Web プログラミング 最終課題レポート

251134 横居 克憲

2025 年 12 月 27 日

ソースコード (GitHub リポジトリ URL):
https://github.com/shibakatsu-oss/webpro_06

1　利用者向け仕様書

1.1　システム概要

本システムは，うまい棒の情報を管理・閲覧するための Web アプリケーションである．利用者は，うまい棒の味，価格，カロリーなどを一覧で確認できるほか，詳細情報の閲覧や新規登録が可能である．

1.2　利用方法

普段 Web サイトを閲覧する際に使用しているブラウザを起動し，http://localhost:8080/umaibo を入力してアクセスする．図 1 のような画面が表示されたらサイトを開けている状態である．

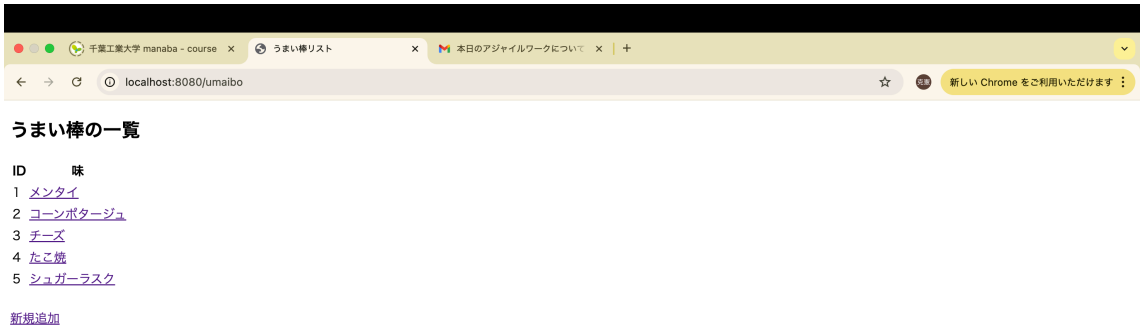


図 1　うまい棒管理システムのサイトを開いた際に表示される画面の例

1.3　操作方法

1.3.1　一覧画面

トップページにアクセスすると，図 1 のような現在登録されているうまい棒の一覧が表示される．ここには，これまでに登録されたうまい棒が一覧表示でみれる．

1.3.2　詳細表示

詳細を表示したい場合は，画面内の青色や紫色になっているうまい棒の味をクリックする．一例として，「コーンポタージュ」をクリックしたら図 3 のように，価格やカロリー数などの詳細が書かれた画面が表示される．一覧画面に戻りたい場合は図**??**にて赤い線で囲っている「一覧に戻る」をクリックする．

