Open-Source Report

Flask-SocketIO

General Information & Licensing

Code Repository	https://github.com/miguelgrinberg/Flask-SocketIO
License Type	MIT
License Description	• The license gives permission to use, copy, modify, merge, publish, distribute, and sublicense the software so long as the copyright notice as well as the permission notice are included alongside it.
License Restrictions	• The license is very open-source friendly, the only restriction is that the author can’t be held liable for how the technology is used.

Magic

What does this technology (library/framework/service) accomplish for you?

SocketIO gives our project access to low latency communication between the clients and the server. The benefit of using SocketIO is that using regular WebSockets may require implementing features such as reconnection and broadcasting for example which are already included in SocketIO.

How does this technology accomplish what it does?

The first step to using socketio is to create a GUID for the handshake to go through. In our project we store a secret key in a private file, and then set the GUID equal to our secret key within the flask instance.

After we have the guid key we can create the flask-socketio server.

Once SocketIO is called it will enter _init_.py creating the instance variables for the app with the given flask standard initialization we created.

Since we called SocketIO with a standard initialization of flask it will then call init_app. _init_app will call socketio.Server. This call will trace through the socketio and engineio libraries in order to deploy eventlet and gevent.

- eventlet: will be called upon for its websocket transports.
- gevent: provides additional websocket support and long polling(however we will disable long polling due to project requirements)

It will also lead to the handshake done in engineio.

Now that websockets are set up we can start the server by calling socketio.run.

Parsing Frames:
We can use socketio.on() to listen for web socket events. This will go to the on() function and then decide on how to handle the event. Once the event is handled _handle_event_internal will send the frame.

Broadcasting:
Sending data to other clients can be done by the use of emit(). This calls upon the emit function in flask_socketio. Then the data that needs to be sent will be handled by _handle_event just like on() and then the frames will be sent to everyone.