Java Chat Application Using Web Sockets

By Daniel Schmidt

**Introduction:**

This Application uses Web Sockets to establish a connection with the local host and to keep that socket open for other users under the same local host to use that chat application. By opening a web socket, multiple users are able to connect to the application at the same time and are able to share information, which in this case are chat messages through that single web socket connection through the local host. Formally, a web socket is an application protocol that provides full-duplex communications between two peers over the TCP protocol.

**Functionality:**

How the application functions itself is that the user sends a message through an input text box calling a javascript function called sendMessage() located in the default.html page. This function sends that message into the websocket object. That message is then processed in the Java function that is located under the @OnMessage element and checks if the message is null and if so it assigns that first text to a username else it sends the Json Data to the websocket to be parsed. After the JSON data has been sent, the message is added to the messagesTextArea value of the text area assuming the value is not null. Similarly the username information is added to the usersTextArea value parameter.

**Features:**

Some of the features that have been added to the chat application through the html portion of the chat application are an options menu, which currently allows the user to change the color of the text of the chat application to either red, blue, green, or back to the default color black, and also has a feature which allows the user to enlarge the text or lower the size of the text in order to make it more readable to the user. These two features are primarily implemented through the html page using javascript and don’t use websockets or the Java portion of the application. These functions are implemented through the use of radio buttons and an options menu where the Text\_Color and Font Size option can be clicked on. How it works is that the user selects either text\_Color or Font Size in the scroll down menu to the right of the Send button in the Chat Application. When the user clicks on one of these options, a list of radio buttons is created in the html through a div tag and an option can be selected. Once you click on an option that triggers the onclick function which is attatched to each radio button which changes the text depending on the option that is chosen. For example, if the color of the text is currently black and the user chooses green, the text color of both the Users text area and the message Text area should be changed to green. This is done by changing the style.color element of the messagesTextArea and usersTextArea to the correct color depending on which one is selected. This is similarly implemented in changing the size of the text by selecting a radio button. Instead of changing style.color element you change the style.fontSize element and call a function that checks which size has been selected and makes adjustments to the text area accordingly.

**Tools:**

The main tools that were used in this project are Eclipse using the Java EE perspective. This project was declared as a dynamic web project, which was necessary in order to allow the application to interact with the internet. Another tool that was necessary to implement this project was the GlassFish 4.0 server which is a popular Application server that is used with the Java EE perspective. All of these tools are necessary to run this project as this is one of the only ways to get the Java to interact with the html and javascript code that I could think of. Another tool that was used in this project was Mozilla Firefox web browser as it was the main browser that was used in order to test this project.

**Issues:**

Some of the main issues with running this application were implementing the JavaScript and html features of this project. The two main browsers that I tested the project on were Mozilla Firefox and Google chrome. While Mozilla was the more reliable browser and would give relatively consistent and correct results Google Chrome was having issues running the application in some instances and in others were having no problems at all running the application. Overall, the results of Chrome were relatively inconsistent while the results of Mozilla were not. Also, the GlassFish server had some issues loading up initially, but after the initial setup was relatively easy to start and use.

**Conclusion:**

Overall, this project was successful in the sense that it was able to produce a chat application that can be connected to by multiple users using websockets. This project was also a great learning experience for me, in the sense that I had no previous experience of web sockets and minor experience with html and javascript which are all areas that I want to expand my knowledge in. There is obviously a lot of work that can be added to this project in order to improve on it both through the html interface, such as adding more user options for the chat application and also some tweaks and improvements to the websocket side of the project as well.