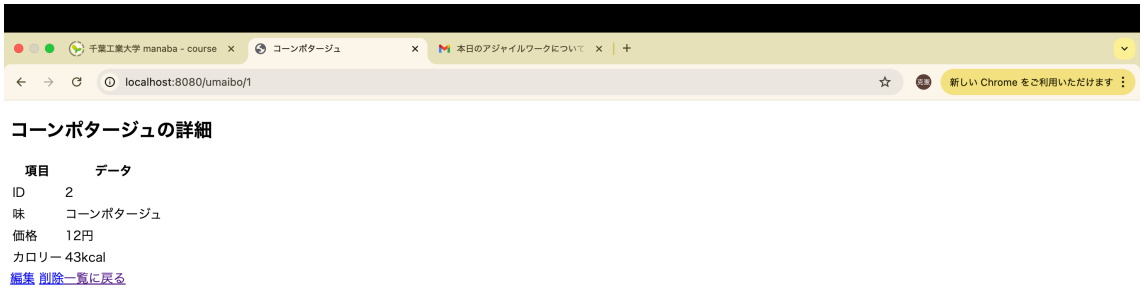


図 2　うまい棒管理システムのサイトのコーンポタージュのページを開いた際に表示される画面の例

図 3　うまい棒管理システムのサイトのどれかのページを開いた際に表示される画面で一覧表示のボタンを示している画像

1.3.3　新規登録

図 4 のように一覧画面下部の「新規追加」ボタンを押すと，図 5 の登録画面に遷移する．そこで必要な情報を入力して「登録」を押すとデータが保存される．

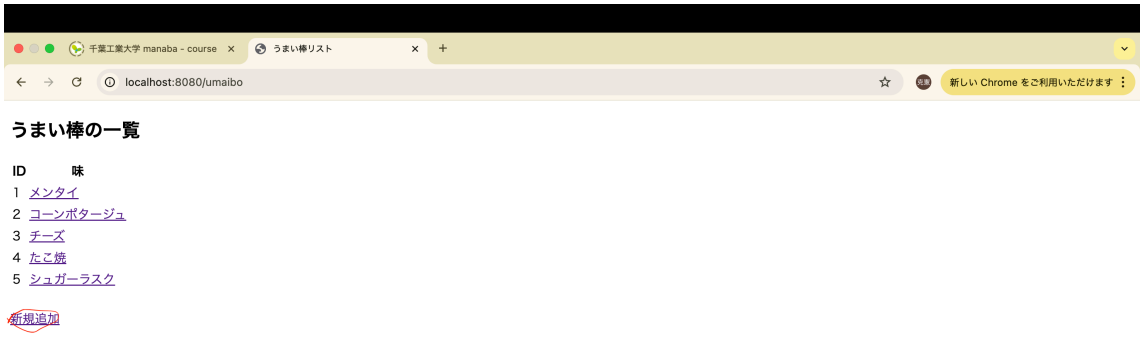
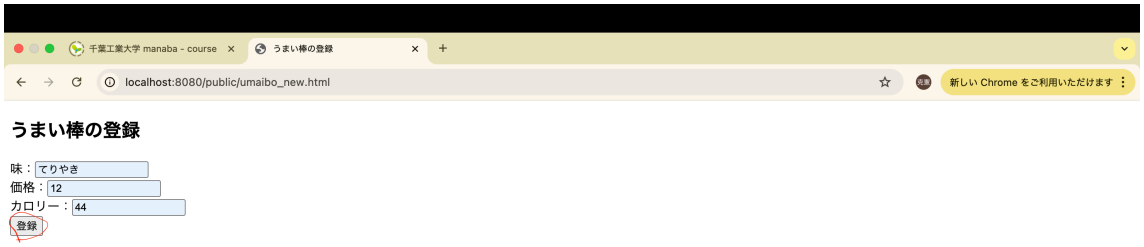


図 4　うまい棒管理システムのサイトを開いた際に表示される画面で新規登録のボタンを示している画像



「味」，「価格」，「カロリー」の項目を埋めたら，まるで囲ってる登録ボタンを押す！！

図 5　うまい棒管理システムのサイトの新規登録ページを開いた際に表示される画面で登録のボタンを示している画像

1.3.4　情報編集

登録した（もしくはされた）情報は，図 6 でマークが付いてるように，「編集」のボタンを押したら図 7 の編集画面に遷移する．そこで変えたい項目を書き換えて「登録」を押すとデータが保存される．

図 6　うまい棒管理システムのサイトのどれかのページを開いた際に表示される画面で編集のボタンを示している画像

「味」，「価格」，「カロリー」の項目を書き換えたら，まるで囲ってる登録のボタンを押す！！
また，やっぱり編集の必要がなかった場合一覧に戻るの部分を押す．

図 7　うまい棒管理システムのサイトの編集ページを開いた際に表示される画面で登録のボタンを示している画像

1.3.5　削除

図 8 に示している詳細画面にある「削除」リンクをクリックすると，その時開いているページの項目が一覧画面から削除される．

図 8　うまい棒管理システムのサイトのどれかのページを開いた際に表示される画面で削除のボタンを示している画像

2　管理者向け仕様書

2.1　システム概要

本システムは，Node.js 環境上で動作する Web アプリケーションである．Web アプリケーションフレームワークとして Express，テンプレートエンジンとして EJS を採用している．

2.2　動作環境

- Node.js
- Web ブラウザ

2.3　導入手順

本システムをサーバー環境へ導入（インストール）する手順は以下の通りである．

1. **Node.js のインストール**:
   公式サイトより Node.js をダウンロードし，インストールする．
2. **ソースコードの配置**:
   任意のディレクトリを作成し，提出されたソースコード一式（`.js` ファイル、`views` フォルダ、`public`

フォルダ）を配置する．

3. **依存ライブラリのインストール**:
   ターミナルで該当ディレクトリに移動し，以下のコマンドを実行して必要なライブラリ（Express, EJS）をインストールする．

   ```
   $ npm install express ejs
   ```

2.4 運用操作

2.4.1 起動と停止

- **システムの起動**:
  以下のコマンドを実行すると，Web サーバーがポート 8080 で起動する．

  ```
  $ node app5.js
  ```

  起動後，ブラウザで `http://localhost:8080/umaibo` 等へアクセスし，画面が表示されることを確認する．
- **システムの停止**:
  起動中のターミナルで `Ctrl + C` キーを入力することで，サーバープロセスを終了（停止）できる．

