Web プログラミング仕様書

25G1137 吉田朝朗

2025 年 12 月 28 日

【ソースコード・リポジトリ】

`https://github.com/tomoro1729-a/webpro_06.git`

1 開発者向け仕様書

本システムは，Node.js および Express フレームワークを用いて構築された，3 つの異なるテーマ（映画・化学元素・ゲーム武器）を扱う Web データベースアプリケーションである．各システムは独立したリソースとして管理されつつも，統一された RESTful な設計指針に基づいて実装されている．

1.1 技術スタックと構成

本開発では、開発効率と保守性を高めるため、以下の技術・ライブラリを採用した。

- **サーバーサイド: Node.js (Express)**
  採用理由: Node.js における標準的な Web サーバー構築ライブラリであり、ルーティング処理や HTTP リクエストの処理を簡潔に記述できるため。
- **テンプレートエンジン: EJS (HTML 動的生成)**
  採用理由: HTML 内にサーバー側のデータを埋め込む処理（動的生成）を効率的に行うため。
- **フロントエンド**: 標準 HTML5, CSS3（静的ファイルとして外部定義）
- **データ管理**: サーバー内メモリによるオブジェクト配列管理

2 共通設計仕様

2.1 ページ構造の決定

各サブシステムは，ユーザーの操作性を考慮し，以下の 4 種類のページ構造を共通で持つ．

1. **一覧ページ (Index)**: 登録データの全件をテーブル形式で表示する．
2. **詳細ページ (Show)**: 特定リソースの全プロパティを閲覧する．
3. **新規登録ページ (New)**: 新規データ入力を受け付けるフォーム．
4. **編集ページ (Edit)**: 既存データの修正を受け付けるフォーム．

2.2 ページ遷移と処理の流れ

本システムはステートレスな HTTP 通信を行うが，操作後のユーザー体験（UX）を最適化するため，以下のリダイレクト戦略を採用している．

- **新規作成/削除後**: データの増減を即座に確認させるため，「一覧ページ」へリダイレクトする．
- **編集完了後**: 修正された内容を再確認させるため，「詳細ページ」へリダイレクトする．

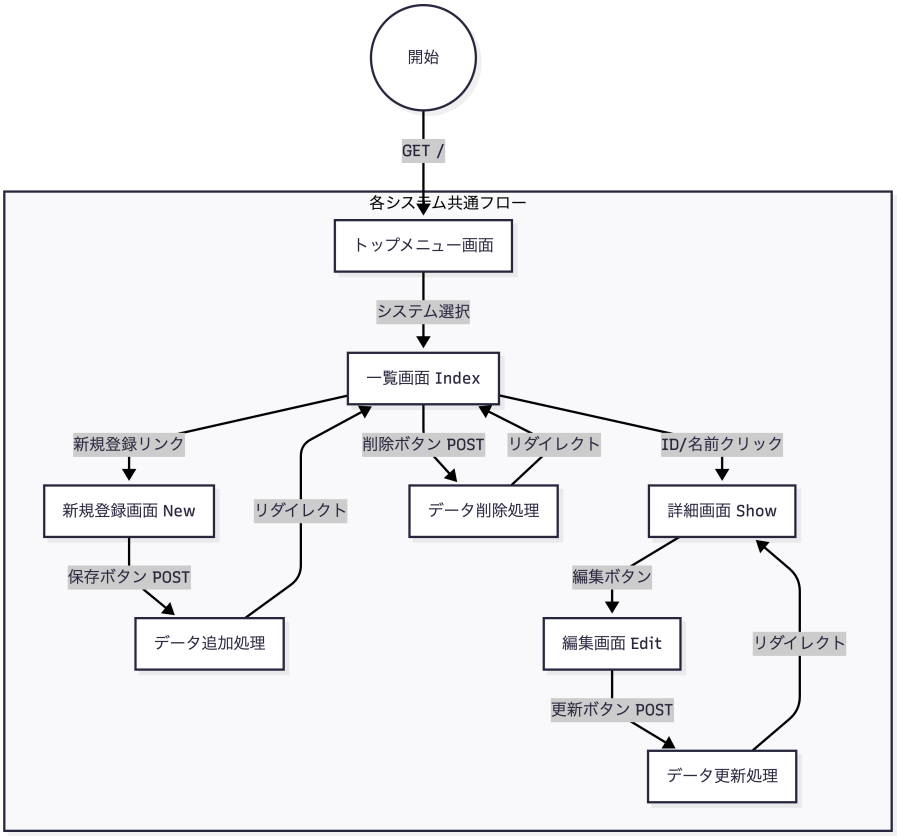

図 1　システムの共通処理フローチャート

3　各サブシステムの詳細設計

本章では，各サブシステムが管理するデータの詳細構造を定義する．なお，各データ構造表における**「必須」**項目の定義は以下の通りである．

- **〇 (必須)**: 新規登録および編集時に，入力フォームにて値の入力が強制される項目（HTML の `required` 属性が付与される）．
- **- (任意)**: 空欄での登録が許容される項目．

