

Web プログラミング レポート 課題

24G1091 田代 壮

1 利用者向けの説明

ここでは、掲示板の利用方法について説明する。まず、掲示板には図 1 のように「名前」と「メッセージ」と書かれた隣にそれぞれ入力するスペースが表示されている。掲示板に何か書き込む場合には適当な「名前」と「メッセージ」を入力し、隣に「送信」と青い背景に囲まれたボタンが表示されているのでそこをクリックすることで掲示板に書き込むことができる。

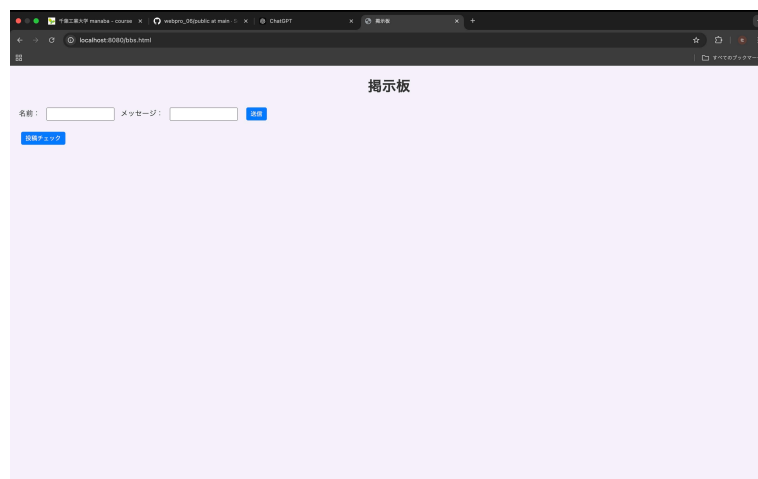


図 1 掲示板の初期表示

書き込んだ投稿を確認する場合には、「名前」などの下に青い背景で囲まれている「投稿チェック」というボタンが表示されているので、それをクリックすると投稿を確認することができる。

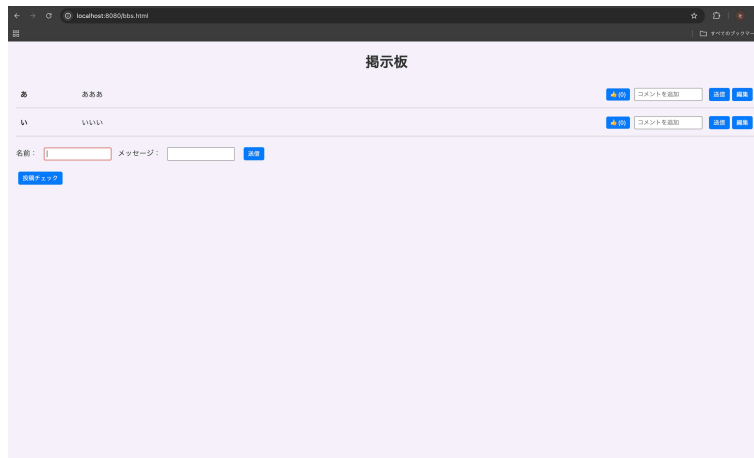


図 2 投稿チェック

次に、掲示板に投稿された文章に対しての機能についての説明をする。投稿された文章に対しての機能は主に 3 つある。投稿された文章には右端に青い背景で囲まれた「(0)」と「コメントを追加」という何か書くスペースと「送信」、そして「編集」というボタンが表示される。

1.1 いいね機能

各投稿には「いいね」ボタン () があり、これを押すことでその投稿に「いいね」を追加することができる。「いいね」ボタンを押すことで「」の横にカウントされている数字が一つ増える。また、「いいね」はカウント 100 まで送ることができる。

1.2 コメント機能

各投稿に対してコメントを入力するスペースがあり、他の投稿に対してコメントを送ることができる。コメントを入力して送信すると投稿に対してのコメントが追加される。一つの投稿に対して複数のコメントを送信することができ、上から順番に表示される。

1.3 投稿編集機能

投稿編集機能とは投稿したメッセージを後から編集することができる機能である。自分の投稿した文章の右端にある編集ボタンを押すことで、投稿を編集することができる。変更後の文章を入力して保存ボタンを押すことで変更が更新される。

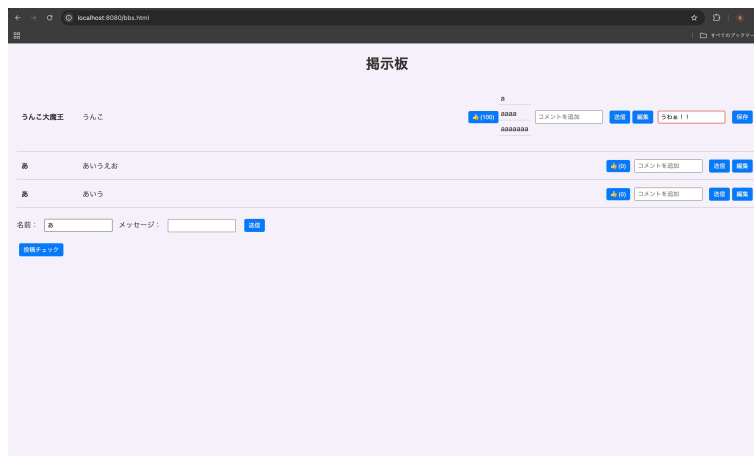


図 3 機能

2 管理者向けの説明

まず, サーバを立ち上げるための準備として node.js をインストールしておく. 「node -v」, 「npm -v」と順にターミナルで実行することでインストールされているかを確認することができる. サーバを立ち上げる時には, 「node app8.js」を実行することでサーバを立ち上げることができる.

