Web プログラミング データ表示の Web アプリ開発

25G1036 河井希天

2025 年 12 月 28 日

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

1　開発者向け

1.1　ニンテンドーシステム

1.1.1　概要

本システムは，任天堂の歴代ハードウェアおよび周辺機器のデータを管理・閲覧するための Web アプリケーションである．ユーザーは Web ブラウザを通じて，製品情報の一覧表示・詳細閲覧・新規登録・編集・削除を行うことができる． 開発およびデモ運用を想定し，データベースサーバーを使用せず，アプリケーションサーバーのメモリ上でデータを管理する構成となっている．

1.1.2　データ構造

本データ構造は，任天堂のハードウェア情報を管理するためのオブジェクト配列として実装されている．各オブジェクトは，一意の識別子である id，製品分類を示す kind，および製品名の name を主要なキーとして構成される．加えて，付帯情報である発売日（day），価格（price），プレイ人数（people）については，数値計算を前提としない文字列型として定義することで，単位表記や日付フォーマットの記述における柔軟性を確保している．

表 1　ニンテンドーシステム データ構造

キー名	型	説明
id	Number	通し番号
kind	String	種別
name	String	製品名
day	String	発売日
price	String	価格
people	String	プレイ人数

1.1.3　ニンテンドーシステム HTTP メソッドとリソース名一覧

本システムでは，任天堂のハードウェア情報を閲覧する一般画面と，情報をメンテナンスするための管理画面を区別して定義している．登録，編集，更新といった各プロセスに対して専用のリソースパスを割り当てることで，ハードウェアデータの正確な CRUD 操作を実現している．

表 2　ニンテンドーシステム HTTP メソッドとリソース名一覧

機能	メソッド	リソース名	説明
一覧表示	GET	/nin	一覧表示画面を表示
新規作成画面表示	GET	/nin/create	新規登録用の入力フォームを表示
詳細表示	GET	/nin/:number	指定された ID のデータの詳細画面を表示
削除	GET	/nin/delete/:number	指定された ID のデータを削除
新規登録処理	POST	/nin	フォームから送信されたデータを受け取り，新規データを登録
編集画面表示	GET	/nin/edit/:number	指定された ID のデータを編集するためのフォームを表示
更新処理	POST	/nin/update/:number	編集フォームから送信されたデータを受け取り，指定された ID の情報を更新

