BÀI THỰC HÀNH
MÔN HỌC: HỆ PHÂN TÁN
CHƯƠNG 3: Tiến trình và Luồng trong HPT

1. Xây dựng một Chat room sử dụng socket.io

1.1. Nội dung thực hành

Websocket là một giao thức trao đổi thông tin cung ứng cơ chế truyền tin 2 chiều song song trên cùng một kết nối TCP. Giao thức Websocket đã được chuẩn hóa bởi IETF như RFC 6455 năm 2011, và bộ WebSocket API trong Web IDL đã được chuẩn hóa bởi W3C.

Socket.IO là một thư viện JavaScript cho các ứng dụng web thời gian thực. Nó cho phép trao đổi thông tin 2 chiều và thời gian thực giữa các web clients và servers. Nó có 2 phần: thư viện cho phía client chạy trên trình duyệt, và thư viện cho server cho Node.js. Cả 2 thành phần đó đều có chung bộ API. Như Node.js, nó có tính hướng sự kiện. Socket.IO sử dụng giao thức WebSocket, tuy nhiên nó có nhiều chức năng hơn như có khả năng quảng bá tới nhiều sockets, lưu trữ dữ liệu liên quan với mỗi client, và thực hiện trao đổi vào ra bất đồng bộ. Ở bài thực hành này, các bạn sẽ học cách sử dụng Socket.IO (ngôn ngữ Node.js) để phát triển một chat room giao diện web dựa trên giao thức WebSocket. Thực tế đây là một hệ thống thời gian thực với các thao tác gửi và nhận dữ liệu. Bạn không thể thực hiện nó theo các cách truyền thống bởi vì với mô hình web truyền thống, client sẽ yêu cầu server và sau đó server mới trả lời client theo một cách tuần tự. Điều này là không khả thi với một ứng dụng chat khi mà cả 2 bên cần trao đổi với nhau cùng lúc. Với WebSocket, cả client và server có thể gửi dữ liệu cho nhau cùng lúc. WebSocket là một kiểu trao đổi thông tin dạng đường ống mở 2 chiều. Bạn cần phải sử dụng thư viện socket.io để làm điều đó.

1.2. Yêu cầu

1.2.1. Lý thuyết

- WebSocket và thư viện socket.io
- node.js

1.2.2. Phần cứng

- Laptop/PC dùng Windows

1.2.3. Phần mềm

- node.js

1.3. CÁC BƯỚC THỰC HÀNH

Đầu tiên cần phải tải xuống và cài đặt node.js: https://nodejs.org/en/download/

Node.js là một công nghệ back-end sử dụng Javascript được chạy bởi server như PHP, Ruby, hay là Python. Javascript sử dụng các sự kiện. Node.js giữ nguyên những đặc điểm đó vì vậy rất dễ dàng để có thể chạy các đoạn mã bất đồng bộ. Node.js có có bộ quản lý gói riêng là: *npm*. Nó khiến cho việc cài đặt, cập nhật và xóa các gói trở nên dễ dàng. Trong bài thực hành này, các bạn sẽ sử dụng express.js. Nó là một framework dạng web dựa trên node.js.

Bây giờ hãy tạo một thư mục tên là ChatRoomApp, vào đó và khởi tạo môi trường phát triển bằng các lệnh sau:

```
>mkdir ChatRoomApp
>cd ChatRoomApp
>npm init
```

Gặp câu hỏi nào thì hãy ấn enter cho đến hết.

Câu hỏi 1: Tệp nào vừa xuất hiện trong thư mục ChatRoomApp? Nó được sử dụng để làm gì?