サーバが起動すると, 「Server is running at http://localhost:8080」と表示される. その後, ブラウザで「http://localhost:8080」を開くと, 掲示板の動作を確認することができる.

2.1 掲示板の動作

掲示板にアクセスすることで, その度に新しい投稿が一覧で表示される. 管理者はその投稿やコメントを閲覧することが出来る.

2.2 いいね機能

特定の投稿に「いいね」を追加するエンドポイント (POST /bbs/:id/like) にリクエストが来ると, 指定された投稿の「いいね」数を一つ増やす. 流れとしては, 投稿 ID を受け取りその ID に対応する投稿を検索し, 見つかった投稿の「いいね」数を増やし, 更新された「いいね」数をレスポンスとして返す.

2.3 コメント機能

(POST /bbs/:id/comment) は指定された投稿に対してのコメントを追加するエンドポイントである. 投稿 ID を受け取り, その投稿にコメントを追加する. コメントリストを更新後にレスポンスとして返す.

2.4 投稿編集機能

指定された投稿を編集して更新するのに (PUT /bbs/:id) というエンドポイントを用いる. 投稿 ID と更新内容を受け取り, 対象の投稿を新しいメッセージに変更して更新する. 更新後の投稿内容をレスポンスとして

返す.

3 開発者向けの説明

ここでは通信時にどのようなデータが流れるかについて説明する. まず全ての通信は JSON 形式で行われる.

新規の投稿があるかを確認する場合には, 「`apiRequest('/check')`」を使ってサーバの (`/check`) エンドポイントにリクエストする. その後, (`checkRes.number`) で現在の投稿数が返される. 投稿を取得する場合は, 「`apiRequest('/read', "POST", start: number)`」を使ってサーバに (`/read`) エンドポイントにリクエストを送りレスポンスとして, 新しい投稿データとして (`readRes.messages`) に格納される. その後, 「`readRes.messages.forEach((post) => addPostToDOM(post));`」の実行することで一つずつ新しい投稿が表示される.

次に, 新しく投稿を追加する場合について説明する. まず, 「`document.querySelector('post').addEventListener('click',`」で「送信」ボタンにクリックイベントを登録する. 「`const name = document.querySelector('name').value;`」と 「`const message = document.querySelector('message').value;`」によって投稿されたデータを取得し 「`const res = await apiRequest('/post', "POST", name, message)`」の部分でサーバに (`/post`) のエンドポイントにリクエストを送り, 新しい投稿内容を送信する.

いかに通信の全体的な流れと github の URL を示す.

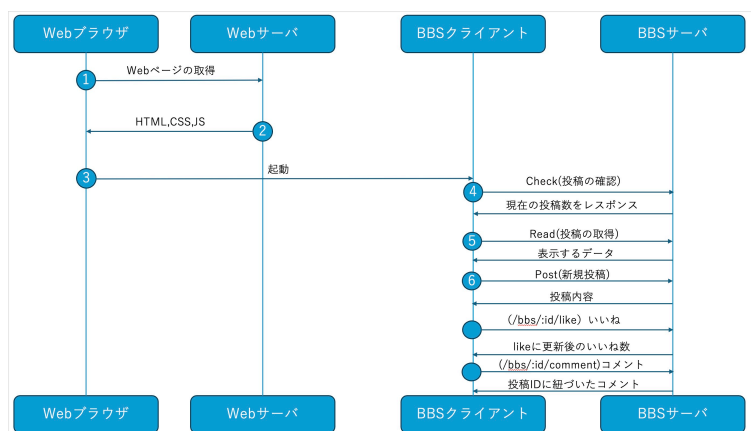


図4 全体的な流れ

github: https://github.com/Sou0202/webpro__06

4 授業で説明していない技術

1:async/await

async/await は非同期処理を行う構文で,API リクエストなど非同期同期に処理を実行するのに便利なものである.async は関数が非同期であることを示し,await を使用して非同期処理を同期的に書くことができる. 採用理由は, 非同期処理を使ってネットワークリクエストを行う場合別の方法もあるが,async/await 構文を用いることでコードがより簡潔にすることができるからである.

2:try-catch

try-catch 構文は、プログラム内で発生する可能性のあるエラーを処理するための仕組みである。エラーが発生してもプログラムが突然停止することなく、エラーを適切に処理し次の処理を続けることができます。try の中にエラーが起きる可能性のある文を書き、catch にそのエラーが起きた場合の対処法を書く。これにより、エラーが起きない場合はそのまま処理が続くが、エラーがもし起きても catch が実行され適切に処理される。採用理由は、try-catch を用いることでエラーの対処の柔軟性を得ることができるからである。エラーによるプログラムの突然の終了をなくすることができるからである。

3:JSON.stringify

JSON.stringify は JavaScript においてオブジェクトや配列を JSON 形式の文字列に変換するメソッドである。これにより、サーバーとの通信などでオブジェクトデータを文字列として送信することが可能となる。採用理由は、フォームから入力された内容（オブジェクトや配列）をサーバーに送信する際には、JavaScript オブジェクトを文字列化する必要があるためである。

4:addPostToDOM

addPostToDOM は引数として与えられた投稿情報に基づいて、HTML 要素を動的に生成し、ページに表示するための処理を行う。例えば、投稿者の名前、メッセージ、またそれに対するいいねやコメントを動的に生成しページに追加する。採用理由としては、機能の明確な分担である。addPostToDOM には、投稿情報を受け取り、HTML に投稿内容を追加するという明確な役割がある。そのため管理や修正もやりやすくなり、また再利用の場合にはこの関数を使いまわせば良いため有用性があるからである。