1.1.4　遷移図

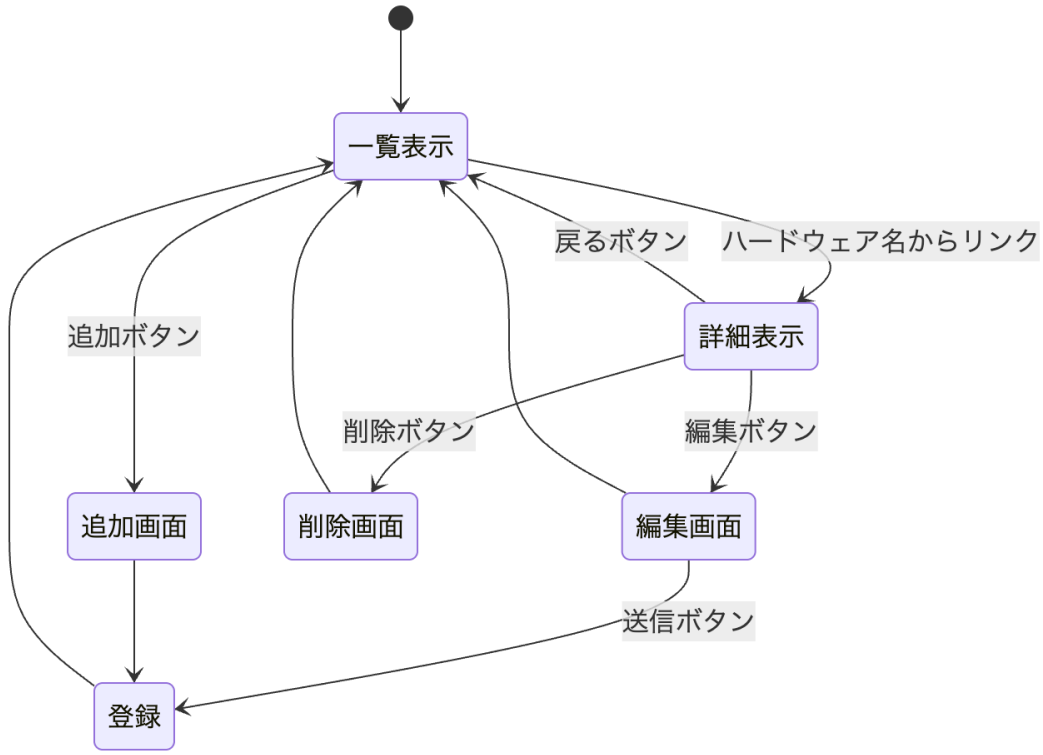

図 1　ニンテンドーシステムの遷移図

1.1.5　リソース名ごとの機能詳細

- GET /nin
  サーバーメモリ上の nintendou 配列から全データを取得し，nin.ejs に渡してレンダリングを行う
- GET /nin/create
  ユーザーが新しいハードウェア情報を入力するための nin.html を返却する．
- POST /nin
  フォームから送信された入力値を受け取る．システム側で新たな id を自動生成し，入力値と合わせてオブジェクト化する．これを nintendou 配列の末尾に追加（push）した後，一覧画面へリダイレクトする．
- GET /nin/:number
  URL パラメータ:number を配列のインデックスとして使用し，特定の一件を抽出する．抽出したデータ nin_detail.ejs に渡して表示する．
- GET /nin/edit/:number
  指定されたインデックス（:number）のデータを配列から取得し，その値を初期値として埋め込んだ状態で nin_edit.ejs を表示する．
- POST /nin/update/:number
  編集フォームから送信されたデータを用いて，指定されたインデックス（:number）の配列要素を直接上書き更新する．更新完了後は一覧画面へリダイレクトする．
- GET /nin/delete/:number
  指定されたインデックス（:number）の要素を配列から物理的に削除（splice）する．処理完了後は一覧画面へリダイレクトする．

1.2　iPhone システム

1.2.1　概要

本システムは，Apple の歴代 iPhone のデータを管理・閲覧するための Web アプリケーションである．ユーザーは Web ブラウザを通じて，製品情報の一覧表示・詳細閲覧・新規登録・編集・削除を行うことができる． 開発およびデモ運用を想定し，データベースサーバーを使用せず，アプリケーションサーバーのメモリ上でデータを管理する構成となっている．

1.2.2　データ構造

本構造は，モバイル端末の基本スペックと市場リリース情報を管理するオブジェクト構造を持つ．id，name を識別子とし，市場投入に関わる day や price を文字列で保持する．特徴的なのは size 項目であり，デバイスの物理的なディスプレイサイズをインチ単位のテキストデータとして定義している．複雑な階層構造を持たず，各世代の端末情報をフラットなオブジェクト配列として格納する，一覧性と拡張性に優れたシンプルなデータ定義を採用している．

表 3　iPhone システム データ構造

キー名	型	説明
id	Number	通し番号
name	String	端末名
day	String	発売日
price	String	価格
size	String	サイズ

1.2.3　iPhone システム HTTP メソッドとリソース名一覧

本システムでは，Apple の iPhone 情報を閲覧する一般画面と，情報をメンテナンスするための管理画面を区別して定義している．登録，編集，更新といった各プロセスに対して専用のリソースパスを割り当てることで，ハードウェアデータの正確な CRUD 操作を実現している．

表 4　iPhone システム HTTP メソッドとリソース名一覧

機能	メソッド	リソース名	説明
一覧表示	GET	/app	一覧表示画面を表示
新規作成画面表示	GET	/app/create	新規登録用の入力フォームを表示
詳細表示	GET	/app/:number	指定された ID のデータの詳細画面を表示
削除	GET	/app/delete/:number	指定された ID のデータを削除
新規登録処理	POST	/app	フォームから送信されたデータを受け取り，新規データを登録
編集画面表示	GET	/app/edit/:number	指定された ID のデータを編集するためのフォームを表示
更新処理	POST	/app/update/:number	編集フォームから送信されたデータを受け取り，指定された ID の情報を更新

