

# Web プログラミング レポート課題

25G1087 粒良 梁雅

2025 年 12 月 27 日

## 1 github のリポジトリの URL

[https://github.com/tsubutsubu00/webpro\\_06.git](https://github.com/tsubutsubu00/webpro_06.git)

## 2 開発者向けマニュアル

### 2.1 1 つ目: はま寿司メニュー一覧

#### 2.1.1 概要

本 web アプリケーションは、現在はま寿司で取り扱っているメニューを効率的に管理・閲覧するためのシステムである。このあと紹介する 2 つ目、3 つ目と異なり、データ量が多いため、情報をカテゴリごとに分類し、その後メニュー一覧に飛ぶようしている点が特徴である。また、メニュー一覧にそれぞれ追加、編集、削除が行える機能を追加した。

#### 2.1.2 データ構造

はま寿司カテゴリ一覧のデータ構造を表 1 に、はま寿司メニュー一覧のデータ構造を表 2 に示す。本 web アプリケーションでは、すべてのメニューデータを `hama_all_menu` に格納している。この `hama_all_menu` は、カテゴリ名である `limited_menu` や `nigiri` などを持っているオブジェクトであり、値として各メニューの配列を保持する多階層構造となっている。このような設計になつてているため、サーバー側で `req.params.url` を変数として利用して、動的にデータを切り替えて取得することが可能である。また、項目内の `id` は、number 型で定義している。この `id` を活用して、詳細表示や編集、削除を行う際に特定のデータを識別するためのパスパラメータとして利用している。

**表 1 はま寿司カテゴリ一覧のデータ構造**

| 項目名 | データ型   | 説明             | 出力例                    |
|-----|--------|----------------|------------------------|
| id  | number | 各カテゴリに割り当てた ID | 0,1,2 など               |
| url | string | パラメータ名         | limited_menu.nigiri など |
| tag | string | カテゴリの名称        | 期間限定, にぎりなど            |

**表 2 はま寿司メニュー一覧のデータ構造**

| 項目名         | データ型   | 説明                  | 出力例                       |
|-------------|--------|---------------------|---------------------------|
| id          | number | 各メニューに割り当てた ID      | 0,1,2 など                  |
| name        | string | メニューの名称             | 巻き寿司など                    |
| price       | string | メニューの価格             | 110 円 (税込), 176 円 (税込) など |
| suuryou     | string | メニューの数量             | 1 貢, 2 貢など                |
| omochikaeri | string | そのメニューがお持ち帰りできるかの可否 | お持ち帰り可など                  |

### 2.1.3 ページ遷移

本 web アプリケーションがどのようにページ遷移をするかを表 3 に、それらを簡潔にまとめた図を図 1 に示す。本 web アプリケーションでは、まず /menu にてカテゴリ一覧を表示し、カテゴリ選択をすると指定されたメニュー一覧 /menu/:url にページ遷移する仕様となっている。また、本システムは追加、編集、削除の CRUD 操作にも対応しており、それらの操作を行うと、ユーザーが指定していたカテゴリのメニュー一覧へリダイレクト処理が行われる設計となっている。

**表 3 はま寿司メニュー一覧のページ遷移**

| 目的     | リソース名                     | HTTP メソッド | 遷移先                        |
|--------|---------------------------|-----------|----------------------------|
| カテゴリ一覧 | /menu                     | GET       | hama.ejs                   |
| メニュー一覧 | /menu/:url                | GET       | hama_menu.ejs              |
| 追加フォーム | /menu/:url/create         | GET       | /public/hama_menu_new.html |
| 新規追加   | /menu/:url/create         | POST      | /menu/:url (メニュー一覧に戻る)     |
| 詳細表示   | /menu/:url/:number        | GET       | hama_menu_detail.ejs       |
| 編集     | /menu/:url/edit/:number   | GET       | hama_menu_edit.ejs         |
| 更新     | /menu/:url/update/:number | POST      | /menu/:url (メニュー一覧に戻る)     |
| 削除     | /menu/:url/delete/:number | GET       | /menu/:url (メニュー一覧に戻る)     |