2.5 運用上の注意

本システムを運用するにあたり，以下の点に注意する必要がある．

- **データの永続性について**:
  本システムは簡易的な実装のため，登録されたデータ（うまい棒やキャラの情報）はサーバーのメモリ（変数）上にのみ保存される．したがって、**サーバーを再起動（停止）すると，追加・変更したデータはすべて初期状態に戻る**仕様である．重要なデータを扱う場合は，別途データベースの実装が必要となる．
- **ポート番号の競合**:
  本システムはポート 8080 を使用する．同一サーバー内で他のアプリケーションがポート 8080 を使用している場合，起動に失敗する．その際は，ソースコード内のポート番号設定を変更するか競合するプロセスを停止する必要がある．

3 開発者向け仕様書

3.1 ファイル構成

本システムのディレクトリ構成は以下の通りである。

- `app5.js`: サーバーサイドのメインプログラム
- `views/`: 画面表示用テンプレート (EJS ファイル)
- `public/`: 静的ファイル (新規登録用 HTML)

4　開発者向け仕様書

4.1　ファイル構成

本システムのディレクトリ構成は以下の通りである。

- `app5.js`: サーバーサイドのメインプログラム
- `views/`: 画面表示用テンプレート (EJS ファイル)
- `public/`: 静的ファイル (CSS, 新規登録用 HTML)

4.2　システム 1: うまい棒管理システム

4.2.1　データ構造

データは，プログラム内の配列変数 `umaiboData` によって管理される．配列内の各要素はオブジェクトであり，以下のプロパティを持つ．

表 1　うまい棒システムのデータ定義

プロパティ名	データ型	説明
id	数値	データ固有の ID
flavor	文字列	うまい棒の味（例: メンタイ）
price	数値	価格（円）
calorie	数値	カロリー（kcal）

4.2.2　画面遷移とリソース設計

本システムでは，以下の URL パターン（リソース）と HTTP メソッドを用いて機能を実装している．URL 内の `:number` は，配列内の要素番号を示す動的なパラメータである．

- **一覧表示**
  - メソッド: GET
  - URL: `/umaibo`
  - 処理: `umaibo_list.ejs` をレンダリングし，全データを一覧表示する．
- **詳細表示**
  - メソッド: GET
  - URL: `/umaibo/:number`
  - 処理: 指定された要素番号の詳細データを `umaibo_detail.ejs` で表示する．
- **新規登録画面**
  - メソッド: GET
  - URL: `/umaibo/create`
  - 処理: 静的ファイル `/public/umaibo_new.html` へリダイレクトする．
- **新規登録処理**
  - メソッド: POST
  - URL: `/umaibo`
  - 処理: フォームから送信されたデータを配列に追加し，一覧画面を表示する．
- **編集画面**

- メソッド: GET
- URL: `/umaibo/edit/:number`
- 処理: 指定データの編集用フォーム（`umaibo_edit.ejs`）を表示する．

- **更新処理**
  - メソッド: POST
  - URL: `/umaibo/update/:number`
  - 処理: 指定要素番号のデータを更新し，一覧画面へリダイレクトする．
- **削除処理**
  - メソッド: GET
  - URL: `/umaibo/delete/:number`
  - 処理: 指定要素番号のデータを配列から削除し，一覧画面へリダイレクトする．

4.3 システム 2: スマブラキャラ一覧システム

4.3.1 データ構造

データは，プログラム内の配列変数 `smashData` によって管理される．配列内の各要素はオブジェクトであり，以下のプロパティを持つ．

表 2 スマブラシステムのデータ定義

プロパティ名	データ型	説明
id	数値	データ固有の ID
name	文字列	ファイター名（例: マリオ）
series	文字列	出典作品（例: スーパーマリオ）
number	文字列	参戦ナンバー（例: 01）

4.3.2 画面遷移とリソース設計

URL 内の `:index` は，配列内の要素番号を示すパラメータである．

- **一覧表示**
  - メソッド: GET
  - URL: `/smash`
  - 処理: `smash_list.ejs` をレンダリングし，全キャラクターを表示する．
- **詳細表示**
  - メソッド: GET
  - URL: `/smash/:index`
  - 処理: 指定された要素番号の詳細を `smash_detail.ejs` で表示する．
- **新規登録画面**
  - メソッド: GET
  - URL: `/smash/create`
  - 処理: `/public/smash_new.html` へリダイレクトする．
- **新規登録処理**
  - メソッド: POST
  - URL: `/smash`

  - 処理: データを配列に追加し，一覧画面を表示する．
- **編集画面**
  - メソッド: GET
  - URL: `/smash/edit/:index`
  - 処理: 編集用フォーム（`smash_edit.ejs`）を表示する．
- **更新処理**
  - メソッド: POST
  - URL: `/smash/update/:index`
  - 処理: データを更新し，一覧画面へリダイレクトする．
- **削除処理**
  - メソッド: GET
  - URL: `/smash/delete/:index`
  - 処理: データを削除し，一覧画面へリダイレクトする．

