G-35

ウェブアプリケーション開発におけるフレームワーク連携のためのモデル設計手法
Model Design Method for Framework Integration in Web Application Development

貞平 怜雄†　　満田 成紀†
Reo Sadahira　　Naruki Mitsuda

1. はじめに

近年のウェブアプリケーションはサーバー側とクライアント側の２つのプログラムが協調して動作している。また、開発効率を上げるためにウェブアプリケーションフレームワークを利用して開発することが頻繁である。しかし、ウェブアプリケーションフレームワークには、固有の設計方針が定められているため、サーバーサイドとクライアントサイドで異なるフレームワークを利用すると、設計方針同士が干渉してしまう可能性がある[1]。

そこで、フレームワークを組み合わせて効率よく開発するためのモデル設計手法を提案する。

2. ウェブアプリケーション開発におけるフレームワーク利用

ウェブアプリケーションはサーバーサイドとクライアントサイドのプログラムが協調して動作している。

サーバーサイドはユーザからは見えない部分を指しており、ウェブサーバー上で実行される。ビジネスロジックの処理、データベースとのやり取り、認証と権限管理、セキュリティ対策（例えば、データの暗号化、CSRF 対策、XSS 対策）などを行う。

一方、クライアントサイドはユーザが直接触れる部分を指しており、ウェブブラウザ上でプログラムが実行される。HTML はウェブページの構造を定義するマークアップ言語であり、CSS は HTML 要素の見た目やレイアウトを設定する。JavaScript はブラウザ上で動的な動作を実装する。

2.1 ウェブアプリケーションフレームワーク

ウェブアプリケーションフレームワークとは、ウェブアプリケーションの開発を効率化し、コードの再利用性や保守性を向上させるためのソフトウェアフレームワークである。ウェブアプリケーションの一般的な機能（ユーザ認証、データベース操作、セッション管理、テンプレートエンジン、URL ルーティングなど）を提供し、開発者が繰り返し行う作業を簡素化することができる。

ウェブアプリケーションフレームワークは、大きく分けてサーバーサイドフレームワークとクライアントサイドフレームワークの二種類に分類される。

2.2 サーバーサイドフレームワーク

サーバーサイドフレームワークはサーバー上で動作し、主にウェブアプリケーションのバックエンドロジックを管理する。主な機能は、データベース操作、認証、ビジネスロジックの処理、HTTP リクエストとレスポンスの処理などがある。主なプログラミング言語は PHP、Python、Java などがあり、これらを利用したフレームワークの代表例は Laravel、Ruby on Rails、Django などが挙げられる。

2.3 クライアントサイドフレームワーク

クライアントサイドフレームワークはユーザのブラウザ上で動作し、主にウェブアプリケーションのフロントエンドを管理する。主な機能は、ユーザーインターフェースの構築、クライアント側のルーティング、データの取得と表示、リアルタイムな更新などがある。代表例として、Vue.js、React.js、Angular.js が挙げられる。

2.4 サーバーサイドとクライアントサイドのフレームワークの連携

サーバーサイドとクライアントサイドのフレームワークを連携させることで効果的なウェブアプリケーションを開発できる。連携によるパフォーマンス面、ユーザエクスペリエンス面、開発効率面での利点を挙げる。

まずはパフォーマンス面の向上について述べる。サーバーサイドではデータの提供や認証などの重要な機能を行い、クライアントサイドではデータのキャッシングやレンダリングを行う。このように役割を分担することで、サーバーサイドへのリクエスト数を減らし、サーバーへの負荷を軽減することができる。

次に、開発効率の向上について述べる。サーバーサイドとクライアントサイドの開発を分離することで、開発者は各サイドの機能実装に集中できるため効率的に開発できる。また、分離によりクライアントサイドはコンポーネント指向の開発を促進し、ソースコードの再利用性が高まり開発効率や保守性が向上する。

最後にユーザエクスペリエンスの向上について述べる。リアルタイムでデータ更新することによりユーザは常に最新の情報を取得することができる。クライアントサイドでインタラクティブなユーザーインターフェースを実装し、サーバーサイドでのデータの処理やビジネスロジックと連携することで、ユーザにとって直感的でレスポンシブな体験を提供する。

