CS2003 W11 Practical

140012021

27/11/15

Overview

The overview of the practical was to build a simple peer-to-peer messaging application that would have a web-based user interface (UI). The UI would communicate with a message agent (MA). The MA would accept the messages via the UI and send them to other people using their respective MA instances. You also needed be able to get messages from other people using the same process. Each users MA instance has to be able to communicate with other MA instances in a peer-to-peer manner to allow transfer of messages between users.

In addition I added some extensions to the practical. Which are shown below.

Tasks completed: 1-7

Extensions completed: I and II.

Design

I decided from the start that it would be easier to make practical in small steps. I first created the the html for the practical, so as soon as I got anything working I could easily use it to see if the messenger was working. Inside the html I also put the JavaScript for the client. I had two main functions inside the client. One that handled the event of getting a message and the other the event of sending an event. When getting a message I would check what type of message it was first, whether it was an agent_message or a delivery. If it was an agent_message then my practical would create a message to print out to the web page using information from the message. The other type of message you can receive is a delivery message from the user saying that your message was successfully delivered. I decided to make the messages I send green if I receive a delivery message and red if I don't. The other function was when I send messages. I would use the information from my own message to display the correct content on the screen, such as the date and the fact that I sent it.

Next I wrote the server for the task. I wrote a function that would read in all the different users that could be used and their port numbers and store them as objects in an array. I also made a function that could get the port number for a corresponding user, which would be used later on when setting up a tcp connection with the other user being messaged. I also created the http server required. I would send a message to a another user by first making the package of the correct format then, I would create the tcp connection with the user using their address and port number. Before the messages were sent they needed to be stringified and before they could be handled they needed to be parsed. I also wrote a bit that would

send messages to the users. If type of the message was an agent_message, I would change the type of that message and other fields of it, to then send it back as a delivery message. Finally I added in try and catch statements to the program in order to make it more robust.

Testing

For testing I tried messaging different users with their servers both on and off. I also tried holding chats with more than one person at a time. Screen shots from testing can be found at the end of the document.

Practical Information and Extension II

URL of solution: http://caf8.host.cs.st-andrews.ac.uk:18250/client2.html

Submission contains READ_ME file which contains execution information.

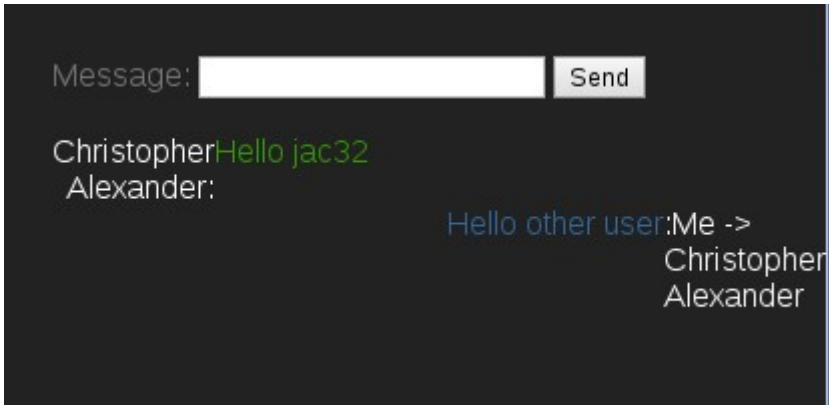
The security issue that exists with the current practical is that if my server is running, whoever loads up the above URL can send messages as if they were me. This could prove a serious issue, it may also be vulnerable to code injection. A way to stop people from randomly using my practical would be to introduce a password before you can send messages.

Evaluation

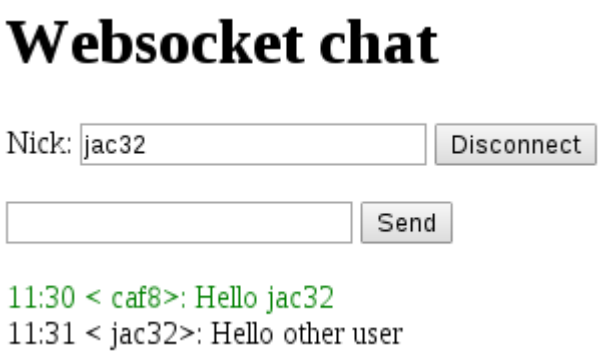
I feel my program was successful in implementing a peer-to-peer messaging application that had a web based UI. It could successfully send and receive messages to/from other MA and display the messages on the screen saying who they were from. I could also highlight the messages I send in green if they were successfully delivered. I also completed two extensions for the practical.

Conclusion

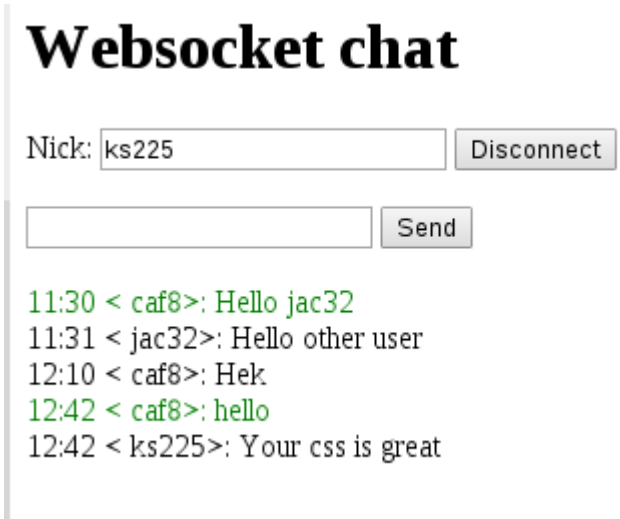
Overall I feel I have achieved the task was set. The practical definitely allowed me to gain a better understanding of how all the different type of sockets work and a better understanding of websockets in particular. If I had more time I would of liked to add many features to my practical and improving the styling of the html page. However due to unfortunate circumstances I wasn't able to put as much effort into the practical as I would of liked.



Screenshot 1: Other user receiving my messages successfully



Screenshot 2: My end of the same chat as above



Screenshot 3: Sending and receiving messages from multiple users.