Open-Source Report: Websockets

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-sock

General Information & Licensing

Code Repository	Flask-sock Repository
License Type	MIT 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-sock is not liable if we use their code and it breaks anything • Flask-sock does not provide a warranty for their code

Magic

Flask-sock has a class called Sock in __init__.py which takes the Flask app as a parameter. This class calls the __init__ function which initializes the websocket to be used. To create a websocket connection for a desired route, you would use the function “route” in the Sock class. Route calls a nested function called ‘decorator’ which then calls another nested function called ‘websocket_route’. ‘websocket_route’ will then create an object from the Server class in ‘ws.py’ from the simple_websocket library. The Server class calls its own __init__ function to set the server type and then completes the websocket handshake with the function ‘handshake’. This handshake function is used to format the HTTP response for the 101 switching protocols. To send and receive data over the websocket connection, the Base class from ‘ws.py’ uses the ‘send’ and ‘receive’ functions. The ‘send’ function uses the ‘Message’ and ‘TextMessage’ classes from the ‘events.py’ file of the wsproto library. The ‘Message’ class has attributes ‘data’ which is the message data of the frame as a string and ‘message_finished’ is a bool that represents whether more frames are expected for that frame. ‘TextMessage’ has just a single attribute ‘data’ which has the same definition as it does in ‘Message’. The ‘receive’ function checks that while the connection is still valid that the input buffer is available to receive bytes so long as the input buffer isn’t already containing an unsent frame. When the connection has been terminated, the ‘receive’ function sends a connection closed message.

Where flask-sock defines “Sock” class
Where “Sock” defines “__init__” function
Where “Sock” defines “route” function
Where “Server” is defined in “ws.py”
Where “Server” defines “__init__”
Where “Server” defines “handshake”
Where “Base” is defined in “ws.py”
Where “send” is defined in “Base” class
Where “receive” is defined in “Base” class
Where “Message” class is defined in “events.py”
Where “TextMessage” class is defined in “events.py”
Where “CloseConnection” class is defined in “ws.py”