データベースおよび演習 最終レポート（2022年7月22日版）

k20127 和田 健[†1,a)]

概要：本レポートは，データベースおよび演習の講義における最終レポートである．最終課題として作成したプログラムについて設計仕様をまとめる．

Final report (version 2022/07/22)

Wada Takeru[†1,a)]

1. 機能概要

本レポートで説明するプログラムは, PHP,MySQL, PHP フレームワークである Laravel を使用して作成を行なった，ToDo 管理アプリケーションのプログラムである．アプリケーション使用者は，最初にログイン画面に遷移するが，会員情報が無いため，新規登録を行う．名前，メールアドレス，パスワードを登録すると，ログインした状態で ToDo のホーム画面に遷移がされる．

ToDo ホーム画面では，タスクの追加，閲覧，編集，削除を行うことが可能である．必須項目であるタスク名と期限を入力し，任意であるタスク詳細を入力してから登録を押すと，データベースへタスクが登録され，一覧へと表示される仕組みになっている．また，このタスク一覧はユーザーによって切り替えられるため，他のユーザーには見られない．編集ボタンを押すと，そのタスクの編集を行うことが可能で，編集画面から保存を押すと，変更が保存され，内容が更新される．また，削除ボタンを押すことで，タスクが完了した際などに削除することが可能である．

[†1] 現在，愛知工業大学
Presently with Aichi Institute of Technology
[a)] wdktru@pluslab.org

2. 利用技術

2.1 PHP

PHP，正式名称”PHP:HypertextPreprocessor”は，PersonalHomePage がその由来で，Web で利用される HTML 形式のようなハイパーテキストを閲覧者の操作によって生成し，動的な画面を作ることに向いている．また PHP は無償で利用が可能でマニュアルも完備しているため，安心して利用が可能である．

PHP は Web サーバー上で動作するサーバーサイド・スクリプト言語と呼ばれており，ブラウザから送信されたリクエストに応えて処理を実行し，結果を HTML 形式に整形してブラウザへ送信をする．また，PHP の設定ファイルに記述するだけで，入力されたデータの文字コード変換や，表示するときの文字コード変換を自動的に行う．

PHP5 からオブジェクト指向が強化され，SPL(クラスライブラリ) が利用可能になり，PHP5.3 から名前空間の導入でクラス名や定数名の競合を回避できるようになり，PHP5.4 では，トレイトというコードを再利用するための仕組みが導入された．[1]

2.2 MySQL/MariaDB

MySQL は，最も普及している SQL データベース管理システムで，オラクル社により開発，流通，およびサポート

が行われている．MySQL データベースは，リレーショナルデータベースと呼ばれる独立したテーブルにデータを保存するデータベースである．この構造は速度を最適化した物理ファイルに分かれており，データベース，テーブル，ビュー，行，およびカラムなどのオブジェクトを持つ論理モデルは，柔軟なプログラミング環境を提供する．MySQL の SQL の部分は”StructuredQueryLanguage”(構造化クエリー言語) を表している．SQL はデータベースにアクセスするために使用される最も一般的な言語で，環境によって直接入力，別の言語で作成されたコードに SQL ステートメントを埋め込む，または SQL 構文を隠す言語固有の API を使用する場合がある．また，MySQL ソフトウェアはオープンソースで，DatabaseServer は非常に高速で信頼性が高く，処理を要することなく快適に実行することができる．[2]

MariaDB とは，上記の MySQL から派生した，オープンソースリレーショナルデータベースシステムであり，GPLv2 ライセンスのもとで利用が可能である．MariaDB community によって開発がされ, MontyProgramAb がその中心幹事となっている．MariaDB は大部分の点で MySQL と同様の動作をする．コマンド，インタフェース，ライブラリ，API など，MySQL に存在するものは MariaDB にも存在する．[3]

この二つの相違点は，MySQL は Oracle が所有し，MariaDB は完全にオープンソースである点や，多くの場合に MariaDB は MySQL より高速である点が挙げられる．

2.3 Laravel

Laravel は，Symphony という，規模の大きい Web アプリケーション開発に特化した PHP フレームワークが基になっており，2011 年にリリースされた比較的新しいフレームワークながら，PHP におけるフレームワークでは世界的に普及がされている．

