データベース及び演習 期末レポート

K20089 西宮　銀河[,†1,a)]

概要：本稿は, データベースおよび演習の授業における最終課題についてのレポートである. 今回はかんたん備忘録という共有可能な TODO リストを作成した. 機能の詳細は以下より述べる.

K20089 Nishimiya Ginga[,†1,a)]

1. 機能概要

かんたん備忘録は他の人とノートを共有することができる TODO リストアプリケーションである. 機能としてはシンプルに備忘録の投稿, 投稿のプライベート化, 自分が投稿した備忘録のチェックを行うことができる. 投稿フォームはスクロールに追従するサイドバーにあるのでどの位置からでも投稿することができる. 投稿フォームのチェックボックスにマークをつけると投稿がプライベート化され, 最新の備忘録からは確認できなくなるが, マイ備忘録からはいつでも確認することができるので見られたくない投稿はチェックをつけて非表示にすることができる.

そして新規登録の際はユーザネームと重複防止のためのメールアドレス, そしてパスワードを設定することができる. また, これらの項目は必ず入力する必要があり, パスワードは 8 文字以上, メールアドレスはメールアドレスの形式のみというバリデーションの制限がある.

登録情報はログイン後サイドバーのリンクから変更することができる. 入力フォームにはユーザデータが取り込まれて表示される.

2. 利用技術

2.1 PHP

PHP とは”PHP:Hypertext Preprocessor”の略で, 直接 HTML に埋め込んで使用することができる汎用プログラミング言語である. 構文は C 言語, Java, Perl から派生しており, それに加えて PHP 独自の構文を組み込むことで PHP 特有の機能を使用できる. PHP はサーバサイド言語であるので, 主に Web サーバ側で動作する. なのでクライアント側からブラウザが HTTP リクエストを送ったとすると, サーバ側が PHP を処理して生成された HTML 文書などをクライアント側に返すという仕組みとなっている. そして PHP はクロスプラットフォーム言語なので Linux, Windows, MacOS はもちろん, NginX や Apache など全ての web サーバ, さらに Microsoft Azure や Amazon AWS などのクラウド環境にも対応している.

また現在の最新バージョンは PHP 8 であり, これは Zend Engine 4 という PHP の実行エンジンの最新バージョンを使用している. これによってオブジェクト指向プログラミングの機能が使えるようになっている.

2.2 MariaDB/MySQL

まず, MySQL は 1995 年に開発されたオープンソースソフトウェアのリレーショナルデータベースである. ’95 年以降は所有者も管理者も転々と変わっていたが 2010 年に所有権が Oracle 社に移るが, 所有権が渡った今でもオープンソースであることは変わっていない.

MySQL はリレーショナルデータベースと呼ばれるデータの保存方法が使われている. リレーショナルデータベースとは 1 つのテーブルに全ての情報を挿入するのではなく, 複数のテーブルに分けてそれぞれを関連づけて管理する方法である. 例えば名前や住所などの顧客データと購入製品や価格などの注文データを扱うテーブルを作るとすると, もし一つのテーブルに全てのデータを入れておくとなった際にデータの重複やデータ削除時の混乱が起こり, 検索が難しくなる. その問題に対処するためにリレーショナルデータベースではキーと呼ばれるユニークな ID を利用して異なるテーブル同士を繋げ, データの取り出しや削除などが容易になっている.

†1 現在, 愛知工業大学
Presently with AICHI INSTITUTE OF TECHNOLOGY University
a) gingin.sol@pluslab.org

そして, MariaDB はこの MySQL をベースに作られたデータベースである. MySQL と比べてパフォーマンス, 安定性, 開示性が高く, もっとも有名なリレーショナルデータベースの一つとされている. MySQL との完全な互換性, GPL の採用, さらには開発者コミュニティが豊富なので MariaDB は MySQL の上位互換と言える.

2.3 CSS

CSS とは”Cascading Style Sheets”の略で, HTML や XML 文書の見栄えなどを整えるために使われるスタイルシート言語である. HTML 内で指定されたタグや id などをセレクタで指定して色や段落などの設定ができる.

2.4 Laravel

Laravel は MVC 方式の PHP フレームワークである. CodeIgniter や Yii などの PHP フレームワークや Ruby on Rails などのプログラム言語の基本的な機能を組み込んでいるので豊富な機能が使用できることが特徴である. なのでスクラッチ開発でもセキュリティ対策などが簡単に実装できるので容易に Web 開発に取り組むことができる. さらに vue.js などの他のフレームワークからコンポーネントを持ってくることも可能なのでより Web アプリケーションの開発が容易になっている.

