WebSocket을 활용한 웹 어플리케이션의 협업 기능 구현

조규철*, 장동건O
*인하공업전문대학 컴퓨터정보과,
O인하공업전문대학 컴퓨터정보과
e-mail: kccho@inhatc.ac.kr, jdajsl0415@naver.com

Collaboration Tool Implementation in Web Applications using WebSocket

Cho Kyu Cheol*, Jang Dong GeonO
*Department of Computer Information, Inha Technical College,
ODepartment of Computer Information, Inha Technical College

● 요 약 ●

본 논문에서는 협업이라는 주제로 WebSocket을 활용한 웹 어플리케이션의 협업 기능 구현을 통해 다양한 사용자들이 자유롭게 소통하고 의견을 전달 할 수 있도록 웹 사이트를 제작하였다. 웹 어플리케이션의 기능인 그림그리기, 채팅, 파일전송 등을 통해 사용자가 전달하려는 정보를 여러 방법을 통해 전달 할 수 있도록 하여 본 연구를 통해 구현된 도구가 협업에 도움을 줄 것으로 기대한다.

키워드: WebSocket, web applications, real-time communication, HTTP protocols

I. Introduction

최근 웹 어플리케이션에서 사용자들 간에 실시간 소통과 협업의 필요성이 높아지고 다양한 방면으로의 소통이 중요해졌다[1]. HTTP 프로토콜을 통한 단방향 통신으로는 원활한 소통에 한계가 있어, 양방향 통신이 가능한 채팅 및 소통 프로그램에 원활한 소통이 가능하도록 WebSocket 기술을 도입하였다.

기존의 HTTP 프로토콜은 단방향 통신에 적합하였으나, 실시간 상호작용이 중요한 현대 웹 환경에서는 한계가 있었다. 그래서 WebSocket을 활용함으로써 양방향 통신을 지원하고, 이에 따른 다양한 기능을 구현하여 사용자 경험을 향상시키고 공유하고자 한 것이 본 연구의 동기이다. WebSocket은 클라이언트와 서버 간 양방향 통신을 지원하여 실시간으로 데이터를 주고받을 수 있다.

양방향 통신을 이용한 실시간 데이터 송수신 기술은 채팅, 그림그리기, 파일 전송 등 다양한 기능을 구현할 수 있어사용자들의 편의와 정보를 전달하는데 적합하다. 또한, HTTP와는 달리 WebSocket은 항상 연결된 상태를 유지하므로, 매번 새로운 연결을 시도하지 않아도 되기 때문에 연결 비용을 줄이고 빠른 통신을 가능하게 한다[2].

HTTP는 그림 1과 같이 클라이언트가 명시적으로 요청을 보내고 서버는 이에 대한 응답을 주는 Request Response 모델을 따르며 WebSocket은 양방향 통신을 지원하며 클라이언트와 서버 간의 한번의 연결을 설정하면 양쪽에서 자유롭게 데이터를 송수신하는 예시이다.

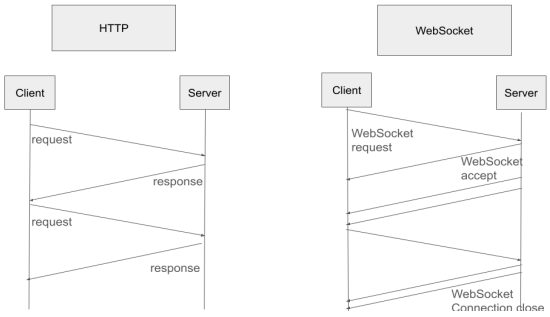

Fig. 1. HTTP vs WebSocket [2]

그림 2와 같이 HTTP에서 사용하는 기존 폴링(Polling)이나 롱 폴링(Long Polling) 방식보다 WebSocket은 더 낮은 지연 시간과 높은 실시간성을 제공한다. 이는 사용자들이 실시간으로 피드백을 받을 수 있게 함으로써 사용자 경험을 향상시킨다[3].