MVC モデルという，処理を Model，View，Controller の 3 つに分別し，機能ごとに開発を進めるモデルが採用されており，処理の内容とそれを描く場所が明確になっており，開発経験が浅くても学びやすく扱いやすい開発が行える．また，ユーザー管理などの基本機能はインストール時に揃っているため，簡単な開発にも向いている．[4]

2.4 Composer

Composer は，PHP での依存関係管理のためのツールで，プロジェクトが依存するライブラリを宣言することができ，それらを管理 (インストール，更新) することができる．本レポートでは，Laravel をインストールするために用いている．[5]

2.5 BootStrap

BootStrap とは，Web アプリケーションや Web サイトを作成するフロントエンド Web アプリケーションフレームワークで，HTML および CSS ベースのデザインテンプレートとして用意されている．また，レスポンシブ Web デザインに対応しているため，PC, スマートフォン，タブレットなどデバイスの画面サイズに最適なレイアウトデザインに切り替えることができる．[6]

3. システム設計

3.1 システム概要

初めにこの Web アプリケーションに接続すると，ログイン画面が表示される．このログイン画面では，既に会員登録をしている場合に，登録したメールアドレスとパスワードを入力する事でログインができる．最初は会員登録がされていないので，新規登録を行う．ログインがされていない場合，画面右上に”新規登録”というボタンがあり，そこから新規登録画面へ遷移する事ができる．新規登録画面では，名前，メールアドレス，パスワードを設定することで新規登録をする事ができる．新規登録を行う事ができると，自動的に登録したアカウントでログインがされ，ToDo アプリケーションのホーム画面へと遷移する．ログインが完了していると，右上の”ログイン”だったボタンは”ホーム”に，”新規登録”だったボタンは登録した名前が表示されたボタンに変わる．

ホーム画面では，タスクの追加が行える．画面上部にタスク名，期限，タスク詳細を入力するテキストエリアがある．タスク名は，todocontents という table の要素$title に，期限は$deadline に，タスク詳細は$content へと格納される．また，この時ログインしているユーザー情報を読み取り，ユーザー id を todocontentstable の要素$user_id に格納することで，タスク一覧に表示するタスクを，ログインしているユーザーが登録したタスクのみを表示するように処理を行なっている．タスク名と期限の設定は必須のため，入力せずに登録を行うと”このフィールドを入力してください”というエラーメッセージとともに，足りない入力箇所が表示される．ここでは HTML 側からバリデーションがされている，かつ，php の Controller，つまりバックエンド側からもバリデーションがされている．

また，ページ下部のタスク一覧表示部分からは，登録したタスクの編集と削除を行うことが出来る．編集ボタンを押すと編集画面に遷移し，各項目のテキストエリアには，既に登録した情報が初期状態で入力されている．ここでもタスク名と期限は必須項目であるため，入力がなかった場合にはエラーメッセージが表示される．削除ボタンを押すとタスクの削除をする事ができ，一覧画面からも消えるようになっている．

タスクの登録が完了すると，登録ボタンの下にメッセー

ジとして”タスクの登録が完了しました”と表示が出る．これは，登録が問題なく行えた場合に，ホーム画面を表示する際に message というセッション変数にメッセージを文字列として代入し，ホーム画面側では，セッション変数に message が入っていた場合にメッセージを表示するようになっている．

3.2 画面遷移

会員登録画面，新規登録画面，ホーム画面などの遷移図は以下の図 1 のようになっている．最初にログイン画面が

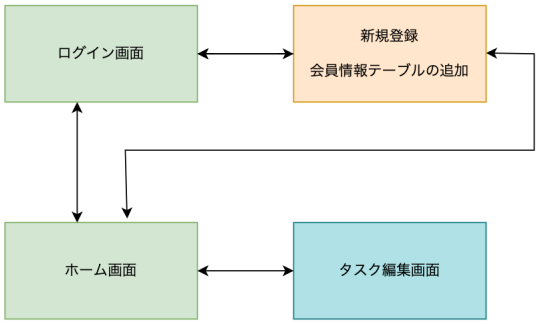

図 1 画面遷移図

表示され，ユーザー登録がされている場合にはログインをすると ToDo トップページへ遷移し，新規登録をする場合には，画面右上の新規登録ボタンを押すと新規登録画面へ遷移する．