1.2.4　遷移図

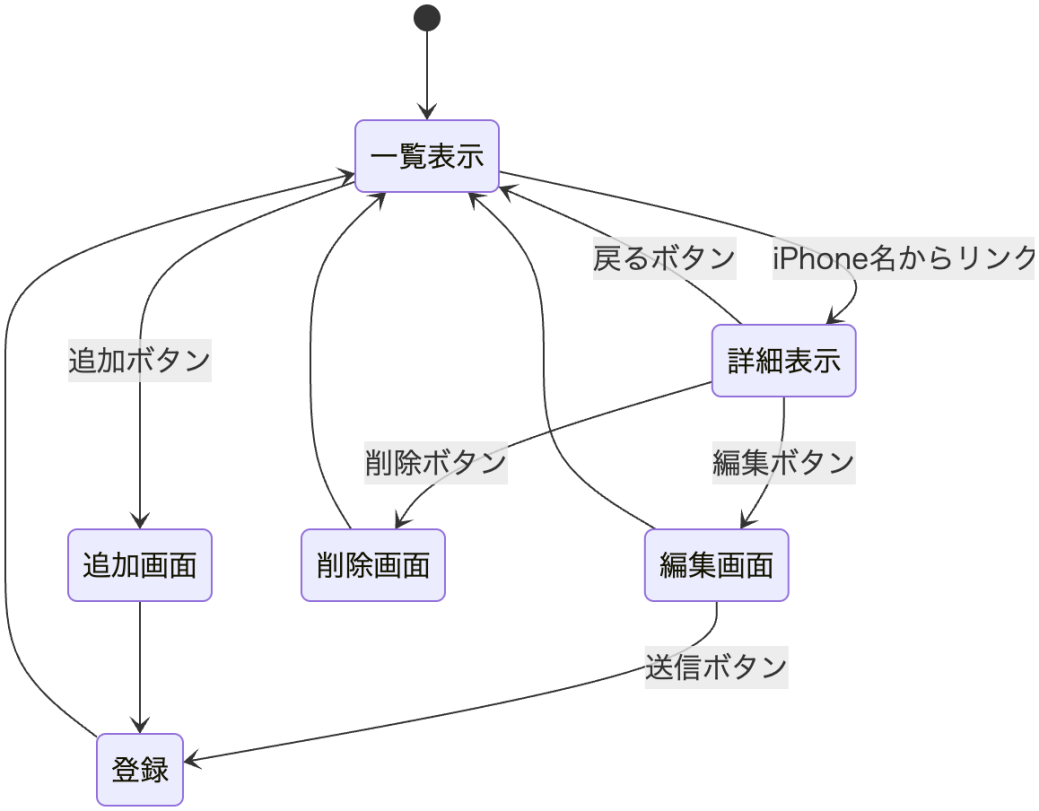

図 2　iPhone システムの遷移図

1.2.5　リソース名ごとの機能詳細

- GET /app
  サーバーメモリ上の apple 配列から全データを取得し，app.ejs に渡してレンダリングを行う
- GET /app/create
  ユーザーが新しいハードウェア情報を入力するための app.html を返却する．
- POST /app
  フォームから送信された入力値を受け取る．システム側で新たな id を自動生成し，入力値と合わせてオブジェクト化する．これを apple 配列の末尾に追加（push）した後，一覧画面へリダイレクトする．
- GET /app/:number
  URL パラメータ:number を配列のインデックスとして使用し，特定の一件を抽出する．抽出したデータ app_detail.ejs に渡して表示する．

- GET /app/edit/:number
  指定されたインデックス（:number）のデータを配列から取得し，その値を初期値として埋め込んだ状態で app_edit.ejs を表示する．
- POST /app/update/:number
  編集フォームから送信されたデータを用いて，指定されたインデックス（:number）の配列要素を直接上書き更新する．更新完了後は一覧画面へリダイレクトする．
- GET /app/delete/:number
  指定されたインデックス（:number）の要素を配列から物理的に削除（splice）する．処理完了後は一覧画面へリダイレクトする．

1.3 ポケモンシステム

1.3.1 概要

本システムは，ニンテンドーのポケモンのデータを管理・閲覧するための Web アプリケーションである．ユーザーは Web ブラウザを通じて，製品情報の一覧表示・詳細閲覧・新規登録・編集・削除を行うことができる． 開発およびデモ運用を想定し，データベースサーバーを使用せず，アプリケーションサーバーのメモリ上でデータを管理する構成となっている．