3. フレームワーク連携のためのモデル設計手法

ウェブアプリケーションフレームワークには、固有の設計方針が定められているため、サーバーサイドとクライアントサイドで異なるフレームワークを利用し連携させる場合には、それぞれの設計方針に合致するモデルを設計しなければならい。

アプリケーションの外部仕様からフレームワーク連携を想定したモデルを設計するには、サーバーサイドとクライアントサイドの境界を適切に定める必要がある。そのため、まずはウェブアプリケーションの参照モデルとして広く利

†和歌山大学，Wakayama University

用されている MVC（Model-View-Controller）モデル[2, 3]を構成し、Model をサーバーサイド、View をクライアントサイドに振り分ける。

ひとつのフレームワークを使ってアプリケーションを実装する場合には、フレームワークの設計方針に合わせて Controller をいずれかのサイドに振り分けることになるが、サーバーサイドとクライアントサイドで異なるフレームワークを利用するためには、Controller の機能を細分化し、それぞれのサイドに振り分ける必要がある。

このようにして振り分けたモデルに対して、さらに、それぞれのフレームワーク固有の設計方針に合致する形でモデルを詳細化する。

4. タスク管理アプリケーションを用いた手法の評価

手法の評価用にサーバーサイドに Laravel[4]、クライアントサイドに Vue.js[5]を用いて、タスク管理アプリケーションの開発を行った。

4.1 タスク管理アプリケーションの仕様

機能として新しいタスクの追加、タスクの一覧表示（図 1）、既存タスクの編集・更新（図 2）、既存タスクの削除からなる。

全体の流れとしては、まずクライアントサイドはデータの取得や送信を行うために、Ajax（axios）を使用して HTTP リクエストを行い、サーバーサイドで提供する RESTful API にアクセスする。そして、サーバーサイドでは、クライアントサイドからのリクエストを受け取り、データベース操作などを行った後、クライアントサイドにレスポンスを返す。最後に、クライアントサイドはレスポンスデータを受け取り、データの表示などの処理を行う。

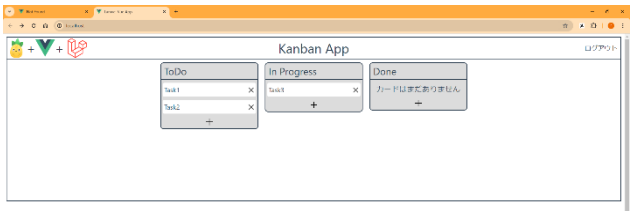

図 1　タスク表示画面

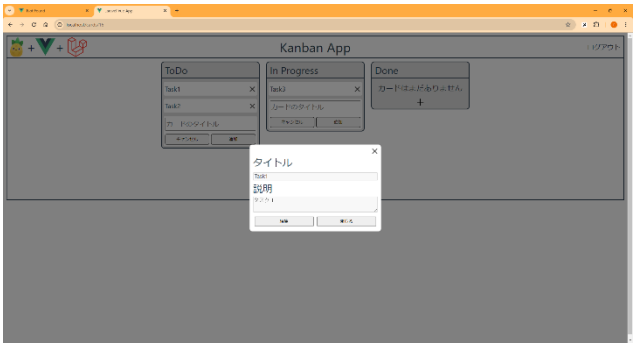

図 2　既存タスクの編集・更新

4.2 利用するフレームワーク

サーバーサイドのフレームワークとして広く利用されている Laravel を採用し、クライアントサイドは利用実績が多く、かつ、Laravel にもサポート機能がある View.js を採用した。ただし、Laravel のサポート機能は利用せず、それぞれは独立したフレームワークとみなせる形で利用している。