新規登録画面からは，ログイン画面とホーム画面に遷移する事ができる．右上のログインボタンを押すとログイン画面へ遷移し，登録情報を入力してから新規登録ボタンを押すと，新規登録とログインが同時に完了しホーム画面へと遷移する．

ホーム画面からは，ログイン画面とタスク編集画面へ遷移する事ができる．ログイン画面へは，右上の自分のユーザー名のボタンをクリックし，ログアウトを選択することで遷移する．タスク編集画面へは，各タスクの編集ボタンを押すことで遷移する

タスク編集画面からは，ホーム画面へ遷移する事ができる．適切に情報を入力し保存ボタンを押すか，右上のホームボタンを押すことでホーム画面へ遷移する．

3.3 データベース設計

3.3.1 ToDo テーブル

```
1  <?php
2  use Illuminate\Database\Migrations\Migration;
3  use Illuminate\Database\Schema\Blueprint;
4  use Illuminate\Support\Facades\Schema;
5
6  class CreateUsersTable extends Migration{
7  /**
8   * Run the migrations.
9   *
10  * @return void
11  */
12 public function up(){
13         Schema::create('users', function (
               Blueprint $table) {
14         $table->bigIncrements('id');
15         $table->string('name');
16         $table->string('email')->unique();
17         $table->timestamp('email_verified_at
               ')->nullable();
18         $table->string('password');
19         $table->rememberToken();
20         $table->timestamps();
21 });
22 }
23 /**
24  * Reverse the migrations.
25  *
26  * @return void
27  */
28 public function down(){
29         Schema::dropIfExists('users');
30 }
31 }
```

上記プログラムは，user テーブルを作成するマイグレーションファイルである．php artisan make:migration ”ファイルネーム”をコマンドラインで入力する事でマイグレーションファイルは作成ができ，pubric function up() 内に作りたいテーブルの変数の定義を記入する．

14 行目では，ユーザーの id を bigIncrements で定義している．これは，自動で採番がされる UNSIGNED BIGINT 型のカラムであることを意味している．

16 行目では，ユーザーのメールアドレスを string で，かつ unique で定義している．これは，メールアドレスの値が一意であることを指定している．

19 行目の rememberToken では，現在のログイン持続認証トークンを格納することを目的とした，NULL 許容の VARCHAR(100) のカラムを作成している．

20 行目の timestamps では，データの作成日，アップデート日が TIMESTAMP カラムで作成される．

3.3.2 ToDo テーブル

```
1  <?php
2  use Illuminate\Database\Migrations\Migration;
3  use Illuminate\Database\Schema\Blueprint;
4  use Illuminate\Support\Facades\Schema;
5
6  class CreateToDoContentsTable extends
       Migration{
```

```
7  /**
8   * Run the migrations.
9   *
10  * @return void
11  */
12 public function up(){
13         Schema::create('todocontents',
               function (Blueprint $table) {
14         $table->bigIncrements('id');
15         $table->bigInteger('user_id');
16         $table->datetime('deadline');
17         $table->string('title',50);
18         $table->string('content')->nullable
               ();
19         $table->timestamps();
20         });
21 }
22 /**
23  * Reverse the migrations.
24  *
25  * @return void
26  */
27 public function down(){
28         Schema::dropIfExists('ToDoContents');
29 }
30 }
```

上記プログラムは，ToDo テーブルを作成するマイグレーションファイルである．id は登録したタスクの id が設定され，user_id にはタスクを登録したユーザーの id が格納されるため，タスクの一覧表示をする際に，現在ログイン状態のユーザーの id と同じ user_id が登録されているタスクのみを表示することで，それぞれのユーザーに自身の登録したタスクのみを表示させることが実現できている．

3.4 システム詳細

本プログラムでは MVC モデルでシステムの設計をしており，つまり Model,View,Controller に php ファイルを分けている．

Model では，データベースへのアクセス，登録，更新や削除などであったり，データの検索や加工など，データに関する処理全般を行う．

View では，Controller から取得したデータを PHP 内の HTML 要素で表示したり，CSS で見た目を管理したりする．

Controller では，Model からデータベースの情報を受け取って View に流したり，View から受け取った情報を Model に流したりなどをする．