1.3.2 データ構造

本構造は，ポケモンの生態データと個体情報を管理するオブジェクト構造を持つ．id，name を基本キーとし，分類に関わる type や class，region を文字列で保持する．最大の特徴は character 項目であり，複雑な階層構造は持たず，複数の特性情報をカンマ区切りのテキストとして格納するシンプルかつ可読性の高いフラットなデータ定義を採用している．

表 5　ポケモンシステム データ構造

キー名	型	説明
id	Number	通し番号
name	String	ポケモンの名称
type	String	タイプ
class	String	分類
character	String	特性
region	String	生息地

1.3.3 ポケモンシステム HTTP メソッドとリソース名一覧

本システムでは，ニンテンドーのポケモン情報を閲覧する一般画面と，情報をメンテナンスするための管理画面を区別して定義している．登録，編集，更新といった各プロセスに対して専用のリソー

スパスを割り当てることで，ハードウェアデータの正確な CRUD 操作を実現している．

表 6　ポケモンシステム HTTP メソッドとリソース名一覧

機能	メソッド	リソース名	説明
一覧表示	GET	/po	一覧表示画面を表示
新規作成画面表示	GET	/po/create	新規登録用の入力フォームを表示
詳細表示	GET	/po/:number	指定された ID のデータの詳細画面を表示
削除	GET	/po/delete/:number	指定された ID のデータを削除
新規登録処理	POST	/po	フォームから送信されたデータを受け取り，新規データを登録
編集画面表示	GET	/po/edit/:number	指定された ID のデータを編集するためのフォームを表示
更新処理	POST	/po/update/:number	編集フォームから送信されたデータを受け取り，指定された ID の情報を更新