4.2.1 Laravel

Laravel は PHP ベースのサーバーサイドフレームワークであり、MVC モデル（Model-View- Controller）を採用しており、ビジネスロジックを担当する Model、ユーザーインターフェースを担当する View、Model と View の制御を担当する Controller に分割される。Laravel では Eloquent ORM（オブジェクト関係マッピング）と Migration（マイグレーション）が備わっている。Eloquent ORM は、データベースと Model に関係性を持たせることでデータの操作を容易にする。Migration は、データベースのスキーマの変更内容（作成や編集、削除）を記述したファイルを作成することで、データベースのバージョン管理を安全かつ容易に行うことができる。

4.2.2 Vue. js

Vue.js は JavaScript ベースのクライアントサイドフレームワークであり、Vue.js は MVVM モデル（Model-View-ViewModel）を採用している。ViewModel が Model と View の状態を監視することで、データの変更を検知すると関連する部分のみが自動的に更新されるため、高いパフォーマンスを実現する。これによりシングルページアプリケーションの開発に向いている。シングルページアプリケーションは初回のみ Web サーバーから Web ページを取得し、ページ遷移時は新しいページを Web サーバーから取得せずに、クライアントサイドで JavaScript を使用して必要な部分のみを更新する。これにより、Web サーバーとの通信コストを抑えて高速に画面を切り替えることができる。

また、コンポーネントベースのアーキテクチャにより、UI を小さな再利用可能なコンポーネントに分割でき、コードの再利用性と保守性が向上する。

4.3 モデル設計の実施結果

アプリケーションに対してサーバーサイドとクライアントサイドの境界を引くために、アプリケーションの構造を把握できる MVC モデルを作成した（図 3）。タスク一覧画面、タスク追加画面は View、タスクカード（カードにタスク名をかいたもの）、タスクリスト（タスクカードをまとめたもの）は Model、読込、追加、更新、削除のようなユーザからのデータ操作のリクエストに対しての処理は Controller に分類される。DB は Model と同じになる。

ただシンプルな MVC モデルであると、Model・View・Controller すべてをサーバーサイド、Model・Controller はサーバーサイド、View のみクライアントサイド、Model・Controller の一部はサーバーサイド、View・Controller の一部はクライアントサイドなどといったように様々なパターンで捉えることができる。そこで、サーバーサイドとクライアントサイドの境界を明確にするための MVC モデルを次に作成した（図 4）。このモデルでは、クライアントサイド

フレームワークを使う前提であるので、View はクライアントサイドに位置する。そして、Model はサーバーサイドに位置する。しかし、Controller は読込・追加・更新・削除それぞれがサーバーサイドとクライアントサイドに分かれて機能する。そして、分かれた Controller の中間に API の処理が入ってくる。API の処理とは、クライアントサイドでは API エンドポイントに HTTP リクエストを送ること、サーバーサイドでは HTTP リクエストを受け取り、適切な Controller の処理にデータを渡すことを指す。

最後に、図 4 のモデルに対して、フレームワークの提供するモジュールを利用するように具体化したモデルが図 5 である。サーバーサイドでは Laravel の設計方針に合わせて Controller や Model の設計が具体化されている。一方、クライアントサイドでは、画面設計に合わせて View。js が提供するコンポーネントを組み合わせるように設計が具体化されている。