本プログラムで使用している MVC の各ファイル名は，以下の表 1 のようになっている．

User.php と Todocontent.php の説明として”テーブル接続用”と記述したのは，本プログラムでは Model ファイル内でのデータ格納は行なっておらず，Controller 内での Eloquent モデルでデータベーステーブルへのデータ格納，更新, 削除等を行なっているため, Model ファイルではテーブルへの接続しか実質的には行なっていないためである．

表 1　各ファイル名

ファイル名	MVC	説明
User.php	Model	users テーブル接続用
Todocontent.php	Model	todocontents テーブル接続用
login.blade.php	View	ログイン画面表示用
register.blade.php	View	新規登録画面表示用
todocontents.blade.php	View	ホーム画面表示用
todocontentsedit.blade.php	View	タスク編集画面表示用
LoginController.php	Controller	ログイン処理用
RegisterController.php	Controller	新規登録処理用
TodocontentsController.php	Controller	タスク追加，編集，削除処理用

3.4.1 Controller

TodocontentsController 内の処理について説明をする．

```
1  public function index(){
2      $todocontents = Todocontent::where('
           user_id',Auth::user()->id)
3      ->orderBy('deadline','asc')
4      ->paginate(7);
5      return view('todocontents',[
6          'todocontents' => $todocontents
7      ]);
8      }
```

上図はホーム画面を表示する際にの View を呼び出す処理である．$todocontents には，todocontents テーブル内のメンバー user_id と，現在ログインしているユーザーの id が一致しているタスクのみを格納している．3 行目の orderBy では，表示するタスクの並び替えを行なっている．上図の場合，メンバー deadline を，asc，つまり昇順で表示するよう指定している．4 行目の pagenate では，ページ内に表示するデータの最大数を指定が出来る．上図の場合，最大で 7 つのタスクを表示し，以降は 2 ページ目に表示する．return では，view の指定した変数へ渡すデータを格納する処理を行なっている．

```
1  public function store(Request $request){
2  // リクエスト確認
3  // dd($request);
4  // バリデーション
5  $validator = Validator::make($request->all()
       ,[
6      'title' => 'required|max:50',
7      'content' => 'max:255',
8      'deadline' => 'required'
9  ]);
10 // バリデーションエラー
11 if($validator->fails()){
12     return redirect('/')
13     ->withInput()
```

```
14      ->withErrors($validator);
15  }
16  //モデル（登録処理）Eloquent
17  $todocontents = new Todocontent;
18  $todocontents ->user_id = Auth::user() -> id
        ;
19  $todocontents -> title = $request -> title;
20  $todocontents -> deadline = $request ->
        deadline;
21  $todocontents -> content = $request ->
        content;
22  $todocontents -> save();
23  return redirect('/')->with('messageタスクの
        登録が完了しました','');
24  }
```

上図は, todocontents.blade.php でタスクの追加を行なった際に，各項目をデータベースのテーブルに実際に追加する処理を行なっている箇所である．$request には追加するタスクの情報が入っており，それらを$validator にコピーすることで，入力された情報に不備がないかを，5 行目から 8 行目で調べている．この時，情報に不備があった場合には，11 行目の if 文に引っかかり，ホーム画面へとリダイレクトがされる．リダイレクト先のホーム画面には，情報に不備のあった箇所が withErrors で渡されるため，どこに不備があったかの情報がホーム画面に表示される仕組みになっている．

17 行目から，本来 Model 内で行う登録処理が，Eloquent モデルで記述されている．18 行目では Todocontent を新しく作りそれぞれのメンバーに$request で取得した情報を格納している．23 行目には return でリダイレクトする URL を渡し，with を使って session にメッセージを渡している．ホーム画面では，このメッセージを受け取って表示する処理がされているため，引数として渡すだけで表示をすることができる．

