In-Class Exercise: Chat App Web Sockets

Instructor: Mackale Joyner

Course Website: https://www.clear.rice.edu/comp504

Goals for this exercise

- Understanding how to use web sockets with your chat app

Important tips and links

NOTE: The instructions below are written for Mac OS and Linux computers, but should be easily adaptable to Windows with minor changes. For example, you may need to use \ instead of / in some commands.

1 Checkout Github Classroom Exercise Repo

First, you'll need to accept the exercise 7 repository by visiting GitHub ex7. Check out your github classroom ex 7 starter code from the remote github repo in IntelliJ from either the "Check out from Version Control" option on the welcome screen or the "Checkout from Version Control" option accessible under the top menu's VCS tab. Use the plus button in the pop-up window to add your GitHub ex7 repo. You should see a ex7 directory created in your working directory with the source code for this exercise.

2 Exercise 7

for this exercise, you will create a simply chat app that uses web sockets as the communication protocol. All messages sent by a client will be broadcast to all the other clients through the server. These messages should appear in the client's view. The localhost chat app web socket location is `ws://localhost:4567/chatapp`. However, for secure protocols (https), the chap app web socket location is `wss://appname.herokuapp.com/chatapp`.

2.1 Server Web Socket

The server-side web socket is implemented using two classes in the `edu.rice.comp504.controller` package. Each class has a `TODO` comment to explain what you'll need to implement. The main class is the `ChatAppController`. This class initializes the web socket defined in `WebSocketController`. The `ChatAppController` includes a `broadcastMessage` method that iterates over all the user client sessions and sends each client a message. This message was initially passed to the server from one of those clients.

The `broadcastMessage` method currently sends an empty JSON object to each client. As the `TODO` indicates, you'll need to add a key-value property to the JSON object. The key will be a string named "userMessage". The value will be a j2html paragraph that has the following format: `[username] says:[message]` where [username] is the unique id name associated with the user session and [message] is the message a client sent to the server.

The `WebSocketController` class represents the server-side web socket as indicated with the `@WebSocket` annotation. This class contains the method `onMessage`. The `onMessage` method is triggered every time a client sends the web socket a message. As the `TODO` suggests, you'll need to broadcast the message to all the clients.

2.2 Client Web Socket

A client's view (view.js) should display every message sent from the server-side web socket. Currently, `updateChatRoom` is empty. The `updateChatRoom` method needs to display the user message embedded in the stringified JSON object server message. So, you'll need to first convert the server message data back to JSON. Now you can pull out the user message from the JSON object and append the user message to the chat area div tag. You should now be able to open multiple browser tabs, navigate to http://localhost:4567, and send messages from one user to another user.

3 Demonstrating and submitting the exercise

You'll need to host your app on heroku. The app name should be [netid]-ex7-chat. Place the app name and your name in the README file.

Please don't forget to commit and push your work to your github classroom repository. To perform a git commit, select VCS on the menu and click on "Commit...". Add a commit message and click on the `Commit` button. To push the local changes, select VCS on the menu, highlight the Git option and select "Push...".