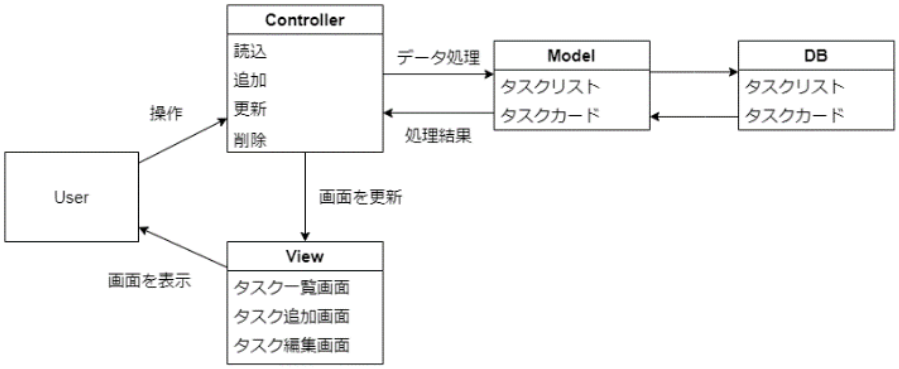

図 3　MVC モデル

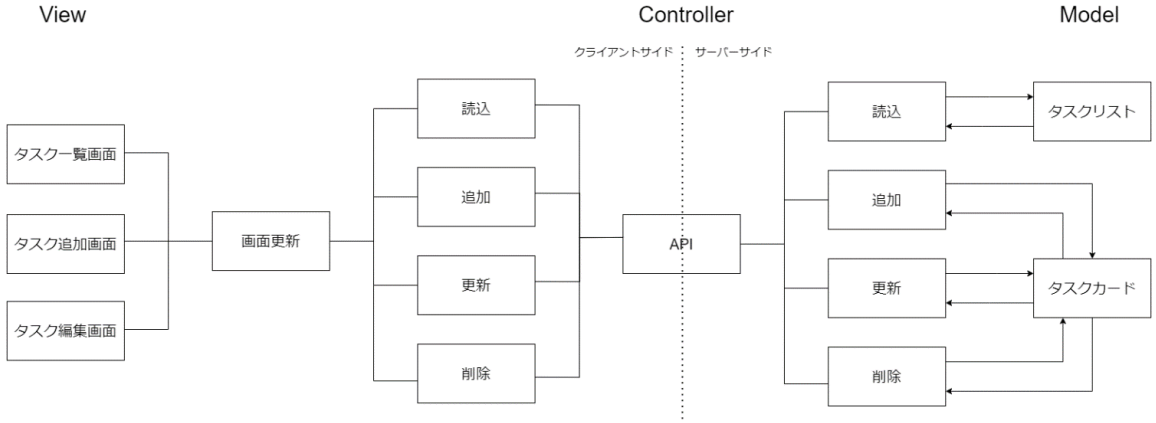

図 4　サーバーサイド・クライアントサイドに分けた MVC モデル

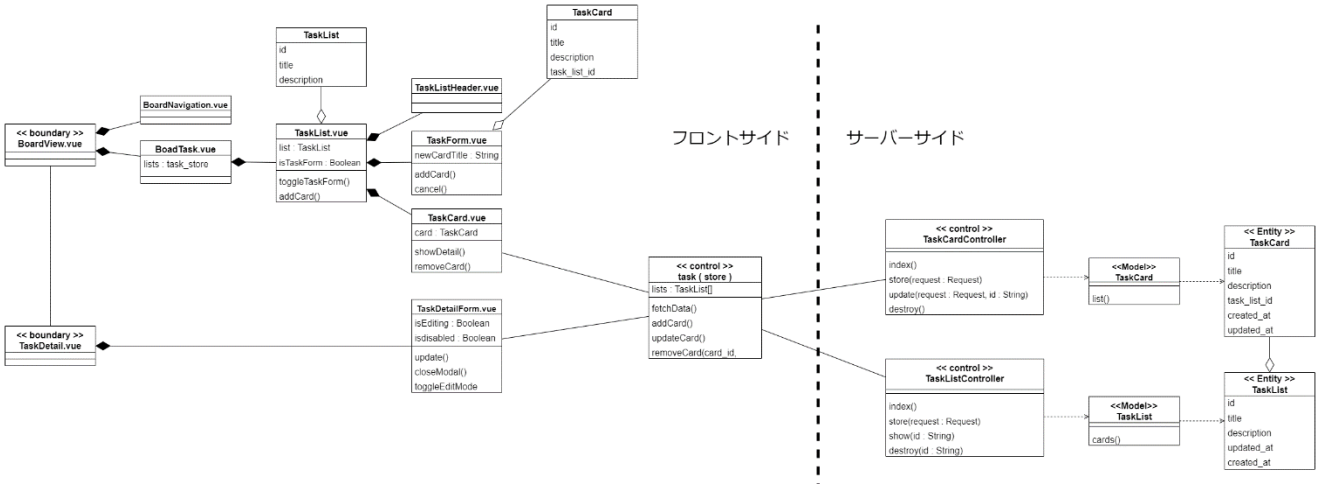

図 5　フレームワーク連携したクラス図

機能
・タスクを登録する
概要
・ユーザがタスクを作成する際に利用される
アクタ
・ユーザ
事前条件
・ユーザはタスク一覧画面を表示していること
基本フロー
1。システムは、タスク一覧画面を表示する
2。ユーザは、プラスボタン（＋）を押す
3。システムは、入力フォームを表示する
4。ユーザは、新しいタスクを入力して、追加ボタンを押す
5。システムは、新しいタスクをデータベースに保存する
6。システムは、タスク一覧画面を更新する。

