MỤC TIÊU:

Kết thúc bài thực hành này bạn có khả năng

- Ứng dụng ServerEndpoint API để tạo Chat Server
- Ứng dụng WebSocket API để tạo Chat Client

PHẦN I

Xây dựng ứng chat cộng đồng có giao diện như sau:

Yêu cầu:

- ✓ Thông báo đến tất cả mọi người khi có người vào/ra
- ✓ Gửi tin nhắn chat cho tất cả mọi người

BÀI 1 (2 ĐIỂM)

Sử dụng ServerEndpoint API để xây dựng chat server có end-point url là "/text/chat" phục vụ cho các WebSocket client với yêu cầu:

- ✓ Gửi thông điệp "Someone joined the chat!" đến tất cả mọi người ngay sau khi chấp nhận kết nối từ client
- ✓ Gửi thông điệp "Someone left the chat!" đến tất cả mọi người ngay trước khi đóng kết nối từ client
- ✓ Khi nhận được tin nhắn chat thì chuyển tiếp tin nhắn đó đến tất cả mọi người

Hướng dẫn:

✓ Tạo lớp TextChatServerEndpoint có tổ chức mã như sau

```
@ServerEndpoint("/text/chat")
public class TextChatServerEndpoint {
        // Duy trì danh sách session của các client đang kết nối
        private static Map<String, Session> sessions = new HashMap<>();

        // Gửi @message đến tất cả client đang kết nối
        private void broadcast(String message) {
        }
        @OnOpen
        public void onOpen(Session session) {
        }
        @OnMessage
        public void onMessage(String message, Session session) {
        }
        @OnClose
        public void onClose(Session session) {
        }
        @OnError
        public void onError(Session session, Throwable throwable) {
        }
}
```

Viết mã cho phương thức broadcast(String)

```
sessions.forEach((id, session) -> {
        try {
                session.getBasicRemote().sendText(message);
        } catch (Exception e) {
                e.printStackTrace();
        }
});
```

✓ Viết mã xử lý các phương thức xử lý sự kiện

@OnOpen

```
        sessions.put(session.getId(), session);
        this.broadcast("Someone joined the chat!");
```

@OnMessage

```
        try {
                this.broadcast(message);
        } catch (Exception e) {
                e.printStackTrace();
        }
```

@OnClose

```
        sessions.remove(session.getId());
```

```
        this.broadcast("Someone left the chat!");
@OnError
        try {
                session.close();
        } catch (IOException e) {
                e.printStackTrace();
        }
```

BÀI 2 (2 ĐIỂM)

Tạo trang web websocket-client.html và sử dụng WebSocket API để xây dựng ứng dụng chat client kết nối đến chat server ở bài 1 thực hiện các yêu cầu sau:

- ✓ Hiển thị tin nhắn chat nhận được từ server lên vùng tin nhắn
- ✓ Gửi tin nhắn chat đến server khi nhấp nút Send
- ✓ Đưa ra thông báo khi gặp lỗi và socket bị đóng

Hướng dẫn:

- ✓ Tổ chức mã trang web websocket-client.html như sau

```
<html>
<head>
        <meta charset="UTF-8">
        <title>Simple Chat - Websockets</title>
        <script src="js/text-chat.js"></script>
</head>
<body onload="init()">
        <div id="messages" style="height: 200px;     overflow: auto;"></div>
        <hr>
        <input id="message">
        <button onclick="send()">Send</button>
</body>
</html>
```

- ✓ Tổ chức file JavaScript text-chat.js như sau

```
var websocket = null; // biến giữ đối tượng WebSocket
// Được gọi tại sự kiện onload của trang web
function init() {
        // Mở kết nối đến chat server
        websocket = new WebSocket('ws://localhost:8080/websocket/text/chat');
        // Xử lý sự kiện chấp nhận kết nối từ server
        websocket.onopen = function(resp) {
        }
        // Xử lý sự kiện nhận tin nhắn chat từ server
```

```
        websocket.onmessage = function(resp) {
        }
        // Xử lý sự kiện lỗi từ server
        websocket.onerror = function(resp) {
        }
        // Xử lý sự kiện đóng kết nối từ server
        websocket.onclose = function(resp) {
        }
}
// Gửi tin nhắn chat đến server, được gọi khi nhấp vào nút Send
function send() {
        var message = document.getElementById("message").value;
        websocket.send(message);
        document.getElementById("message").value = '';
}
```