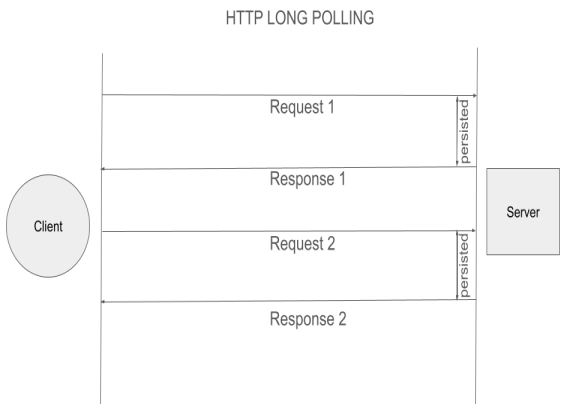

Fig. 2. Polling vs WebSocket[3]

Fig. 4. Drawing Data From Client

Fig. 5. Canvans Drawing using WebSocket

II. Collaboration Tool Implementation

WebSocket을 통한 양방향 통신은 웹 어플리케이션에서의 상호작용과 협업을 가능하게 한다. Websocket 활용을 통해 본 연구에서는 다양하고 현실적인 기능들을 구현하여 사용자 경험을 높이고자 하였다.

2.1 Canvas Drawing

WebSocket을 기반으로 한 양방향 통신은 그림 3과 같은 구현으로 그림 그리기 기능에 적용할 수 있다. 특히, 여러 사용자가 동시에 한 캔버스에 접근하여 그림 4와 같이 실시간으로 그림을 공유하고 수정할 수 있도록 구현할 수 있다. 이는 지리적으로 다른 위치에 있는 사용자들 간의 창의적 협업을 가능하게 하며, 그림 5와 같이 사용자들이 제 생각을 제시하고 표현할 수 있는 기능이다.

2.2 Sending Files

WebSocket을 통한 파일 처리는 그림 6와 같이 사용자가 웹 어플리케이션에서 파일을 선택하면, 해당 파일을 안전하게 서버로 전송하고 서버는 그림 7과 같이 실시간으로 수신하여 처리를 수행하는 기능을 제공한다. 파일 처리 과정은 사용자가 파일을 선택하면 클라이언트에서는 해당 파일을 읽기 위해 FileReader를 사용한다. FileReader를 통해 읽어온 파일 데이터는 WebSocket을 활용하여 서버로 전송된다. 서버는 WebSocket을 통해 실시간으로 수신하고, 수신된 파일 메시지를 적절히 처리하여 필요한 동작을 수행한다. 이러한 파일 처리 방식을 통해 사용자들은 실시간으로 파일을 교환함으로써 협업이 가능해진다.

```
canvas.addEventListener("mousemove", (e) => {
    if (!isDrawing) return;

    const x = e.clientX - canvas.getBoundingClientRect().left;
    const y = e.clientY - canvas.getBoundingClientRect().top;

    ctx.beginPath();
    ctx.moveTo(lastX, lastY);
    ctx.lineTo(x, y);
    ctx.lineWidth = 2; // 그림 두께 설정
    ctx.stroke();

    // 그림 그리기 이벤트를 WebSocket으로 서버에 전송
    const message = JSON.stringify({ type: 'draw', x: lastX, y: lastY, newX: x, newY: y, color: selectedColor });
    webSocket.send(message);

    lastX = x;
    lastY = y;
});
```

Fig. 3. Drawing Client using WebSocket

```
function sendFile() {
    var fileInput = document.getElementById('file');
    var file = fileInput.files[0];

    if (isAllowedFileType(file)) {
        // 추가 세션 정보 (예: 사용자 ID)를 포함합니다.
        var sessionData = {
            type: 'sessionInfo',
            user: loggedInUserId,
            // 다른 보내고 싶은 세션 데이터가 있다면 추가하세요.
        };
        webSocket.send(JSON.stringify(sessionData));

        // 파일 이름과 함께 첫 번째 청크를 보냅니다.
        webSocket.send('filename:' + file.name);

        var reader = new FileReader();
        reader.onload = function (e) {
            // 파일 데이터를 보냅니다.
            webSocket.send(e.target.result);
        };

        reader.onloadend = function () {
            // 파일 전송이 끝났음을 알립니다.
            webSocket.send('end');
            alert("파일 전송이 완료 되었습니다.");
        };

        // 파일을 ArrayBuffer로 읽습니다.
        reader.readAsArrayBuffer(file);
    } else {
        alert('유효하지 않은 파일 형식입니다. 텍스트 파일(.txt)
    }
}
```