また Laravel にはプロジェクトの操作に便利な Artisan コマンドというものが搭載されている. このコマンドは Symphony というフレームワークを使って作られており, これを使用してデータベースの操作, ローカルサーバの立ち上げ, モデル・ビュー・コントローラの生成が簡単にできるようになっている.

さらにプログラムからデータベースに楽にアクセスするために Eloquent と呼ばれる ORM(Object Relational Mapper) が組み込まれている. これによって関数やモデルクラスの使用で SQL の操作が可能となっている.

またビューでは blade と呼ばれる Laravel 独自のテンプレートファイルが使用可能で, これを使用することで php ファイル内で条件に応じた処理や別ファイルの継承などを行って便利に表示をさせることができる.

2.5 HTTP

HTTP とは”Hypertext Transfer Protocol”の略でハイパーテキストを転送するためのアプリケーション層プロトコルである. 主にサーバ側とクライアント側との通信の目的のために使われるが別の用途でも使用されることがある. 基本的な動作としてはクライアント側がサーバ側にリクエストを送信するためにポートを開き, サーバ側からレスポンスが帰ってくるまで待機するという仕組みとなっている. また, HTTP は主に TCP/IP 層の上の通信で使用されるが, UDP のようなトランスポート層でも使用されることがある.

3. システム設計

3.1 システム概要

Laravel を使用して 3 つのコントローラ, 2 つのモデル, 11 のビューによって構成されている.

SimpleNoteController クラスは主に特にデータベースとの連携などを行わない処理の際に呼び出されるコントローラである. ここではトップページの表示, 新規登録の確認画面の表示, ログアウトなどでセッションの操作を行うことがほとんどである.

次に UserController クラスである. ここではユーザデータの操作を行っており, 新規登録の完了, ログイン情報のチェック, 登録情報の編集が機能となっている. またデータベースにユーザデータを挿入する際のパスワードのハッシュ化もここで行っている.

最後に NoteController クラスである. ここでは備忘録の操作を行っている. トップページでの表示の際に id の降順にペジネーションしての表示, 備忘録の投稿, マイページでの表示が主な機能となっている. さらにプライベート投稿であるかのチェックもここで行っており, プライベート投稿であればフラグを立てたものをデータベースに登録している. そして備忘録を表示する際にはビュー側でこのフラグを検知して非表示にするかを判断している. また, これら UserController と NoteController はバリデーションもここで行っており, それぞれに独自の Request クラスを実装してリクエストが渡された際にクラスの機能に応じたバリデーションを行うことでユーザ入力のルール違反をチェックしている. 主にパスワードは 8 文字以上, 投稿文字数は 400 文字以内などのルール設定がここで行われ, チェックされている.

そしてモデルは User モデルクラスと Note モデルクラスがあり, それぞれ User モデルが Note モデルの主テーブルであることの宣言やデータベース挿入時のバリデーションなどの関数が搭載されている.

3.2 画面遷移

本システムの画面遷移図は図 1 のようになっている. まずトップページでは未ログインの状態ではサイドバーは投稿フォームとログインボタン, 既ログインの状態では投稿フォームとマイページ, 登録情報編集, ログアウトへのリンクが設置されている. それ以外はどちらも備忘録の表示のみで同じ構成となっている. ログインの際にはメールアドレスとパスワードの入力フォームが設置されている. ここにログイン情報を入力することでログイン状態になることができる. またデータが不十分であったり, 登録データが存在しない場合はエラー表示がされる.

新規登録画面では投稿の際に表示されるユーザネーム,

一意の識別のために使用されるメールアドレス, パスワードの設定を要求される. これらも条件を満たしていないとエラー表示がされ, 満たしている場合は確認画面でチェックを促したのちに登録完了となり, ユーザデータがデータベースに挿入される. ログイン後は設定したユーザネームが投稿フォームに表示される. マイ備忘録ではログインしたユーザの自分の投稿を見ることができる. そして登録情報変更では新規登録画面と同じフォームで登録した情報を変更することができる.

備忘録の投稿においては未ログインであれば名無し, 既ログインであれば登録時のユーザネームを使って投稿されるのでログインしていてもいなくても投稿することができる. しかし未ログインの場合はプライベート投稿であるかのチェック工モックは無くなっている.

