web プログラミングレポート

24G1052 久家力孔

2025 年 1 月 7 日

1　ソースコード

https://github.com/KugeRiku/webpro_06.git

2　利用者向け仕様書

2.1　サービス概要

本サービスでは，ユーザーが名前とメッセージを入力し，掲示板に投稿することができる．また表 1 に記した，ユーザエクスペリエンスを向上させるための三つの追加機能がある．

表 1　本サービスにおける機能

投稿の検索	キーワードを基に投稿を検索し、関連する結果のみを表示する
新規投稿の通知	新しい投稿があれば通知する
「いいね」機能	自分が気に入った投稿に「いいね」ボタンを押すことができる

2.2　画面のレイアウト

表 2 に，本サービスの基本的な画面構成を記した．

表 2　画面のレイアウトについて

投稿フォーム	ページ上部に、名前と新規投稿の入力欄、送信ボタンがある
投稿一覧	ページの左上部に表示されていく
検索バー	ページ中央部分に、検索したいキーワードの入力欄がある

2.3　サービス利用の流れ

メッセージを投稿するためには，名前と投稿したいメッセージを入力欄に入力する．そして入力欄の右部にある送信ボタンを押すことで、メッセージが掲示板に投稿される．投稿フォーム下部の投稿チェックボタンを押すことで，新しく投稿されたメッセージを確認することができる．検索バーにキーワードを入力してから検

索ボタンを押すことで，入力したキーワードのみを含むメッセージを表示することができる．

3　管理者向け仕様書

3.1　サービス概要

この掲示板サービスは，Node.js ベースで構築された Web アプリケーションであり，四つの主要機能が存在する．

- ユーザーが投稿を送信、閲覧できる機能.
- 投稿をキーワードで検索できる機能.
- 新規投稿を通知する機能.
- 各投稿に対して「いいね」を付けることができる機能.

3.2　必要ソフトウェア

サービスを立ち上げるために以下のソフトウェアが必要である

- Node.js
- npm（Node.js に同梱）
- Google Chrome, Firefox, Safari のいずれか

3.3　セットアップ手順

まず，プロジェクトのディレクトリに以下のファイルを配置する

- app7.js（サーバースクリプト）
- bbs.html（HTML 構造ファイル）
- bbs.js（クライアントサイドスクリプト）
- bbs.css（スタイルシート）

次に依存ライブラリをインストールするため、以下のコマンドを実行する

```
npm install express ejs
```

そして，以下のコマンドを実行してサーバーを起動する

```
node app7.js
```

成功時，次のメッセージが表示される

```
Example app listening on port 8080!
```

4　開発者向け仕様書

4.1　サービス概要

この掲示板サービスは，Node.js ベースで構築された Web アプリケーションである．サーバーサイドでは express フレームワークを利用して HTTP リクエストを処理し，クライアントサイドでは，JavaScript による投稿の送信，検索，通知機能を実現した．

4.2　サーバーサイド（app7.js) の構造と変数の説明

変数について

- bbs：配列形式で掲示板の投稿データを保持。投稿は例として以下のような構造を持つ

```
{ name: "Riku", message: "Hello", likes: 3 }
```

- app：Express のインスタンスを保持

投稿の送信 (/post)

- メソッド：POST
- データ形式の名前：application/x-www-form-urlencoded
- 送信データの具体例

```
name=Riku&message=Hello
```

- 説明：クライアントから送信された name と message をサーバーの掲示板データに追加
- レスポンスデータの形式
  - データ形式の名前：JSON
  - レスポンスデータの具体例

```
{ "number": 3 }
```

  - 説明：サーバーに保存されている投稿の総数（number）を返す

投稿の取得 (/read)

- メソッド：POST
- データ形式の名前：application/x-www-form-urlencoded
- 送信データの具体例

```
start=10
```