図 6　ユースケースシナリオの例

5. 考察

まずはクライアントサイドについて考察する。クライアントサイドフレームワークはコンポーネント指向で設計されていることが多い[6, 7]。そこで、コンポーネント指向に沿った画面設計図を描くべきだと考える。画面設計図を描くことで画面に登場するコンポーネント、つまりアプリケーションのボタンやフォームなどの部品をあらかじめ設定しておく。それにより、以後のモデル作成にもスムーズに適用することができる。

コンポーネントをもとに、ユースケースシナリオを作成する（図 6）。シナリオを先に作成することで、シーケンス図を作成するための機能の流れや Model や Controller などのサーバーサイドの構造を把握しやすくなる。

次に、ユースケースシナリオをもとに、ロバストネス図を描き、コンポーネントなどを Boundary、Control、Entity に分類する。そして、ロバストネス図から MVC モデルへと発展させる。また、ユースケースシナリオをもとに、シーケンス図を作成することで、より詳細な処理の流れとサーバーサイドとクライアントサイドの境界を定めることができる。

結果として、アプリケーションを適切にサーバーサイドとクライアントサイドに分ける手順として、以下のように考えている。

(1) 画面設計図を作成し、コンポーネントやオブジェクトを設定する
(2) 設定したコンポーネントやオブジェクトをもとに各機能のユースケースシナリオを作成する
(3) ユースケースシナリオをもとにロバストネス図を作成し、オブジェクトを Boundary、Control、Entity に分類する
(4) 各機能のシーケンス図を作成する
(5) アプリケーション全体のクラス図を作成する

6. おわりに

本研究は、ウェブアプリケーション開発において、サーバーサイドとクライアントサイドで異なるフレームワークを利用し連携させるためのモデル設計手法を提案するものである。

タスク管理アプリケーションを題材として設計手法の実施評価を行い、より適切にサーバーサイドとクライアントサイドに振り分ける手順について考察を行った。今後の課題として、考察した結果にもとづいて手法の改善を行い、開発効率が向上することを確認することが考えらえる。

参考文献

[1] 満田成紀，福安直樹，ウェブアプリケーションフレームワーク利用に潜む課題～実プロジェクトデータを題材に～，コンピュータソフトウェア, 27 巻, 3 号, 2010, p.3_2-3_12
[2] 大木幹雄，ボトムアップな MVC アーキテクチャモデルの概念的な構成方法に関する考察,電子情報通信学会技術研究報告，100，440，2000，p.57-63
[3] 結縁祥治，加藤敬史，加藤大樹，阿草清滋，Web オートマトン: MVC モデルに基づく Web アプリケーションの動作モデル，Information and Media Technologies，2006，p.66-79
[4] 竹澤有貴，栗生和明，新原雅司，大村創太郎，PHP フレームワーク Laravel Web アプリケーション開発バージョン 8.x 対応，ソシム株式会社，2021
[5] 川口和也，喜多啓介，野田陽平，手島拓也，片山真也，Vue.js 入門 基礎から実践アプリケーション開発まで，技術評論社，2018
[6] 荻野慶，小野康一，深澤良彰，コンポーネント指向 Web アプリケーションフレームワークにおけるモジュール性の向上のための一手法，電子情報通信学会技術研究報告，2005，p.13-18
[7] 水野友貴．松本啓之亮，森直樹,モデル駆動ソフトウェア開発へのコンポーネントベース技術の適用,電気学会論文誌C，133，12，2013，p.2275-2281