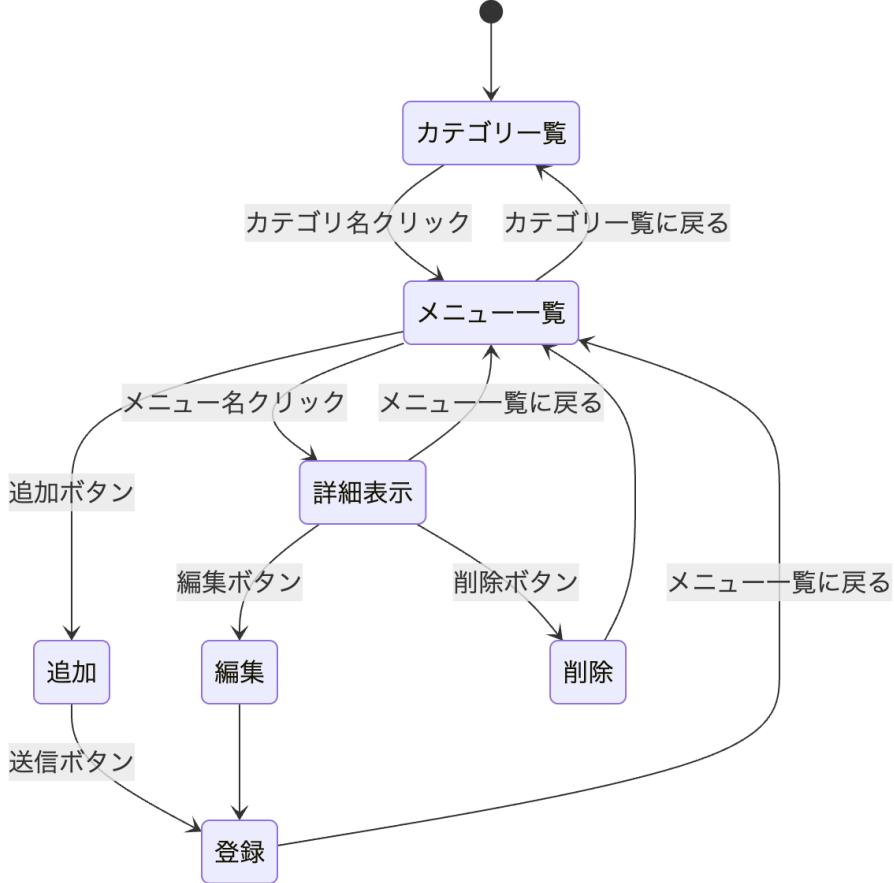


図1 はま寿司メニュー一覧のページ遷移図

#### 2.1.4 リソースごとの機能の詳細

第2.1.3節で作成した表を基に、リソースごとの機能の詳細について説明する。/menuのカテゴリを一覧表示するリソースでは、すべてのカテゴリ情報を持つ hama\_menu を hama.ejs に渡し、ユーザーに見たいページの選択を提示する役割を持つ。ユーザーに選択されたカテゴリは、その後、メニュー一覧へと遷移する。/menu/:url のメニューを一覧表示するリソースでは、パラメータ url をキーとして利用し、連想配列である hama\_all\_menu から該当するカテゴリの配列を動的に抽出した後に、表示する役割を持っている。ユーザーが選んだメニューを押すと、詳細表示へと遷移する。/menu/:url/:number の詳細表示を行うリソースでは、:number を固有の ID として扱う。find メソッドを用いて、item.id == number により検索をかけることで、配列の順序に依存しないようなデータ抽出を行うことができる。このようなロジックを使った理由として、他の web アプリケーションで使用している URL の数字を配列を使って何番目に入れるかのロジックはシンプルな構造であるが、はま寿司のシステムは、メニューの追加や削除が頻繁に行われる可能性が高いという観点から不適切であると判断したためである。具体的には、メニューを削除