- 説明：start で指定したインデックス以降の投稿をサーバーから取得
- レスポンスデータの形式
  - データ形式の名前：JSON
  - レスポンスデータの具体例

```
{
    "messages": [
        { "name": "Riku", "message": "Hello", "likes": 3 },
        { "name": "Sora", "message": "Hi there!", "likes": 0 }
    ]
}
```

  - 説明：取得された投稿データを配列として返す

新規投稿数の確認 (/check)

- メソッド：POST
- データ形式の名前：application/x-www-form-urlencoded
- 送信データ：なし
- 説明：現在の投稿数を取得し、クライアントに通知
- レスポンスデータの形式
  - データ形式の名前：JSON
  - レスポンスデータの具体例

```
{ "number": 15 }
```

  - 説明：掲示板に存在する投稿の総数を返す

検索機能 (/search)

- メソッド：POST
- データ形式の名前：application/x-www-form-urlencoded
- 送信データの具体例

```
keyword=Hello
```

- 説明：指定された keyword を含む投稿を検索し、結果を返す
- レスポンスデータの形式
  - データ形式の名前：JSON
  - レスポンスデータの具体例

```
{
    "results": [
        { "name": "Riku", "message": "Hello", "likes": 3 }
    ]
```

```
}
```

  - 説明：keyword を含む投稿データを配列として返す

いいね機能 (/like)

- メソッド：POST
- データ形式の名前：application/x-www-form-urlencoded
- 送信データの具体例

```
id=0
```

- 説明：指定された投稿（id で識別）の「いいね」数をインクリメント
- レスポンスデータの形式
  - データ形式の名前：JSON
  - レスポンスデータの具体例

```
{ "success": true, "likes": 4 }
```

  - 説明：
    * success：処理が成功したかを示す（true または false）。
    * likes：投稿の更新後の「いいね」数。

4.3 クライアントサイド（bbs.js) の構造と変数の説明

変数について

- number：現在表示中の投稿数を管理
- lastCheckedNumber：最後にチェックした投稿数を記録

主な関数とその役割

- renderPost(post, id)：投稿データを HTML 要素として生成し、掲示板に追加
- loadPosts(start)：指定インデックス以降の投稿をサーバーから取得し、表示
- setInterval：一定間隔で新規投稿数をチェックし、通知

4.4 データのフロー

以下にクライアントからサーバーへのリクエストの処理フローを示す

1. クライアントが/post エンドポイントにデータを送信（例：name=Riku&message=Hello）
2. サーバーは express.urlencoded でデータを解析し、req.body に保存
3. bbs.push() を使い、投稿データを bbs 配列に追加

4. 処理が成功すると、サーバーは投稿数（例：{"number": 3}）を JSON 形式で返す

同様に，/read，/search，/like などのエンドポイントも，リクエストデータを処理し，必要なレスポンスを生成する．

4.5 独自使用技術の概要と採用理由

DOMContentLoaded

- **概要**：HTML の解析が完了し、DOM ツリーの読み込み完了後に発火するイベント
- **採用理由**：HTML 構造が準備されていない状態でスクリプトが実行されるエラーを防ぐため

forEach

- **概要**：配列の各要素に対して、一つずつ処理を行うためのメソッド
- **採用理由**：シンプルで可読性が高く、ループ処理を簡潔に記述できるため
- **例**

```
response.messages.forEach((post, id) => renderPost(post, id));
```

テンプレートリテラル（${response.likes}）

- **概要**：バッククォート（‘）で囲んだ文字列中に変数や式を埋め込む構文
- **採用理由**：ボタンの表示内容を分かりやすく設定するため
- **例（「いいね」の記号は pdf に表示されないので’b’ で代用）**

```
likeButton.innerText = ‘b ${response.likes}‘;
```

setInterval

- **概要**：一定時間ごとに指定した関数を繰り返し実行する
- **採用理由**：リアルタイムでの新規投稿チェックを実現するため
- **例**：

```
setInterval(() => {
    fetch("/check", { method: "POST" })
        .then((response) => response.json())
        .then((data) => alert(‘新しい投稿が ${data.number} 件あります ‘));
}, 5000);
```