3.1　システム 1：ドラえもん映画一覧システム

歴代作品の興行データ等を管理する．

3.1.1　データ構造

プロパティ	型	必須	内容
id	Integer	○	システム一意識別子
title	String	○	映画の題名
year	Integer	○	公開された西暦
revenue	String	-	興行収入記録
director	String	-	映画監督の氏名
explain	String	-	あらすじ・概要説明

3.2　システム 2：周期表元素一覧システム

原子番号を一意識別子（ID）として利用し，科学データを管理する．

3.2.1　データ構造

プロパティ	型	必須	内容
number	Integer	○	原子番号（ID 兼用）
symbol	String	○	元素記号（H, He 等）
name	String	○	日本語名
mass	Float	○	原子量
explain	String	-	性質や発見の経緯等の記述

3.3 システム 3：FE 風花雪月 武器一覧システム

ゲームバランスに関わる複数の数値パラメータを管理する．

3.3.1 データ構造

プロパティ	型	必須	内容
id	Integer	○	武器一意識別子
name	String	○	武器の名称
type	String	○	分類（剣，槍，斧，弓等）
might	Integer	-	攻撃の威力
hit	Integer	-	命中率（%）
critical	Integer	-	必殺率（%）
range	String	○	射程距離（1, 1-2 等）
weight	Integer	-	武器の重さ
rank	String	○	武器レベル（E, D, A 等）
explain	String	-	武器の特性や特殊効果の記述

4 UI/デザイン設計方針

本システムでは，3 つのサブシステム間で操作感と視覚的な統一性を保つため，以下のデザイン方針を採用している．

4.1 スタイルシートの共通化

全ページ共通のスタイル定義を `/public/style.css` に集約し，これを各 EJS テンプレートから読み込む構成とした．これにより，将来的なデザイン変更時に全ページへ即座に反映できる保守性を確保している．

4.2 配色のルール

視認性を高めるため，以下の配色ルールを統一して適用している．

- **ベースカラー**: 白 (#FFFFFF) - 背景色として使用
- **メインカラー**: 黒 (#333333) - 文字色として使用
- **アクセントカラー**: 青系 - リンクおよびボタンに使用し，クリック可能な要素であることを明示

5 実装上の工夫と安全性

- **一貫性の維持**: CSS ファイルを `/public/style.css` に外部化し，全リソース間でデザインの整合性を

確保した．

- **入力制御**: フォームの数値項目には `type="number"` を使用し，ブラウザ側でのバリデーション機能を活用した．
- **削除の安全化**: 削除操作は `POST` メソッドによる送信と確認ダイアログ（JavaScript）を併用し，意図しないデータ消失を防止した．

6 管理者向け仕様書

本章では，本システムの導入（インストール）から，日常的な起動・停止手順，およびトラブルシューティングについて記述する．

6.1 ソースコードの入手と環境構築

6.1.1 動作要件

本システムを稼働させるためには，以下のソフトウェア環境が必要である．

- **OS**: Windows, macOS, Linux いずれか
- **Runtime**: Node.js (LTS 推奨)
- **Manager**: npm (Node.js に標準同梱)

6.1.2 インストール手順

以下の手順に従い，GitHub リポジトリからソースコードを取得し，依存ライブラリをインストールする．

1. **リポジトリのクローン**: ターミナル（コマンドプロンプト）を開き，以下のコマンドを実行してソースコードをダウンロードする．

   Listing 1 Git クローンコマンド

   ```
   git clone https://github.com/tomoro1729-a/webpro_06.git
   cd webpro_06
   ```

2. **依存パッケージのインストール**: プロジェクトフォルダ内で以下のコマンドを実行し，`package.json` に記載されたライブラリ（Express, EJS 等）を一括導入する．

   Listing 2 パッケージインストールコマンド

   ```
   npm install
   ```

6.2 システムの起動と終了

6.2.1 起動方法

環境構築完了後，以下のコマンドで Web サーバーを起動する．

```
node app.js
```

起動に成功すると，コンソールに `Server is running on http://localhost:8080` と表示される．ブラウザで上記 URL へアクセスすることでシステムを利用開始できる．