Bạn phải cài một số gói cần thiết để phát triển chat room:

```
>npm install --save express
>npm install --save nodemon
>npm install --save ejs
>npm install --save socket.io
```

Giải thích về 4 gói vừa cài:

- *express*: đây là một micro framework ứng dụng dạng web cho node.js
- *nodemon*: gói này có tác dụng phát hiện ra các thay đổi và tự động restart lại server. Sử dụng nó sẽ rất thuận tiện so với các câu lệnh truyền thống.
- *ejs*: ejs là một template engine dùng để đơn giản hóa việc sinh mã HTML.
- *socket.io*: thư viện cho WebSockets

Trong file package.json, bạn hãy thay đổi mã như sau:

```
"scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "nodemon app"
  }
```

Bây giờ bạn phải phát triển 2 phần: client và server.

Tạo 1 file app.js mới (nó dùng để chạy server và tất cả các gói) và đưa vào đoạn mã sau:

```
const express = require('express')
const app = express()


//set the template engine ejs
app.set('view engine', 'ejs')

//middlewares
app.use(express.static('public'))


//routes
app.get('/', (req, res) => {
     res.send('Hello world')
})

//Listen on port 3000
server = app.listen(3000)
```

Bây giờ hãy chạy ứng dụng của bạn với lệnh:

```
>npm run start
```

Câu hỏi 2: Mở trình duyệt và gõ vào đó địa chỉ *http://localhost:3000,* bạn sẽ nhận được thông điệp gì?

Thêm các đoạn mã sau vào file *app.js*:

```
//socket.io instantiation
const io = require("socket.io")(server)


//listen on every connection
io.on('connection', (socket) => {
         console.log('New user connected')
})
```

Ở đây, đối tượng *io* sẽ cho phép chúng ta sử dụng thư viện *socket.io*. Đối tượng *io* bây giờ sẽ nghe trên mỗi kết nối đến với ứng dụng của bạn. Mỗi lần có một người dùng mới kết nối đến, nó sẽ hiển thị thông điệp "New user connected" lên console.

Câu hỏi 3: Bạn hãy thử reload (Ctrl-R) lại trình duyệt. Bạn có nhìn thấy gì mới xuất hiện trên cửa sổ không? Nếu không có gì xuất hiện hết thì là vì sao?

Bây giờ thì socket.io mới được cài đặt trên phần server. Kế tiếp, chúng ta sẽ làm điều tương tự trên phần client.

Bạn chỉ cần thay đổi một dòng trong file *app.js*. Cụ thể, bạn không muốn hiển thị thông điệp "Hello world" nữa, nhưng một cửa sổ thật với các ô chat box, ô nhập

vào username, thông điệp, và nút Send. Để làm được điều đó bạn phải có một file html (trong trường hợp này là file *ejs*) khi truy cập vào thư mục root "/".

Thay đổi thành phần routes trong file *app.js* như sau:

```
//routes
app.get('/', (req, res) => {
        res.render('index')
})
```

Trong thư mục ChatRoomApp của bạn, hãy tạo 2 thư mục con mới tên là *public* và *views*.

```
>mkdir public
>mkdir views
```

Trong thư mục views, hãy tạo tệp tên là index.ejs
Trong thư mục views, hãy tải file index.ejs từ đường link:
https://github.com/anhth318/ChatRoomApp/blob/master/views/index.ejs,
hoặc tự tạo file và viết nội dung sau:

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" const="text/html;charset=UTF-8"
/>
    <link href="http://fonts.googleapis.com/css?family=Comfortaa"
rel="stylesheet" type="text/css">
    <link rel="stylesheet" type="text/css" href="style.css" >
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.0.4/socket.io
.js"></script>
    <title>Distributed Systems Course</title>
  </head>

  <body>
    <header>
      <h1>Chat room for Distributed Systems class</h1>
    </header>

    <section>
      <div id="change_username">
                    <input id="username" type="text" />
                    <button id="send_username" type="button">Change
username</button>
      </div>
    </section>

    <section id="chatroom">
      <section id="feedback"></section>
    </section>



    <section id="input_zone">
      <input id="message" class="vertical-align" type="text" />
```

```
      <button id="send_message" class="vertical-align"
type="button">Send</button>
    </section>