をした後に追加を行うと、それ以降のデータの順番がずれてしまうが、データに紐づいている ID は変化しないためである。/menu/:url/create の新規追加を行うリソースでは、res.sendFile を用いて、hama\_new.html を読み込む。フォーム送信時には、既存の最終データの ID に 1 を加算する new\_id を生成し、新しく作成したデータを配列の末尾に push により追加する役割を持っている。/menu/:url/edit/:number の編集を行うリソースでは、詳細表示でも利用した find メソッドで特定したオブジェクトの各要素を、編集画面で表示し編集する役割を持っている。/menu/:url/update/:number の更新を行うリソースでは、詳細表示や編集と同様 find メソッドで特定したオブジェクトの各要素を、フォームから送信されたデータで直接上書きする機能を持っている。/menu/:url/delete/:number の削除を行うリソースでは、:number によってパスパラメータで指定された ID と合致するデータを find メソッドにより特定した後、splice メソッドによって配列から削除する役割を持っている。また、誤って削除ボタンを押した際に onclick="return confirm()" を導入することで、誤って削除することを防ぐ利用者への配慮も行った。

## 2.2 2つ目:tex 数学記号一覧

### 2.2.1 概要

本 web アプリケーションは、LaTex で使用される数学記号を一覧表示するシステムである。閲覧するだけでなく、数学記号をブラウザ上で追加・編集・削除のできる機能を備えている。

### 2.2.2 データ構造

tex 数学記号一覧のデータ構造を表 4 に示す。項目名に示しているこれらのデータは、すべて tex\_data に格納している。id は、number 型で定義しており、それ以外の要素は、文字列として出力するために、string 型で定義されている。特に、id の数値を活用して、詳細表示や編集、削除を行う際に特定のデータを識別するためのパスパラメータとして利用している。

表 4 tex 数学記号一覧のデータ構造

| 項目名     | データ型   | 説明             | 出力例                      |
|---------|--------|----------------|--------------------------|
| id      | number | 数学記号に割り当てた ID  | 0,1,2 など                 |
| symbol  | string | tex で出力される数学記号 | =, ≠ など                  |
| name    | string | 数学記号の名称        | 等号, 不等号など                |
| command | string | tex で使うコマンド    | =, \neq など               |
| genre   | string | 数学記号のジャンル      | 等号, 不等号, 演算子など           |
| mean    | string | 数学記号の持つ意味      | 等しいことを示す, 等しくないことを示す, など |

### 2.2.3 ページ遷移

本 web アプリケーションがどのようにページ遷移をするかを表 5 に、それらを簡潔にまとめた図を図 2 に示す。本システムのトップページは、\tex で、「一覧表示」の役割を担っている。CRUD の要素である追加、編集、削除を行うと、遷移先としてトップページである\tex にリダイレクト処理によって戻るような設計となっている。この CRUD によって、ユーザーインターフェース向上につなげている。

表 5 \tex 数式記号一覧のページ遷移

| 目的     | リソース名               | HTTP メソッド | 遷移先                  |
|--------|---------------------|-----------|----------------------|
| 一覧表示   | /tex                | GET       | \tex.ejs             |
| 追加フォーム | /tex/create         | GET       | /public/tex_new.html |
| 詳細表示   | /tex/:number        | GET       | \tex_detail.ejs      |
| 追加     | /tex                | POST      | /tex (一覧表示に戻る)       |
| 編集     | /tex/edit/:number   | GET       | \tex_edit.ejs        |
| 更新     | /tex/update/:number | POST      | /tex (一覧表示に戻る)       |
| 削除     | /tex/delete/:number | GET/POST  | /tex (一覧表示に戻る)       |

