고급소프트웨어실습1 8주차 과제

컴퓨터공학 20172141 김미소

1. 주요 컴포넌트의 기능 및 구조

- ChatLogItem

```
function ChatLogItem({chat}) {

  return (
    <div>
    <UsernameContainer>
      {chat.username}<ChatContainer>: {chat.message}</ChatContainer>
    </UsernameContainer>
  </div>
  );

}
```

chat 객체 하나를 인자로 받아 한 줄로 표현한다.

"유저이름: 메시지"와 같은 형태로 채팅 로그 한 줄을 표현해주는 컴포넌트이다. CSS 사용을 위해 UsernameContainer 안에 ChatContainer가 존재한다.

- ChatLogTemplate

```
function ChatLogTemplate({chats}) {

  // chats 배열을 받아 저장해 둔 채팅 로그를 보여줌
  // ChatLogItem 컴포넌트 사용
  return (

    <ChatContainer>
      <h1>Chat Log</h1>
      {/* 8주차 실습 구현 */}
      {
        chats.map((chat) => {
          return <ChatLogItem chat={chat}/>
        })
      }
    </ChatContainer>
  );

}
```

Chats 배열을 받아 채팅 로그를 보여준다. ChatContainer 태그 안에 h1 태그를 두어 Chat Log라는 제목을 표시하고 그 아래에 채팅 로그를 쭉 보여준다. ChatLogItem 컴포넌트에 chat을 전달하여 "유저이름: 메시지"와 같은 형태로 표시한다.

- ChatTemplate

```
function ChatTemplate() {
    const [ chats, setChats ] = useState([])

    // chats 배열에 채팅 추가
    const getChatLog = (username, message) => {
        // 8주차 실습 구현
        setChats([...chats,{username, message}])
    }

    return (
        <Container>
            <MessenagerTemplate getChatLog={getChatLog} />
            <ChatLogTemplate chats={chats} />
        </Container>
    );
}
```

페이지 전체를 구성하는 역할을 하는 컴포넌트이다. getChatLog는 chats 배열에 새로운 chat을 추가하는 기능을 한다. setChats를 통해 기존의 chats뒤에 username, message를 추가한다. getChatLog는 MessenagerTemplate에서 사용된다. MessenagerTemplate은 사용자 이름과 메시지를 입력하여 채팅 메시지를 올릴 수 있는 기능을 한다. 그리고 ChatLogTemplate을 이용하여 채팅 로그를 보여줄 수 있다.

- MessenagerTemplate

```
// Sending Message 부분
function MessenagerTemplate({getChatLog}) {
  const [ MsgState, setMsgState ] = useState({ message: "", username: ""})

  const onMsgChange = (e) => {
    setMsgState({ ...MsgState, [e.target.name]: e.target.value })
  }

  // 메시지 입력 창에서 빈 경우에는 '메시지를 입력하세요'라는 내용의 경고 창 출력 후 리턴(alert 함수 사용)
  // 사용자의 이름 입력 창이 빈 경우에는 사용자 이름을 '이름없음'으로 저장
  // MsgState의 속성(키)들의 값을 빈 문자열로 초기화
  const onMsgSubmit = (e) => {
    e.preventDefault();
    // 8주차 실습 구현
    if (!MsgState.message) {
      alert('메시지를 입력하세요')
      return;
    }
    if(!MsgState.username){
      MsgState.username = '이름없음'
    }
    getChatLog(MsgState.username, MsgState.message)
    setMsgState({message:"", username:""})
  }
```

```
return (
    <Form onSubmit={onMsgSubmit}>
      {/* 8주차 실습 구현 */}
      <Title>Sending Message</Title>
      <UserName>
        <TextField
        name ="username"
        type = "text"
        onChange={e=> onMsgChange(e)}
        value={MsgState.username}
        label="User Name"/>
      </UserName>
      <MessageContents>
        <TextField
        name ="message"
        type = "text"
        onChange={e=> onMsgChange(e)}
        value={MsgState.message}
        variant="outlined"
        label="Message"/>
      </MessageContents>
      <Button>Send</Button>
    </Form>
);
}
```

Sending message 부분을 담당하는 컴포넌트로 사용자 이름과 메시지를 입력하고 send 버튼을 누르면 채팅 로그에 채팅이 올라갈 수 있도록 한다.

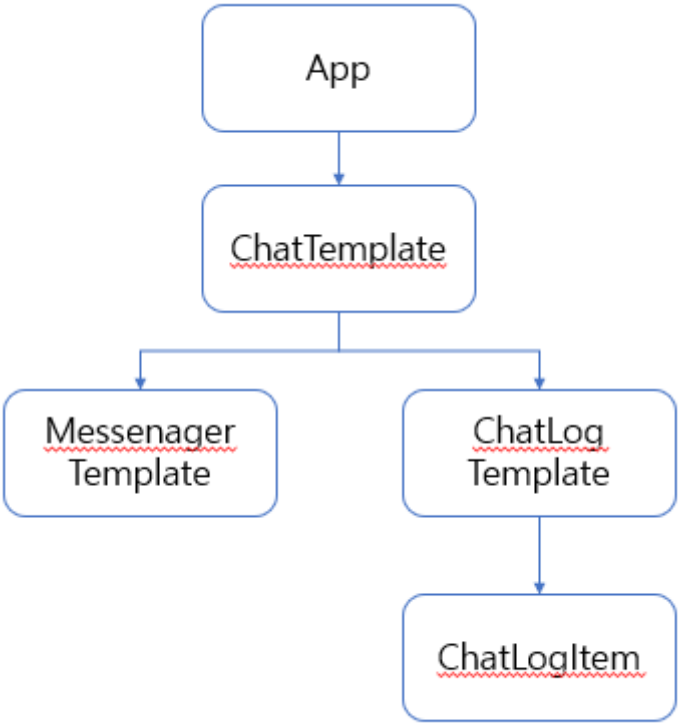
onMsgState는 form에서 사용될 onChange 함수이다.

그리고 onMsgSubmit은 form에서 사용되는 onSubmit 함수인데 메시지 입력을 하지 않고 메시지를 전송한 경우에는 '메시지를 입력하세요'라는 내용의 경고 창을 출력하고 리턴한다. 그리고 사용자 이름을 입력하지 않은 경우에는 사용자 이름을 '이름없음'으로 저장하고 메시지를 보낸다. 그리고 입력된 username과 message는 getChatLog를 이용하여 chats 배열에 추가된다. 그 이후에는 MsgState의 속성 값을 빈 문자열로 초기화한다.

Form 태그 안에는 username을 입력하는 UserName 태그와 message를 입력하는 MessageContents 태그가 있고 입력된 사용자 이름과 메시지를 전송할 수 있는 버튼이 존재한다.

각 컴포넌트 함수 아래에는 css에 해당하는 코드들이 작성되어 있다.

2. 컴포넌트 간 상호관계



컴포넌트의 전체적인 구조는 위와 같이 이루어져 있다. ChatTemplate 컴포넌트는 MessenagerTemplate 컴포넌트와 ChatLogTemplate 컴포넌트를 호출하여 사용하고 ChatLogTemplate은 채팅 로그를 쭉 보여주므로 ChatLogItem 컴포넌트를 사용하여 채팅 로그를 보여준다.