    <script src="http://code.jquery.com/jquery-
latest.min.js"></script>
    <script src="chat.js"></script>
  </body>
</html>
```

Trong thư mục *public*, hãy tạo file *chat.js* với nội dung sau:

```
$(function(){
     //make connection
     var socket = io.connect('http://localhost:3000')
});
```

(chú ý là nếu bạn làm việc với 2 máy tính thì hãy thay localhost bằng địa chỉ IP của server).

Tải file *style.css* và đặt nó trong thư mục *public*:
https://github.com/anhth318/ChatRoomApp/blob/master/public/style.css

Câu hỏi 4: Refresh trang *localhost:3000*, bạn nhìn thấy thông điệp nào?

Ở phía client, bây giờ bạn muốn làm chủ một số hành động khác như: gửi tin nhắn, nghe tin nhắn mới, và gửi đi username. Thay đổi mã nguồn của *chat.js* như sau:

```
$(function(){
   //make connection
   var socket = io.connect('http://localhost:3000')

   //buttons and inputs
   var message = $("#message")
   var username = $("#username")
   var send_message = $("#send_message")
   var send_username = $("#send_username")
   var chatroom = $("#chatroom")

   //Emit message
   send_message.click(function(){
       socket.emit('new_message', {message : message.val()})
   })

   //Listen on new_message
   socket.on("new_message", (data) => {
       message.val('');
       chatroom.append("<p class='message'>" + data.username + ": " +
data.message + "</p>")
   })

   //Emit a username
   send_username.click(function(){
          socket.emit('change_username', {username : username.val()})
   })
});
```

Trong file app.js, hãy thay đổi mã nguồn của phần *listen on every connection* như sau:

```
//listen on every connection
io.on('connection', (socket) => {
    console.log('New user connected')

    //default username
    socket.username = "Anonymous"

    //listen on change_username
    socket.on('change_username', (data) => {
          socket.username = data.username
    })

    //listen on new_message
    socket.on('new_message', (data) => {
        //broadcast the new message
        io.sockets.emit('new_message', {message : data.message,
username : socket.username});
    })
})
```

Với sự kiện *new_message,* bạn có thể thấy chúng ta gọi đặc tính socket của đối tượng *io*. Nó biểu diễn tất cả các socket đã kết nối. Vậy dòng này sẽ gửi một tin nhắn đến tất cả các sockets. Chúng ta muốn hiển thị tin nhắn được gửi bởi một user lên màn hình của tất cả các users khác (và của chính user gửi

Bây giờ hãy test ứng dụng của bạn. Mở đường dẫn http://localhost:3000/ trong một vài tab của trình duyệt của bạn, thử chat trong room.

Nếu bạn muốn hiển thị dòng "User A is typing ..." khi đang có người dùng khác gõ thì bạn có thể làm như sau: Mở file *app.js* , thêm vào đoạn mã sau dưới hàm *io.on()*

```
//listen on typing
socket.on('typing', (data) => {
         socket.broadcast.emit('typing', {username :
socket.username})
})
```

Mở file *chat.js*, thêm vào mã như sau:

Trong phần "//buttons and inputs", thêm vào dòng sau:

```
var feedback = $("#feedback")
```

Trong phần "//Listen on new_message", sửa lại code như sau:

```
//Listen on new_message
socket.on("new_message", (data) => {
    feedback.html('');
    message.val('');
    chatroom.append("<p class='message'>" + data.username + ": " +
data.message + "</p>")
})
```

Thêm vào 2 phần như sau:

```
//Emit typing
message.bind("keypress", () => {
    socket.emit('typing')
})

//Listen on typing
    socket.on('typing', (data) => {
    feedback.html("<p><i>" + data.username + " is typing a
message..." + "</i></p>")
})
```

Bây giờ bạn có thể thử refresh lại cửa sổ trình duyệt của bạn để thử nghiệm chat room.

Câu hỏi 5: Bây giờ bạn hãy thử gõ gì đó lên một tab. Cùng lúc đó, nhìn sang tab khác của người dùng khác, bạn thấy gì?