4.4 システム 3: スプラトゥーン武器一覧システム

4.4.1 データ構造

データは，プログラム内の配列変数 `splaData` によって管理される．配列内の各要素はオブジェクトであり，以下のプロパティを持つ．

表 3　スプラトゥーンシステムのデータ定義

プロパティ名	データ型	説明
id	数値	データ固有の ID
name	文字列	ブキ名（例: わかばシューター）
sub	文字列	サブウェポン名（例: スプラッシュボム）
special	文字列	スペシャルウェポン名（例: グレートバリア）

4.4.2 画面遷移とリソース設計

URL 内の `:index` は，配列内の要素番号を示すパラメータである．

- **一覧表示**
  - メソッド: GET
  - URL: `/splatoon`
  - 処理: 全ブキデータを `spla_list.ejs` で一覧表示する．
- **詳細表示**
  - メソッド: GET
  - URL: `/splatoon/:index`
  - 処理: 指定されたブキの詳細を `spla_detail.ejs` で表示する．
- **新規登録画面**
  - メソッド: GET
  - URL: `/splatoon/create`
  - 処理: `/public/spla_new.html` へリダイレクトする．
- **新規登録処理**

  - メソッド: POST
  - URL: `/splatoon`
  - 処理: データを配列に追加し，一覧画面を表示する．
- **編集画面**
  - メソッド: GET
  - URL: `/splatoon/edit/:index`
  - 処理: 編集用フォーム（`spla_edit.ejs`）を表示する．
- **更新処理**
  - メソッド: POST
  - URL: `/splatoon/update/:index`
  - 処理: データを更新し，一覧画面へリダイレクトする．
- **削除処理**
  - メソッド: GET
  - URL: `/splatoon/delete/:index`
  - 処理: データを削除し，一覧画面へリダイレクトする．

4.4.3　画面遷移図

本システムの画面遷移（3 システム共通）を図 9 に示す．

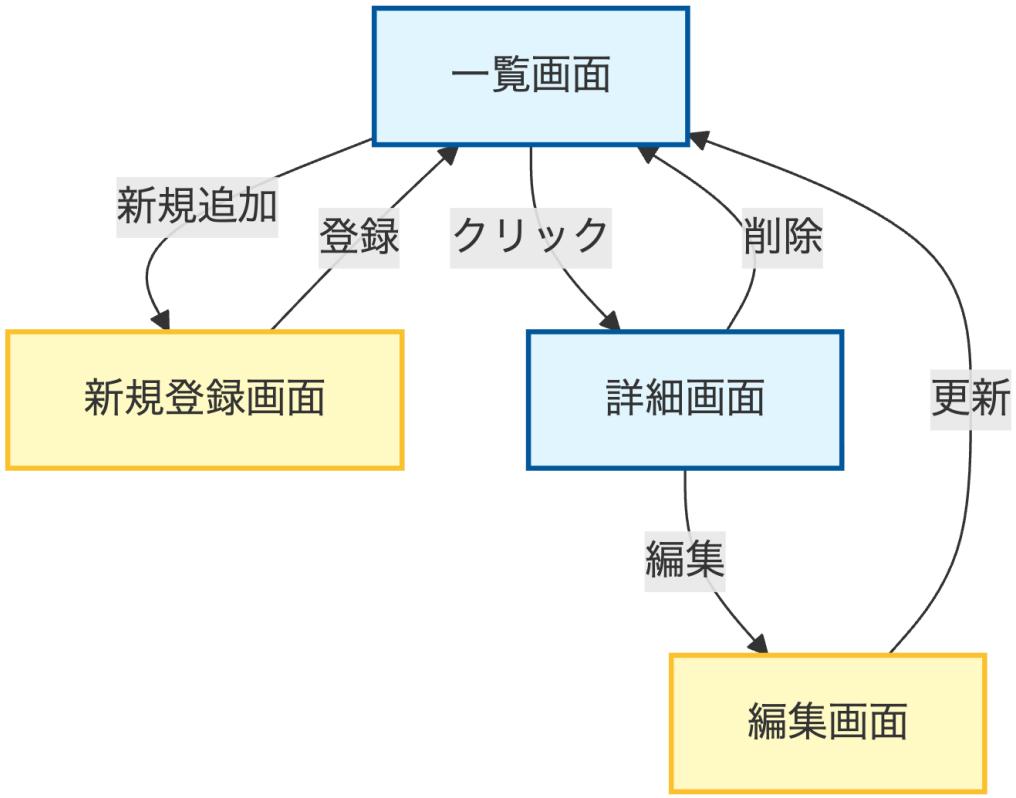


図 9　画面遷移図