✓ Viết mã xử lý cho các sự kiện

```
WebSocket.onopen
        console.log("onopen", resp);
WebSocket.onmessage
        var message = resp.data;
        var html = document.getElementById('messages').innerHTML;
        document.getElementById('messages').innerHTML =
`${html}<p>${message}</p>`;
        console.log("onmessage", resp.data);
WebSocket.onerror
        alert('An error occured, closing application');
        console.log("onerror", resp);
WebSocket.onclose
        alert(resp.reason || 'Goodbye');
        console.log("onclose", resp);
```

PHẦN II

Xây dựng ứng dụng chat với cấu trúc tin nhắn chat (Message) được quy định là {text, type, count, sender}. Trong đó:

- ✓ Text: lời thoại
- ✓ Type: loại tin nhắn (0: vào, 1: ra, 2: lời thoại)
- ✓ Count: số clients hiện tại (null khi type là 2)
- ✓ Sender: username của người gửi (null với type là 0 hoặc 1)

BÀI 3 (2 ĐIỂM)

Sử dụng ServerEndpoint API để xây dựng Chat Server

Hướng dẫn:

- ✓ Tạo lớp Message mô tả cấu trúc dữ liệu của tin nhắn giao tiếp giữa client và server

```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Message {
   private String text;
   private int type;
   private String sender;
   private int count;
}
```

- ✓ Khai báo thư viện phụ thuộc hỗ trợ chuyển đổi JSON và Java Object

```
<dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
        <version>2.9.1</version>
</dependency>
```

- ✓ Xây dựng lớp MessageEncoder chuyển đổi Message Object sang chuỗi JSON

```
public class MessageEncoder implements Encoder.Text<Message> {
        private ObjectMapper mapper = new ObjectMapper();
        @Override
        public void destroy() {}
        @Override
        public void init(EndpointConfig config) {}
        @Override
        public String encode(Message message) throws EncodeException {
                try {
                        return mapper.writeValueAsString(message);
                } catch (JsonProcessingException e) {
                        throw new EncodeException(message, "Unable to encode");
                }
        }
}
```

✓ Xây dựng lớp MessageDecoder chuyển đổi chuỗi JSON sang Message Object

```
public class MessageDecoder implements Decoder.Text<Message> {
        private ObjectMapper mapper = new ObjectMapper();
        @Override
        public void destroy() {}
        @Override
        public void init(EndpointConfig config) {}
        @Override
        public Message decode(String json) throws DecodeException {
                try {
                        return mapper.readValue(json, Message.class);
                } catch (IOException e) {
                        throw new DecodeException(json, "Unable to decode");
                }
        }
        @Override
        public boolean willDecode(String json) {
                return json.contains("type") && json.contains("text");
        }
}
```

✓ Xây dựng ChatServerEndpoint

```
@ServerEndpoint(
        value = "/json/chat/{username}",
        encoders = MessageEncoder.class,
        decoders = MessageDecoder.class
)
public class JsonChatServerEndpoint {
        private static Map<String, Session> sessions = new HashMap<>();
        private void broadcast(Message message) {
        }
        @OnOpen
        public void onOpen(@PathParam("username") String username, Session
session){
        }
        @OnError
        public void onError(Session session, Throwable throwable) {
        }
        @OnMessage
        public void onMessage(Message message, Session session) {
        }
```

