calls to the raw stream and reads from it to separate the headers from the body.
https://github.com/sinank/gevent-websocket/blob/5020669b0439fd49f054830c51b1aa1602b7d086/geventwebsocket/websocket.py#L200

Once it has started reading from the raw frame it will begin to completely receive the frame by calling to read whatever amount of bytes the length was in the headers. This function returns the headers, and payload.
https://github.com/sinank/gevent-websocket/blob/5020669b0439fd49f054830c51b1aa1602b7d086/geventwebsocket/websocket.py#L209

Using the returned headers, and payload gevent-websocket library then calls read_message(), within this function it will use the data returned from read_frame(), the code from this function is the most familiar because it goes into an infinite loop collecting all data within the frame.
https://github.com/sinank/gevent-websocket/blob/5020669b0439fd49f054830c51b1aa1602b7d086/geventwebsocket/websocket.py#L244

The function checks if a new frame is being collected.
https://github.com/sinank/gevent-websocket/blob/5020669b0439fd49f054830c51b1aa1602b7d086/geventwebsocket/websocket.py#L247

Sending a Frame:

The function is found on line 315 of the gevent-websocket/geventwebsocket/websocket.py library. The description of this function is " Send a frame over the websocket with message as its payload"
https://github.com/sinank/gevent-websocket/blob/5020669b0439fd49f054830c51b1aa1602b7d086/geventwebsocket/websocket.py#L315

Line 323 is where we check if the payload data/message we're sending will be of type bytes or string
https://github.com/sinank/gevent-websocket/blob/5020669b0439fd49f054830c51b1aa1602b7d086/geventwebsocket/websocket.py#L323

Line 328 is where the header is created for the send frame
https://github.com/sinank/gevent-websocket/blob/5020669b0439fd49f054830c51b1aa1602b7d086/geventwebsocket/websocket.py#L328

Line 331 is where the frame is actually sent.
https://github.com/sinank/gevent-websocket/blob/5020669b0439fd49f054830c51b1aa1602b7d086/geventwebsocket/websocket.py#L331