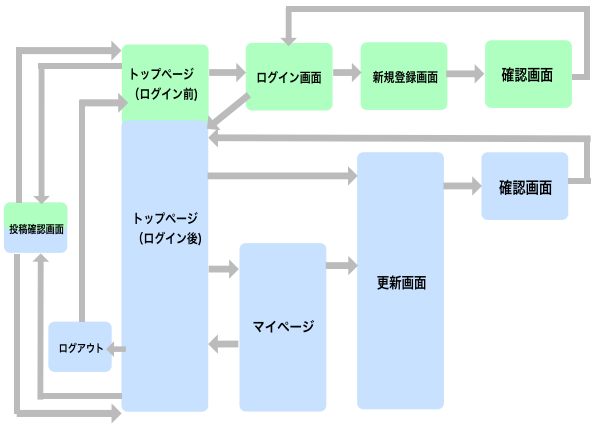

図 1 状態遷移図

3.3 データベース設計

本システムで使用されているデータベースでは次の 2 つのテーブルが使われている.

- ユーザ情報テーブル
- 備忘録テーブル

まずユーザ情報テーブルの構成は表 1 のようになっている. 個々の識別用の email カラム, ユーザネームを格納する user_name カラム, パスワードを格納する password カラムがこのテーブルでは使用されている. またメールアドレスとパスワードはともに 128 文字, ユーザネームは 50 文字の領域が割り当てられている. どちらも指定文字数を超えるとエラーが出るようになっているので格納時の例外は出ることはない.

表 1 ユーザ情報テーブル構成

Field	Type	Null	Key	Default	Extra
id	int(10) unsigned	NO	PRIMARY	NULL	auto_increment
email	varchar(128)	NO	UNIQUE	NULL	
user_name	varchar(50)	NO		NULL	
password	varchar(128)	NO		NULL	

次に備忘録テーブルである. 備忘録テーブルの構成は表 2 のようになっている. 投稿時の西暦から秒数までの時間を格納する created_at カラム, タイトルを格納する subject カラム, 本文を格納する text カラム, 誰が投稿したかのユーザ id を格納する user_id カラム, プライベート投稿であるかの真偽が格納されている is_private カラムがこのテーブルで使用されている. また, subject カラムは 100 文字, text カラムは 500 文字の領域が割り当てられている. 本文である text カラムは 400 文字までという制限が設けられているが, 念のため余分に領域を割り当てている. is_private カラムでは 1bit の tinyint 型が付けられているのでこの値が 1 か 0 かでプライベート投稿であるかをチェックしている.

今回は Laravel を使用しているのでテーブル同士の関連づけが簡単にできるようになっている. よって user_id に格納された id のユーザを参照してこの備忘録が誰によって投稿されたものなのかを表示している. この機能はマイページでの表示にも使われている.

表 2 備忘録テーブル構成

Field	Type	Null	Key	Default	Extra
id	int(10) unsigned	NO	PRIMARY	NULL	auto_increment
created_at	timestamp	NO		current_timestamps()	
subject	varchar(100)	NO		NULL	
text	varchar(500)	NO		NULL	
user_id	int(11)	YES		NULL	
is_private	tinyint(1)	NO		NULL	

3.4 システム詳細

前述の通り, 今回は Laravel を使用しているので MVC 方式でファイルを分けて構成されている. このシステムで主に働いているファイルは以下の通りである.

- コントローラ
  - SimpleNoteController.php
  - UserController.php
  - NoteController.php
- モデル
  - User.php
  - Notebook.php
- ビュー
  - index.blade.php
  - login.blade.php
  - register.blade.php
  - registerConfirm.blade.php
  - registerComplete.blade.php
  - editUser.blade.php
  - editConfirm.blade.php
  - editComplete.blade.php
  - mypage.blade.php
  - uploadConfirm.blade.php
  - template.blade.php

- リクエストクラス
  - EditRequest.php
  - RegisterRequest.php
  - UploadRequest.php

SimpleNoteController クラスでは特に深くデータベースとの連携がない処理を行っているので index メソッドや login メソッド, register メソッドではそのままトップページや登録画面を表示するのみとなっている．また, regist_confirm メソッドでは新規登録の確認画面の際に html の方に表示するデータの転送とセッションへの格納により入力したデータを確認として表示させている．

また logout メソッドではセッションに格納してあるログイン ID とユーザネームを消してリダイレクトすることでログアウト機能を実現している．

UserController クラスではユーザ情報をモデルとビュー間でやり取りする処理が行われている．regist_complete メソッドではユーザ情報をデータベースに格納させる処理を行っている．前ページからセッションで受け取ったユーザデータにバリデーションを行ったのちにユーザネームとメールアドレス, さらにパスワードをハッシュ化した状態で格納させている．そして login_check メソッドではログイン時のデータチェックを行っている．データベース内から入力メールアドレスと登録メールアドレスを比較して一致したものがない, またはメールアドレスは一致していたとしてもパスワードが一致していない場合はその趣旨のエラーをビューに返すようにしている．

edit メソッドは登録情報を編集する際に入力フォームに登録データを入力しておくために呼び出され, edit_confirm メソッドで regist_confirm メソッドと同じく確認画面のためのデータ転送, セッションへの格納を行っている．そして edit_complete メソッドで入力された変更情報を再挿入している．