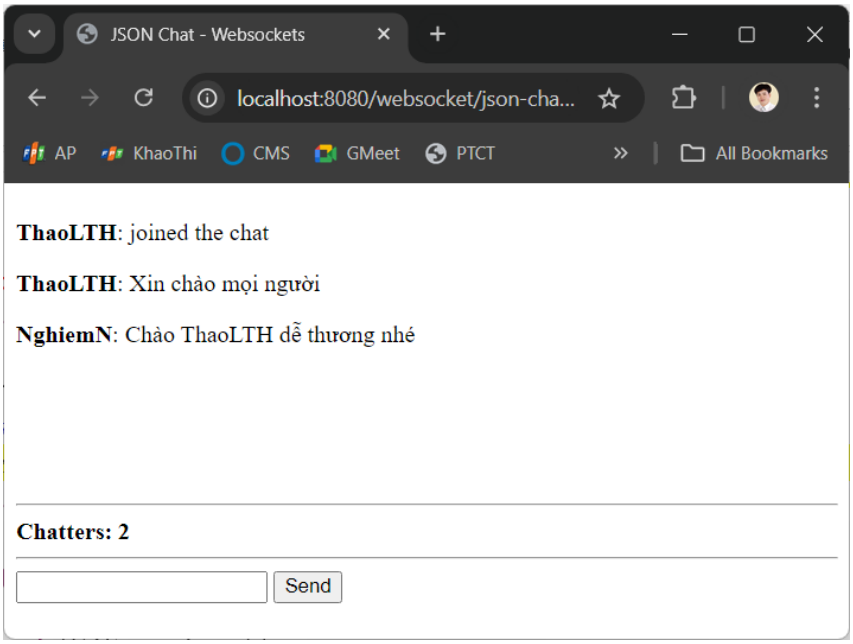
```
        @OnClose
        public void onClose(Session session) {
        }
}
```

✓ Viết mã cho các phương thức xử lý sự kiện

```
broadcast(Message)
        sessions.forEach((username, session) -> {
                try {
                        session.getBasicRemote().sendObject(message);
                } catch (IOException | EncodeException e) {
                        e.printStackTrace();
                }
        });
@OnOpen
        if (sessions.containsKey(username)) {
                throw new RuntimeException("Username already exists");
        } else {
                session.getUserProperties().put("username", username);
                sessions.put(username, session);
                Message message =
                                new Message("joined the chat", 0, username,
                                sessions.size());
                this.broadcast(message);
        }
@OnMessage
        this.broadcast(message);
@OnClose
        String username = (String) session.getUserProperties().get("username");
        sessions.remove(username);
        Message message = new Message("left the chat", 1, username,
sessions.size());
        this.broadcast(message);
@OnError
        try {
                session.close();
        } catch (IOException e) {
                throw new RuntimeException("Unable to close session");
        }
```

BÀI 4 (2 ĐIỂM)

Sử dụng WebSocket API để xây dựng Chat Client có giao diện như sau:



Hướng dẫn:

✓ Tạo json-chat.html có tổ chức mã như sau

```
<html>
<head>
	<meta charset="UTF-8">
	<title>JSON Chat - Websockets</title>
	<script src="js/json-chat.js"></script>
</head>
<body onload="init()">
	<div id="messages" style="height: 200px;  overflow: auto;"></div>
	<hr>
	<b id="client-count"></b>
	<hr>
	<input id="message">
	<button onclick="send()">Send</button>
</body>
</html>
```

✓ Tạo file json-chat.js có tổ chức mã như sau

```
var username = null;
var websocket = null;
function init() {
	while (username === null) {
		username = prompt("Enter username");
	}
```

```
        websocket = new
WebSocket(`ws://localhost:8080/websocket/json/chat/${username}`);
        websocket.onopen = function(resp) {
                console.log("onopen", resp);
        }
        websocket.onmessage = function(resp) {
        }
        websocket.onerror = function(resp) {
                alert('An error occured, closing application');
                console.log("onerror", resp);
        }
        websocket.onclose = function(resp) {
                alert(resp.reason || 'Goodbye');
                console.log("onerror", resp);
        }
}

function send() {
        var input = document.getElementById("message");
        var msg = {sender: username, text: input.value, type: 2}
        websocket.send(JSON.stringify(msg));
        input.value = '';
}
```

Viết mã cho hàm xử lý sự kiện WebSocket.onmessage

```
var msg = JSON.parse(resp.data);
var output = document.getElementById('messages');
output.innerHTML = `${output.innerHTML}<p><b>${msg.sender}</b>:
${msg.text}</p>`;
if(msg.type != 2){
        document.getElementById('client-count').innerHTML = `Chatters:
${msg.count}`;
}
```

BÀI 5 GIẢNG VIÊN CHO THÊM (2 ĐIỂM)