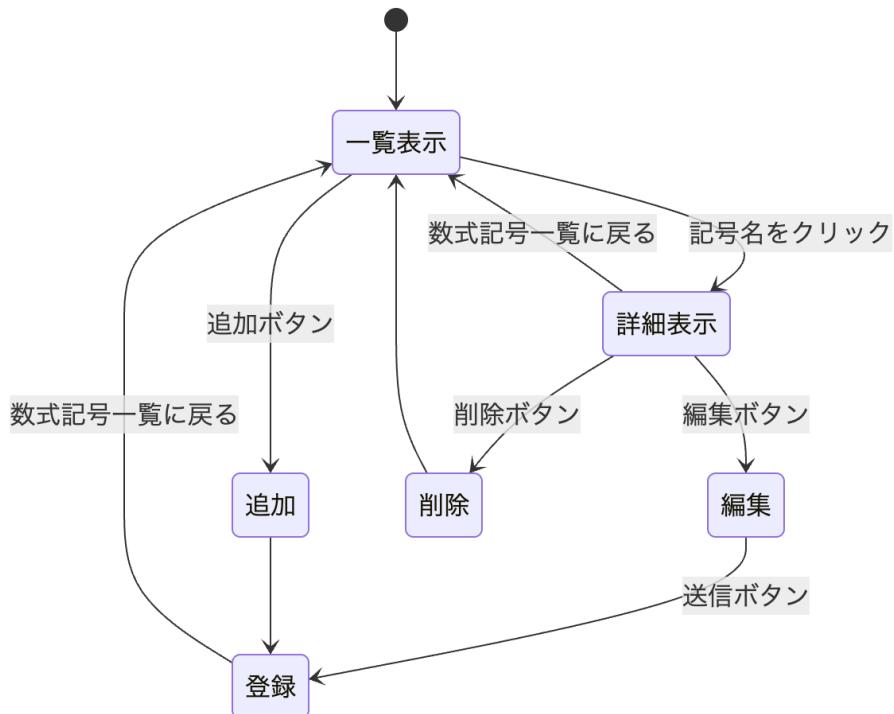


図 2 \tex 数式記号一覧のページ遷移図

#### 2.2.4 リソースごとの機能の詳細

第 2.2.3 節で作成した表を参考に、リソースごとの各機能について説明する。一覧表示することが目的である /tex のリソースでは、配列 `tex_data` の中に格納されているデータをすべて取得し、`tex.ejs` に渡して表形式で表示する役割を持っている。その /tex に表示されるデータは URL として /tex/:number の詳細表示のリンクへとべるようになっている。/tex/:number のリソースでは、URL から取得したパスパラメータ `:number` を配列の番号として割り当て、`tex_data[number]` によって特定のデータを抽出して `tex_detail.ejs` を使って表示する役割を持っている。/tex/create や /tex の追加処理を行うリソースでは、追加ボタンを押すと、/tex/create にアクセスし、`tex_new.html` にリダイレクトし、入力画面を表示する。フォーム送信時には、既存の最終データの ID に 1 を加算する `new_id` を生成し、新しく作成したデータを配列の末尾に `push` により追加する役割を持っている。/tex/edit/:number の編集を行うリソースでは、`tex_data[number]` から現在の値を読み込み、それを編集画面で表示する役割を持っている。/tex/update/:number の更新を行うリソースでは、送信されたデータで配列の該当する部分 `tex_data[number]` の各要素を直接上書きする機能を持っている。/tex/delete/:number の削除を行うリソースでは、パスパラメータで指定した場所を `splice` というコードの第 1 引数に渡し、配列から該当する要素を 1 件削除する。削除を行ったあとは、tex 数学記号一覧のページである /tex にリダイレクトする。また、誤って削除ボタンを押した際に `onclick="return confirm()` を導入することで、誤って削除することを防ぐ利用者への配慮も行った。

### 2.3 3つ目: 大乱闘スマッシュブラザーズファイター一覧

#### 2.3.1 概要

本 web アプリケーションは、nintendo Switch 用ゲームソフト「大乱闘スマッシュブラザーズ」(以降、スマブラ) に登場するファイター (キャラクター) 全員を一覧表示するシステムである。一覧表示だけでなく、ファイターの追加や編集、削除を行うことができる。

#### 2.3.2 データ構造

スマブラファイター一覧のデータ構造を表 6 に示す。項目名として示しているデータは、すべて `sumabura_data` に格納している。第 2.2.2 節と同様に、`id` は、`number` 型で定義しており、それ以外の要素は、文字列として出力するために、`string` 型で定義している。特に、`id` の数値を活用して、詳細表示や編集、削除を行う際に特定のデータを識別するためのパスパラメータとして利用している。