3.4.2 View

View は Controller から呼び出され，Controller から受け取ったデータを各所に埋め込み，それを表示する．

```
1  @extends('layouts.app')
2  @section('content')
3  <div class="card-body">
4  <div class="card-title">
5  <h2> タスク追加</h2>
6  </div>
7  <!-- バリデーションエラーの表示に使用-->
8  @include('common.errors')
9  <form action="{{ url('todocontents') }}"
       method="POST" class="form-horizontal">
10 @csrf
11 |<!-- 省略>
12 <input type="text" name="title" class="form-
       control" required>
13 |<!-- 省略>
14 @if(session('message'))
15 <div class="alert alert-success">
16     {{session('message')}}
17 </div>
18 @endif
19 |<!-- 省略>
20 @if(count($todocontents) > 0)
21 <div class="card-body">
22 |<!-- 省略>
23 @foreach($todocontents as $task)
24         <tr><!-- タスクタイトル-->
25         <td class="table-text">
26     <div><h1>{{ $task -> title}}</h1></div>
27     <div>{{ $task -> content}}【期限】
28         {{$task -> deadline}}</div>
29         |<!-- 省略>
30 @endforeach
31 <form action="{{ url('todocontentsedit/'.
       $task->id) }}" method="POST">
32 @csrf
33 <button type="submit" class="btn btn-primary
       "> 編集
       </button>
34 </form>
35 |<!-- 省略>
36 <div class="col-md-4 offset-md-4">
37 {{$todocontents->links()}}
38 </div>
```

上図は todocontents.blade.php の一部である．

8 行目では，Controller で作成した validator の値がここに入り，表示がされる

9 行目では，POST で情報を送信することを定義し，form を送信した時に情報を送る URL を指定している．

12 行目では，テキストフィールドを用意している．末尾にある”required”は，必須項目であることを意味し，入力がない状態で form を送信すると，フロントエンドでエラーメッセージを出力するため送信されること自体を防ぐ設計が可能になる．

14 行目から 17 行目では，タスクの登録を行なった際に session に格納されたメッセージを表示する処理を行なっている．

20 行目では，これ以降の処理が，table に入っているデータが空でない時，厳密には自分の登録したタスクが存在する時に行われる処理であることを条件式として設定している．

23 行目では，foreach で table に入っている自分の追加したタスクを取り出し$task に入れ，26 行目から 28 行目で，$task に入ったデータの中からそれぞれメンバーを取り出して表示している．

31 行目から 34 行目では，編集ボタンを押した際の処

理がされている．form の情報を POST で送信する相手の URL は，編集ボタンのある task の id を一部に用いている．todocontentsedit/は編集画面共通の URL で，以降はタスクの固有の id によって変わる．

37 行目の$todocontents-¿links() は，TodocontentsController 内の paginate(7); に対応しており，7 件目以降の表示されなかったタスクを表示しているページのリンクをレンダーする事ができる．

3.4.3 ルーティング

HTTP リクエストに対するルーティング設定を行なっているのは，routes フォルダ内の web.php である．

Listing 1

web.php

```
1  <?php
2  use App\Todocontent;
3  use App\Todocontentedit;
4  use Illuminate\Http\Request;
5
6  URL::forceScheme('http');
7
8  // 以下は全て Controller
9  // ホーム画面
10 Route::get('/','TodocontentsController@index
       ');
11 // 登録処理
12 Route::post('/todocontents','
       TodocontentsController@store');
13 // 更新画面
14 Route::post('/todocontentsedit/{task}','
       TodocontentsController@edit');
15 // 更新処理
16 Route::post('/todocontents/update','
       TodocontentsController@update');
17 // 更新画面の GET
18 Route::get('/todocontentsedit/{id}','
       TodocontentsController@update');
19 // コンテンツを削除
20 Route::delete('/todocontent/{task}','
       TodocontentsController@destroy');
21 // ログインなど Auth
22 Auth::routes();
23 Route::get('/home', '
       TodocontentsController@index')->name('
       home');
```

6 行目では，生成される URL を http で接続させている．

10 行目以降では，それぞれページを表示させるだけの箇所には get で path を渡し，データの送信を行う箇所には post で path を渡している．

3.4.4 テストデータの作成

テストデータの作成には，Seeder と Faker を用いている．Seeder はデータベースにテストデータを挿入する機能で，Faker はそのデータを自動生成するライブラリである．

Listing 2

web.php

```
1  public function run(){
2  $faker = Faker\Factory::create('ja_JP');
3  for($i = 0;$i < 10;$i++){
4  App\Todocontent::create([
5      'title' => $faker->word(),
6      'user_id' => $faker->numberBetween(1,2),
7      'deadline' => $faker->dateTime('now'),
8      'content' => $faker->word(),
9      ]);
10 }
11 }
```