1.3.4　遷移図

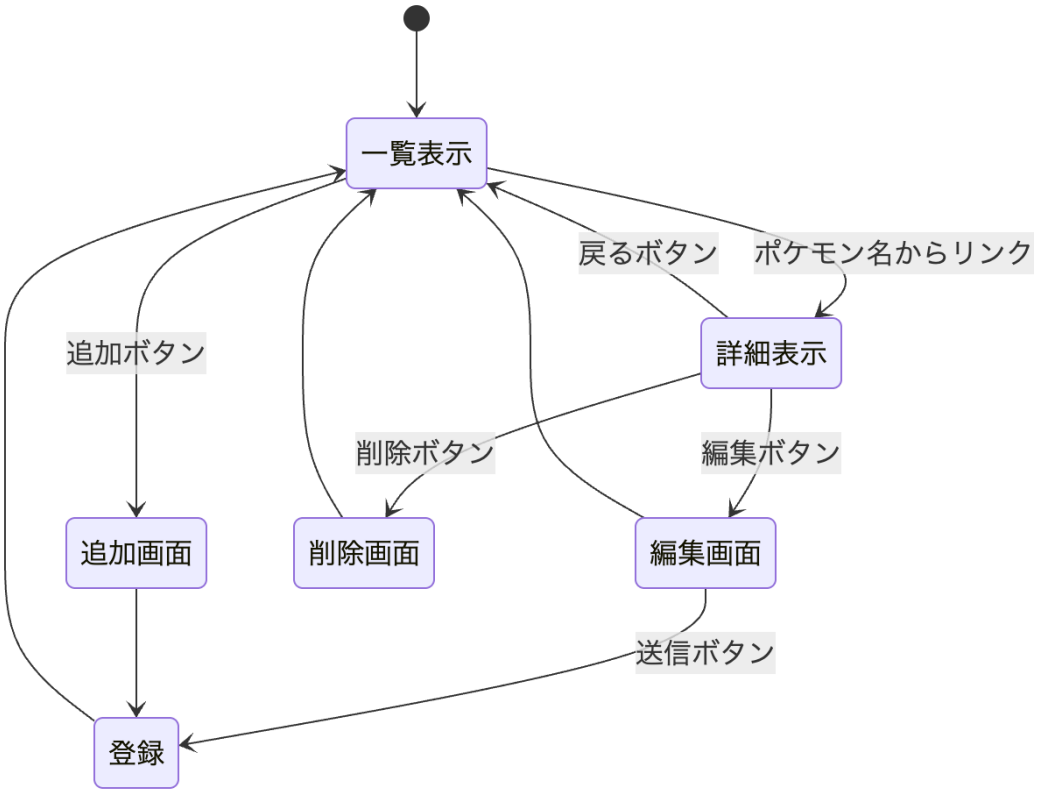

図 3　ポケモンシステムの遷移図

1.3.5　リソース名ごとの機能詳細

- GET /po
  サーバーメモリ上の apple 配列から全データを取得し，app.ejs に渡してレンダリングを行う
- GET /po/create

ユーザーが新しいハードウェア情報を入力するための app.html を返却する．

- POST /po
  フォームから送信された入力値を受け取る．システム側で新たな id を自動生成し，入力値と合わせてオブジェクト化する．これを apple 配列の末尾に追加（push）した後，一覧画面へリダイレクトする．
- GET /po/:number
  URL パラメータ:number を配列のインデックスとして使用し，特定の一件を抽出する．抽出したデータ app_detail.ejs に渡して表示する．
- GET /po/edit/:number
  指定されたインデックス（:number）のデータを配列から取得し，その値を初期値として埋め込んだ状態で app_edit.ejs を表示する．
- POST /po/update/:number
  編集フォームから送信されたデータを用いて，指定されたインデックス（:number）の配列要素を直接上書き更新する．更新完了後は一覧画面へリダイレクトする．
- GET /po/delete/:number
  指定されたインデックス（:number）の要素を配列から物理的に削除（splice）する．処理完了後は一覧画面へリダイレクトする．

2　管理者向け仕様書

2.1　インストール方法

本プログラムを動作させるための環境構築手順を以下に記す．本手順ではパッケージ管理システムとして **Homebrew** を使用する．ターミナルを起動し，以下のコマンドを実行して Homebrew をインストールする．

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

パスワードの入力が求められるので自身の MacOS のパスワードを入力し，プロンプトが表示されるまで待つ．次に以下の二つのコマンドを順にを実行する．

```
( echo; echo 'eval "$(/opt/homebrew/bin/brew shellenv)"') >> ~/.zprofile
eval "$(/opt/homebrew/bin/brew shellenv)"
```

node.js はバージョン更新が早いので，使用するバージョンを管理するツールとして nodebrew をインストールする．以下の四つのコマンドを順に実行する．

```
brew install nodebrew
nodebrew setup
```

```
    echo 'export PATH=$HOME/.nodebrew/current/bin:$PATH' >> ~/.zshrc
    source ~/.zshrc
```

つぎに node.js の最新版をインストールする．以下の二つのコマンドを順に実行する．

```
    nodebrew install stable
    nodebrew ls
```

ここで最新バージョンの番号を確認し，以下のコマンドを最新バージョンに設定する．ここでは番号を v24.1.0 と仮定する．以下の二つのコマンドを実行する．

```
    nodebrew use v24.3.0
    npm install -g npm
```

2.2 起動方法

本システムを起動する手順について説明する．まず初めに，ターミナルを起動し，本システムのプロジェクトディレクトリに移動する．次に，以下のコマンドを実行し本システムを起動する．

```
    node repo.js
```

ターミナルに `Server is running on http://localhost:8080/repo` と表示されたら起動できている状態である．

2.3 終了方法

本システムを終了する手順について説明する．ターミナルで本システムが起動している状態で，`Ctrl + C` キーを同時に押すことで終了できる．

2.4 起動できない場合

本システムが起動しない場合は，以下の手順で確認と再試行を行う．まず，ターミナルで適切なディレクトリに移動しているか，およびシステムが使用するポート番号が他のアプリケーションと競合していないかを確認し，再度起動を試みる．次に，実行環境のバージョンを確認する．Node.js や npm が古いことが原因で不具合が生じる場合があるため，これらを最新バージョンにアップデートした上で，改めて起動を試行する．以上の手順でも解決しない場合は，GitHub からリポジトリを再度クローンし，正規の手順に従ってインストールと起動を一からやり直す．

2.5 分かっている不具合

本システムには，更新データが永続化されないという課題がある．具体的には，情報の追加や編集を完了した状態であっても，システムを再起動するとそれらの更新情報が失われ，操作前の状態に戻ってしまう問題が報告されている．