表6 スマブラファイター一覧のデータ構造

| 項目名     | データ型   | 説明                          | 出力例                |
|---------|--------|-----------------------------|--------------------|
| id      | number | 各ファイターに割り当てた ID             | 0,1,2 など           |
| name    | string | ファイター名                      | マリオ, ドンキーコングなど     |
| series  | string | 各ファイターが登場する作品               | スーパーマリオ, ドンキーコングなど |
| nannido | string | プレイヤーの上位 5%である VIP に到達する難易度 | ★★★★☆など            |

### 2.3.3 ページ遷移

本 web アプリケーションがどのようにページ遷移をするかを表7に、そのページ遷移図を図3に示す。本システムのトップページは、/sumabura で、「一覧表示」の役割を担っている。第2.2.3節と同様に、CRUD の要素である追加、編集、削除を行うと、遷移先としてトップページである /sumabuta にリダイレクト処理によって戻るような設計となっている。このCRUDによって、ユーザーインターフェース向上につなげている。

表7 スマブラファイター一覧のページ遷移

| 目的     | リソース名                    | HTTP メソッド | 遷移先                       |
|--------|--------------------------|-----------|---------------------------|
| 一覧表示   | /sumabura                | GET       | sumabura.ejs              |
| 追加フォーム | /sumabura/create         | GET       | /public/sumabura_new.html |
| 詳細表示   | /sumabura/:number        | GET       | sumabura_detail.ejs       |
| 追加     | /sumabura                | POST      | /sumabura (一覧表示に戻る)       |
| 編集     | /sumabura/edit/:number   | GET       | sumabura_edit.ejs         |
| 更新     | /sumabura/update/:number | POST      | /sumabura (一覧表示に戻る)       |
| 削除     | /sumabura/delete/:number | GET/POST  | /sumabura (一覧表示に戻る)       |