Fig. 6. Send File Code

2.3 Real-time chat

실시간 채팅 기능은 기존의 HTTP 프로토콜보다 낮은 지연 시간과 빠른 응답 속도를 제공하여 사용자들 간의 효과적인 의사소통을 가능하게 한다. 업무 협업이나 원격 교육 환경 또는 실시간 질문 응답과 같은 상황에서 그림 8과 같이 메시지 전송과 그림 9과 같은 수신메시지를 통해 실시간으로 데이터를 송수신함으로써 온라인상에서 소통할 수 있는 큰 장점이다. 그리고 이러한 특성은 사용자들이 실시간으로 정보를 교환하고 협력할 수 있어 작업의 효율성을 향상할 수 있고 지연 없이 실시간으로 소통을 가능하게 지원한다.

```
function sendMessage() {
    var message = messageInput.value;
    if (message.trim() !== "") {
        if (message.length <= MAX_MESSAGE_SIZE) {
            const chatData = { type: 'chat', user: loggedInUserId, text: message };
            webSocket.send(JSON.stringify(chatData));

            // Assuming you want to display your own message immediately
            chatContent.innerHTML += loggedInUserId + ": " + message + "<br>";
        } else {
            console.log("Message too large, cannot send.");
        }
        messageInput.value = "";
    }
}
```

Fig. 8. Send Message to Client using WebSocket

```
클라이언트로부터의 채팅 메시지: {"type":"chat","user":"dg","text":"안녕하세요"}
```

Fig. 9. Data Receive From Clien

III. Conclusions

본 연구에서는 WebSocket을 도입하여 웹 어플리케이션에서의 상호작용과 협업을(이) 가능한 프로그램을 제작하였다. 양방향 통신의 강점을 살려 구현한 그림 그리기, 파일 전송, 채팅 등의 기능은 사용자 경험을 개선하였으며, 협업 기능을 강화하는 데 큰 의미를 두었다. 추후 연구는 안정적이고 확장 가능한 웹 기반 협업 환경을 제공하기 위해 WebSocket 기술의 활용을 계속하여 탐구할 것이다. 사용자들에게 개선된웹 환경을 제공함으로써 창의적인 아이디어를 실시간으로 공유할 수 있을 것으로 기대한다.

```
public void onBinaryMessage(ByteBuffer msg, boolean last, Session session) {
    System.out.println("이진 데이터 처리 ");
    // 이진 데이터를 저장할 ByteArrayOutputStream 초기화
    if (bos == null) {
        bos = new ByteArrayOutputStream();
    }
    // ByteBuffer에서 데이터를 읽어와 ByteArrayOutputStream에 쓰기
    while (msg.hasRemaining()) {
        bos.write(msg.get());
    }
    // 이진 데이터의 마지막 부분인 경우 파일의 끝을 처리
    if (last) {
        handleFileEnd(session, currentFilename, bos.toByteArray());
    }
}
```

```
private void handleFileStart(String filename) {
    currentFilename = filename;
    System.out.println("파일명: " + currentFilename);
    bos = new ByteArrayOutputStream();
}

private void handleFileEnd(Session session, String filename, byte[] fileData) {
    try {
        bos.flush();
        bos.close();

        // currentFilename이 null이 아니고 현재 처리 중인 파일 이름과 일치하면 DB에 저장
        if (currentFilename != null && currentFilename.equals(filename)) {
            saveFileDataToDB(session, fileData);
        } else {
            System.out.println("전송된 파일 이름과 현재 처리 중인 파일 이름이 일치하지 않습니다.");
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Fig. 8. WebSocket File Send Servlet

REFERENCES

[1] Li, M., Wang, J., & Wang, W. (2019). "A Survey on Real-Time Collaborative Editing Techniques." IEEE Access, 7, 104377-104393.

[2] WebSocket vs HTTP: How Are These Two Different. https://www.wallarm.com/what/websocket-vs-http-how-are-these-2-different

[3] Long Polling vs WebSockets - which to use in 2023 https://ably.com/blog/websockets-vs-long-polling