NoteController クラスでは備忘録情報をモデルとビュー間でやり取りする処理が行われている．upload メソッドではトップページで入力された備忘録情報を受け取り, プライベート投稿であるかのフラグを boolean に変換した状態でセッションに格納した上で確認画面に飛ばしている．そして upload_complete メソッドで実際に入力されたデータをデータベースに格納するという仕組みになっている．ユーザデータの格納と違うところはプライベート投稿であるかの boolean を判別する部分のみで他はほとんど同じとなっている．そして mypage メソッドではマイページで表示する備忘録の条件指定を行っている．現在ログインしているユーザの ID のみで条件を絞り, 降順で表示させることで自分の投稿のみを表示させるようにしている．

User モデルクラスでは自動繰上げである id カラムの保護, notebooks テーブルが従テーブルであることの宣言, バリデーションを指定している．また, Notebook モデルクラスでは自動繰上げ, 自動更新である id カラムと created_at カラムの保護, users テーブルが主テーブルであるのことの宣言, 同じくバリデーションを指定している．

リクエストクラスは確認画面に飛ぶ前に入力情報が条件に合っているかのバリデーション機能を使用するために実装している．今回は新規登録, 登録情報の変更, 備忘録の投稿の際にバリデーションを使用しているので, regist_confirm メソッドでは RegisterRequest クラスを呼び出して新規登録時のバリデーション, edit_confirm メソッドでは EditRequest クラスを呼び出して登録情報変更時のバリデーション, upload メソッドでは UploadRequest クラスを呼び出して備忘録投稿時のバリデーションを行って, 機能ごとそれぞれのバリデーションを実装している．

3.5 変数設計

今回はセッション変数と POST 変数を多く使用してデータのやり取りをしている．ここで使用されている変数の一覧は以下のようになっている．

- セッション変数
  - *user_name：ユーザネームの格納
  - *mail：メールアドレスの格納
  - *password：パスワードの格納
  - login_id：ログインしているユーザ ID の格納
  - login_user：ログインしているユーザネームの格納
  - *title：備忘録の件名の格納
  - *texts：備忘録の本文の格納
  - *isPrivate：プライベート投稿であるかのフラグの格納
- POST 変数
  - user：ユーザネームの格納
  - mail：メールアドレスの格納
  - pass：パスワードの格納
  - isLoginError：ログインエラーである時のフラグの格納
  - notes：備忘録データの格納
  - title：備忘録の件名の格納
  - texts：備忘録の本文の格納
  - isPrivate：プライベート投稿であるかのフラグの格納

4. 実装

4.1 実装環境

本システムの実装環境は以下の通りである．

- MariaDB 10.4.21
- PHP 8.1.4
- Laravel 4.2.10

- Composer 2.3.9

4.2 環境設定

4.3 動作検証

5. まとめ

参考文献

[1] PHP Group . ”一般的な情報” . php. https://www.php.net/manual/ja/faq.general.php . (閲覧日:2022/05/29)
[2] PHP Group . ”Zend API: PHP のコアをハックする” . php . http://php.adamharvey.name/manual/ja/internals2.ze1.zendapi.php . (閲覧日:2022/05/29)
[3] PHP TUTORIAL . ”What is PHP” . phptutorial.net . https://www.phptutorial.net/php-tutorial/what-is-php/ (閲覧日:2022/05/29)
[4] KINSTA . ”MySQL とは？初心者にわかりやすい説明” . KINSTA . (更新日:2020/07/03) . https://kinsta.com/jp/knowledgebase/what-is-mysql/ . (閲覧日:2022/06/01)
[5] MariaDB Foundation . ”MariaDB Server: The open source relational database” . MariaDB Foundation . https://mariadb.org/ . (閲覧日:2022/06/01)
[6] Mark Smallcombe . ”MariaDB vs MySQL: 徹底比較” . integrate.io . (更新日:2020/09/03) . https://www.integrate.io/jp/blog/mariadb-vs-mysql-everything-you-need-to-know-ja/ . (閲覧日:2022/06/01)
[7] MDN contributors . ”CSS: カスケーディングスタイルシート” . mdn web docs . (更新日:2021/07/18) . https://developer.mozilla.org/ja/docs/Web/CSS . (閲覧日:2022/06/01)
[8] MDN contributors . ”HTTP” . mdn web docs . (更新日:2021/09/18) . https://developer.mozilla.org/ja/docs/Web/HTTP . (閲覧日:2022/06/01)