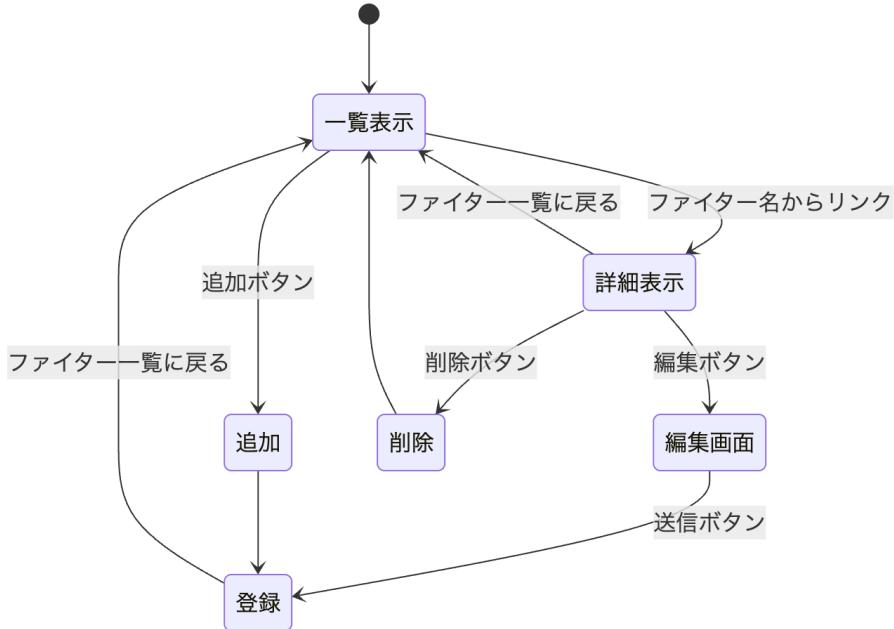


図3 スマプラファイター一覧のページ遷移図

#### 2.3.4 リソースごとの機能の詳細

第2.3.3節で作成した表を参考に、リソースごとの各機能について説明する。/sumaburaの一覧表示するリソースでは、配列 `sumabura_data` に格納しているデータを `sumabura.ejs` に渡して表形式で表示する役割を持っている。その /sumabura に表示されるデータは URL として機能し、/sumabura/:number の詳細表示のリンクとしてとべるようになっている。/sumabura/:number の詳細表示を行うリソースでは、URL から取得したパスパラメータ `:number` を配列の番号として割り当て、`sumabura_data[number]` によって特定のデータを抽出して `sumabura_detail.ejs` で表示する役割を持っている。/sumabura/create や /sumabura の追加の処理を行うリソースでは、追加ボタンを押すと、/sumabura/create にアクセスし、`sumabura_new.html` にリダイレクトすることで、入力画面を表示する。フォーム送信時には、既存の最終データの ID に 1 を加算する `new_id` を生成し、新しく作成したデータを配列の末尾に `push` により追加する役割を持っている。/sumabura/edit/:sumabura の編集を行うリソースでは、`sumabura_data[number]` から、現在の値を読み込み、それを編集画面で表示する役割を持っている。/sumabura/update/:number の更新を行うリソースでは、送信されたデータで配列の該当する部分 `sumabura_data[number]` の各要素を直接上書きする機能を持っている。/sumabura/delete/:number の削除を行うリソースでは、パスパラメータで指定した場所を `splice` というコードの第 1 引数に渡し、配列から該当する要素を 1 件削除する。削除を行ったあとは、ファイター一覧のページである /sumabura にリダイレクトする。また、誤って削除ボタンを押した際に `onclick="return confirm()"` を導入することで、誤つて削除することを防ぐ利用者への配慮も行った。

### 3 管理者向けマニュアル

#### 3.1 インストール方法

本 web アプリケーションを利用するには、JavaScript 実行環境を整える必要がある。具体的には、`node.js` をダウンロードする必要がある。`node.js` は JavaScript で OS の機能にアクセスするプログラムを書くことができるため、導入が必須である。

そのような `node.js` をダウンロードするには、ターミナルを開き、ルートディレクトリで以下のコマンドを実行する。

```
1 nodebrew install stable  
2 npm install -g npm
```

Listing 1 `node.js` のダウンロード方法

次に、`node.js` がダウンロードできているか確認するために、以下のコードを実行する。

```
1 node -v
```

このコードを実行したら、v24.3.0 などのように表示されれば、`node.js` のダウンロードが完了し、実行環境の構築完了である。

#### 3.2 起動方法

ターミナルを開き、対象ディレクトリ（ここでは `webpro_06`）まで移動し、以下のコードを実行することで Web サーバーを立ち上げることができる。

```
1 node app_kadai.js
```

Listing 2 Web サーバーの起動方法

起動に成功すると、ターミナル上に Example app listening on port 8080! というメッセージが出力される。

### 3.3 起動できない場合の対処法

起動ができない場合の主な原因は 2 つある。第一に、ポートを複数開いている可能性がある。具体的には、すでに他のプログラムが 8080 番ポートを使用している場合、起動に失敗する。このような場合は、使っていない方のプログラムのポートを終了させる必要がある。第二に、ディレクトリ指定を間違える原因である。これは、`app_kadai.js` のあるディレクトリではない場所でソースコード 2 を実行してしまうことで起動できない場合がある。このような場合においては、`ls` コマンドで現在どの位置にいるかを確認し、正しいディレクトリへ移動することで解決することができる。

### 3.4 終了方法

サーバーを停止させるには、ソースコード 2 を実行中のターミナルで以下のコマンドを入力することで停止することができる。

```
1 Ctrl + C
```

**Listing 3 Web サーバーの終了方法**

### 3.5 わかっている不具合

わかっている不具合として、サーバーを終了すると、追加・編集・削除を行ったにも関わらず、初期化してしまう不具合がある。このような不具合を解決するためには、外部データベースを活用することで解決することができる。

## **4 利用者向けマニュアル**

**4.1 概要**

**4.2 基本的な操作の流れ**

**4.3 操作手順の詳細**

**4.3.1 システムの起動**

**4.3.2 データの閲覧方法**

**4.3.3 新しくデータを登録する方法**

**4.3.4 既存データの修正方法**

**4.3.5 データの削除方法**