6.2.2 終了方法

サーバーを停止する場合は，ターミナル上でキーボードの **Ctrl** + **C** を入力する．これによりプロセスが安全に終了する．

6.3 トラブルシューティング

起動時または操作時に問題が発生した場合の対処法を以下に示す．

表 1　よくあるエラーと対処法

現象・エラーメッセージ	推定される原因	対処法
`command not found: node`	Node.js がインストールされていない，またはパスが通っていない．	公式サイトより Node.js のインストーラをダウンロードし，セットアップを行う．
`Error: Cannot find module...`	依存ライブラリ（`node_modules`）が不足している．	`npm install` コマンドを再度実行し，完了を待つ．
`EADDRINUSE :::8080`	ポート 8080 番が既に使用されている．	既に起動している別の `node` プロセスを終了するか，PC を再起動する．

6.4　システムの制約事項と既知の課題

本システムは学習用プロトタイプであり，運用上以下の制約が存在する．

- **データの揮発性**: データベースを使用せず，サーバーのメモリ上（変数）でデータを管理している．そのため，**サーバーを再起動・停止すると，追加・編集したデータは初期状態にリセットされる**．恒久的な運用を行う場合は，別途データベース（SQLite, MySQL 等）の実装が必要である．
- **同時編集の競合**: 排他制御を行っていないため，複数の管理者が同一データを同時に編集した場合，最後に保存操作を行った内容で上書きされる仕様である．

7　利用者向け仕様書

本章では，一般利用者を対象に，本データベースシステムの基本的な操作手順について解説する．本システムは「ドラえもん映画」「元素周期表」「武器データ」の 3 つのサブシステムから構成されているが，すべての画面で共通の操作方法を採用している．ここでは例として，システムごとの共通手順を示す．

7.1　システムへのアクセス

7.1.1　トップメニュー

ブラウザでシステムにアクセスすると，最初にメニュー画面が表示される．利用したいデータベースの「システムを開く」ボタンをクリックすることで，各管理画面へ移動する．

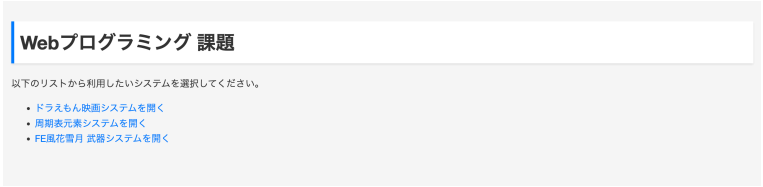

図 2　システム選択メニュー画面

7.2　データの閲覧

7.2.1　一覧画面（リスト表示）

各システムのトップ画面では，登録されている全データが表形式で一覧表示される．

- **詳細リンク**: データの ID や名前をクリックすると，そのデータの詳細情報画面へ移動する．
- **削除ボタン**: データの右側にある削除ボタンを押すと，その行のデータが削除される（詳細は後述）．

ID	公開年	タイトル	操作
1	1980年	のび太の恐竜	削除
2	1981年	のび太の宇宙開拓史	削除
3	1982年	のび太の大魔境	削除
4	1983年	のび太の海底鬼岩城	削除
5	1984年	のび太の魔界大冒険	削除
6	1985年	のび太の宇宙小戦争	削除
7	1986年	のび太と鉄人兵団	削除
8	1987年	のび太と竜の騎士	削除
9	1988年	のび太のパラレル西遊記	削除
10	1989年	のび太の日本誕生	削除

図 3　データ一覧表示画面の例

7.2.2　詳細画面

一覧画面から選択したデータの全項目（説明文などの長いテキスト含む）を確認できる．この画面には「編集」ボタンがあり，登録内容の修正を行うことができる．

図 4　詳細情報確認画面

7.3　データの管理（追加・編集・削除）

7.3.1　新規データの追加

一覧画面にある「新規登録」リンクをクリックすると，登録フォームが表示される．必要な項目（名前，数値など）を入力し，「登録」ボタンを押すことで新しいデータがデータベースに追加される．

図 5　データ入力フォーム

7.3.2　既存データの編集

詳細画面にある「編集」ボタンをクリックすると，現在のデータが入力された状態でフォームが開く．内容を修正して「更新」ボタンを押すと，変更内容が反映される．

7.3.3　データの削除

一覧画面の各行にある「削除」ボタンをクリックする．**注意**: 削除操作を行うとデータは即座に消去され，元に戻すことはできないため，慎重に操作を行うこと．