2 行目では create の中に ja_JP を設定しているため，日本語に対応したテストデータが作られる．

5 行目では$faker の word() では，ランダムな単語が生成される．

6 行目では$faker の numberBetween では，第一引数から第二引数までの間の整数をランダムに生成される．

7 行目では$faker の dateTime('now') では，現在の時刻までの日付からランダムに dateTime が生成される．

4. 実装

4.1 実装環境

プロセッサ:1.1GHz クアッドコア IntelCore i5

メモリ:8GB 3733MHz LPDDR4X

4.2 環境設定

OS:macOS BigSur バージョン 11.6.5

XAMPP:OS X 7.4.28

PHP:7.4.30

Laravel:6.20.44

Composer:version 2.3.10

4.3 動作検証

README.md に記述していた手順，”cd k20127_DBlastissue/cms”で cms に移動し，”php artisan serve –port=8080”でサーバを起動して動作検証を行った．既に登録しているメールアドレスで新規登録を行うと，図 2 のようにエラー表示が出るようになっていることを確認できた．また，正常に新規登録が行えた場合には，問題なくログインが可能であることを確認できた．

また，タスク追加の際に必須項目を入力しなかった場合にも，フロントエンドでも，バックエンドでもバリデーションの表示が行えることを確認できた．そのため，間違えて入力せず登録ボタンを押しても入力した情報が消えることなくエラー表示が出せていた．また，編集画面の際に

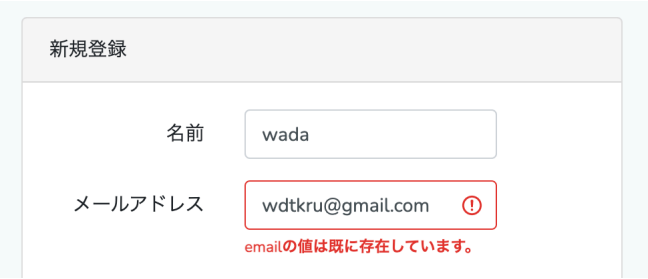

図 2 メールアドレスでのエラー

も送信する前にエラー表示を行えていて，タスクの保持している値を編集の inputfield の初期値に渡せていることも確認できた．

開発を AWS の Cloud9 で行っていたため，Cloud9 で作成したプログラムを，local に移行して動かす際の.env の設定や，Cloud9 での動作検証と同じように動作検証が行えるようにするための環境構築などにとても手こずったが，無事 0 からデータベースの作成，ユーザー，テーブルの作成，サーバーの起動，Web アプリケーションの動作確認までを行うことができた．そして，それを README に書くことができた．

5. まとめ

今回初めて Laravel を用いてプログラムを 1 から作ったことで，MVC モデルの相互関係や，Laravel での開発方法などを知ることができた．サンプルプログラムではあまり MVC モデルがどのように関連しているのかを理解しきれていなかったため，とても理解を深めることができた．

また，中間レポートの際には自分で考えて実装をすることが無かったため，理解はしたけど自分で作れるものなのか，という不安があった．しかし実際にプログラムを作ってみて，初めてでも Laravel は作りやすく，比較的簡単に実装を行えた．

参考文献

[1] PHP とはなんでしょう? - Manual,2022 年 7 月 20 日閲覧, https://www.php.net/manual/ja/intro-whatis.php
[2] 1.3.1 MySQL とは - MySQL 5.6 リファレンスマニュアル,2022 年 7 月 20 日閲覧, https://dev.mysql.com/doc/refman/5.6/ja/what-is-mysql.html
[3] General Quetions - MariaDB Knowledge Base,2022 年 7 月 20 日閲覧, https://mariadb.com/kb/en/general-questions/
[4] Installation - Laravel - The PHP Framework For Web Artisans,2022 年 7 月 20 日閲覧, https://laravel.com/docs/9.x
[5] Introduction - Composer,2022 年 7 月 20 日閲覧, https://getcomposer.org/doc/00-intro.md
[6] Bootstrap · The most popular HTML, CSS, and JS library in the world.,2022 年 7 月 20 日閲覧, https://getbootstrap.com/