
ブログアプリ作成
授業資料一覧
< Seed Tech School />

< ブログアプリの仕様 />

- 実装する仕様

- **ログイン（認証）機能**がある

- 投稿機能の**CRUD処理**があり、投稿の
  - **新規作成・保存**
  - **一覧表示**
  - **詳細表示**
  - **更新**
  - **削除**

  ができる。

- コメント記入画面があり、コメントが記事に対してつけられる

<Laravel 環境構築手順 - Laravelインストール/>

Step1 Laravelのインストール（コマンド操作： XAMPP/htdocs下）
composer create-project --prefer-dist laravel/laravel（アプリ名）"バージョン"

ex) composer create-project --prefer-dist laravel/laravel lara-blog "7.*"
※上記は、htdocsに移動しターミナル(Mac)・git bash(Windows)上で行う

※下記以降は、アプリのフォルダに移動し VSCodeかターミナル(Mac)・git bash(Windows)上で行う
Step2 .envファイルの編集（DB_DATABASE, DB_SOCKET）
DB_SOCKET=/Applications/XAMPP/xamppfiles/var/mysql/mysql.sock
DB_DATABASE=任意の名前

Step3 DB作成（phpmyadminにて.envファイルに記載のDB名でデータベース作成）

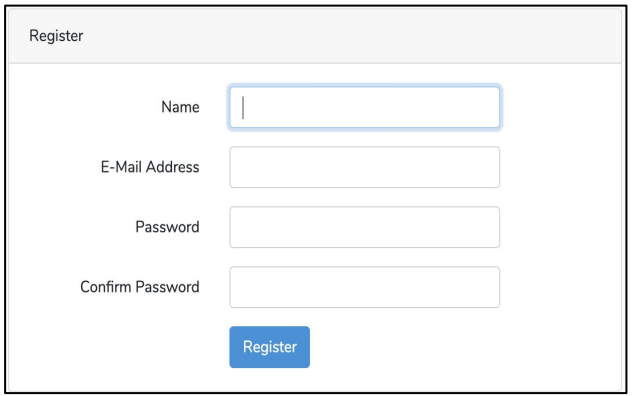
Step4 migrationファイル作成とmigrateの実行 ※テーブル作成の際に実行

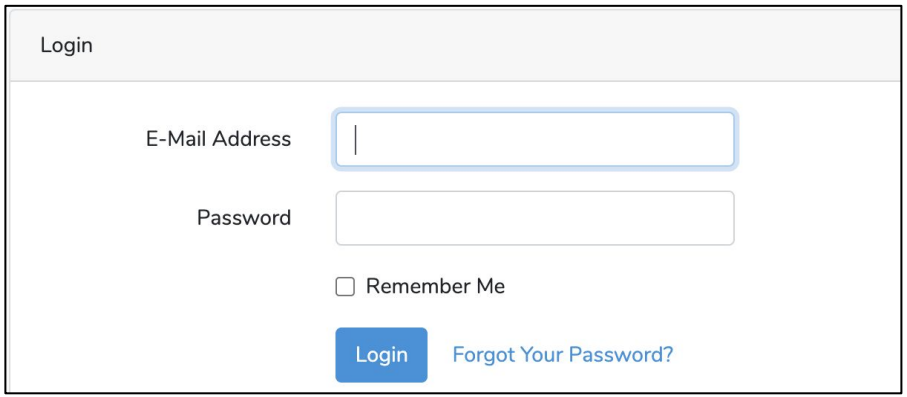
Step5 php artisan key:generate実行

Step6 php artisan serve

< 認証機能の作成 />

認証機能はユーザーを認識する必要のある全てのシステムで必須です





< 認証機能コマンド/>

認証機能（Auth）作成 → npmのインストールが完了している必要がある

Step1	composer require laravel/ui "^2.0"
Step2	php artisan ui vue --auth
Step3	npm install
Step4	npm run dev

< モデル・マイグレーションファイルの作成 />

コマンド	php artisan make:model Post --migration

- app配下に、Post.phpが作成される
  class Post ができあがる
- database/migration配下に、日時 _create_posts_table.phpが作成される

↓

マイグレーションファイル（日時 _create_posts_table.php）に必要な情報を記述する

↓

コマンド	php artisan migrate

- データベースにpostsテーブルが作成される

< ビューファイルの準備 />

- 下記よりファイルをダウンロード

  https://github.com/NexSeed00/new_practice/blob/master/Laravel/lara_blog-main.zip?raw=true

- resources/viewsフォルダ内にダウンロードしたデータを移動

- ビューファイルをそれぞれ「blade.php」と書き換える

- 内容の編集（動画を見ながらゆっくりやっていきましょう！）

< ログイン後の画面 />


ユーザA
投稿一覧
新規投稿
タイトル : おはよう
内容 : 今日のセブは快晴
投稿者：Seed Techさん
詳細へ
投稿日時 : 2021/11/08
Copyright © Seedkun Inc.

< ルーティングの基本 - CRUD命名 />

CRUD処理を作成する際の命名（**「7つのアクション」を覚えましょう！**）

HTTPヘッダ	URL	アクション	役割	Route Name
GET	/comments	index	一覧表示	comments.index
GET	/comments/create	create	新規投稿	comments.create
POST	/comments	store	新規投稿保存	comments.store
GET	/comments/{id}	show	詳細画面	comments.show
GET	/comments/{id}/edit	edit	編集更新画面	comments.edit
PUT/PATCH	/comments/{id}	update	更新処理	comments.update
DELETE	/comments/{id}	destroy	削除	comments.destroy

< ルーティングの基本 />

routes/web.phpに記載　※コントローラー名はその都度変わります。

一覧表示	Route::get('/comments, 'CommentController@index')->name('comments.index');
新規投稿	Route::get('/comments/create','CommentController@create')->name('comments.create');
新規投稿保存	Route::post('/comments, 'CommentController@store')->name('comments.store');
詳細画面	Route::get('/comments/{id}', 'CommentController@show')->name('comments.show');
編集更新画面	Route::get('/comments/{id}/edit', 'CommentController@edit')->name('comments.edit');
更新処理	Route::put('/comments/{id}', 'CommentController@update')->name('comments.update');
削除	Route::delete('/comments/{id}', 'CommentController@destroy')->name('comments.destroy');

★上記を一行にまとめて下記のように記載できる

Route::resource('comments', 'CommentController');

< 投稿CRUD処理 一覧画面 />

< コントローラーの作成 />

コマンド	php artisan make:controller PostController

- app/Http/Controllers配下に、PostController.phpが作成される
  class PostController ができあがる

< 投稿CRUD処理 新規作成画面 />

< 補足:@csrfについて/>

csrf_tokenを意識する(Readouble参照)

Laravelでは、クロス・サイト・リクエスト・フォージェリ(CSRF)からアプリケーションを簡単に守れます。クロス・サイト・リクエスト・フォージェリは悪意のある攻撃の一種であり、信頼できるユーザーになり代わり、認められていないコマンドを実行します。

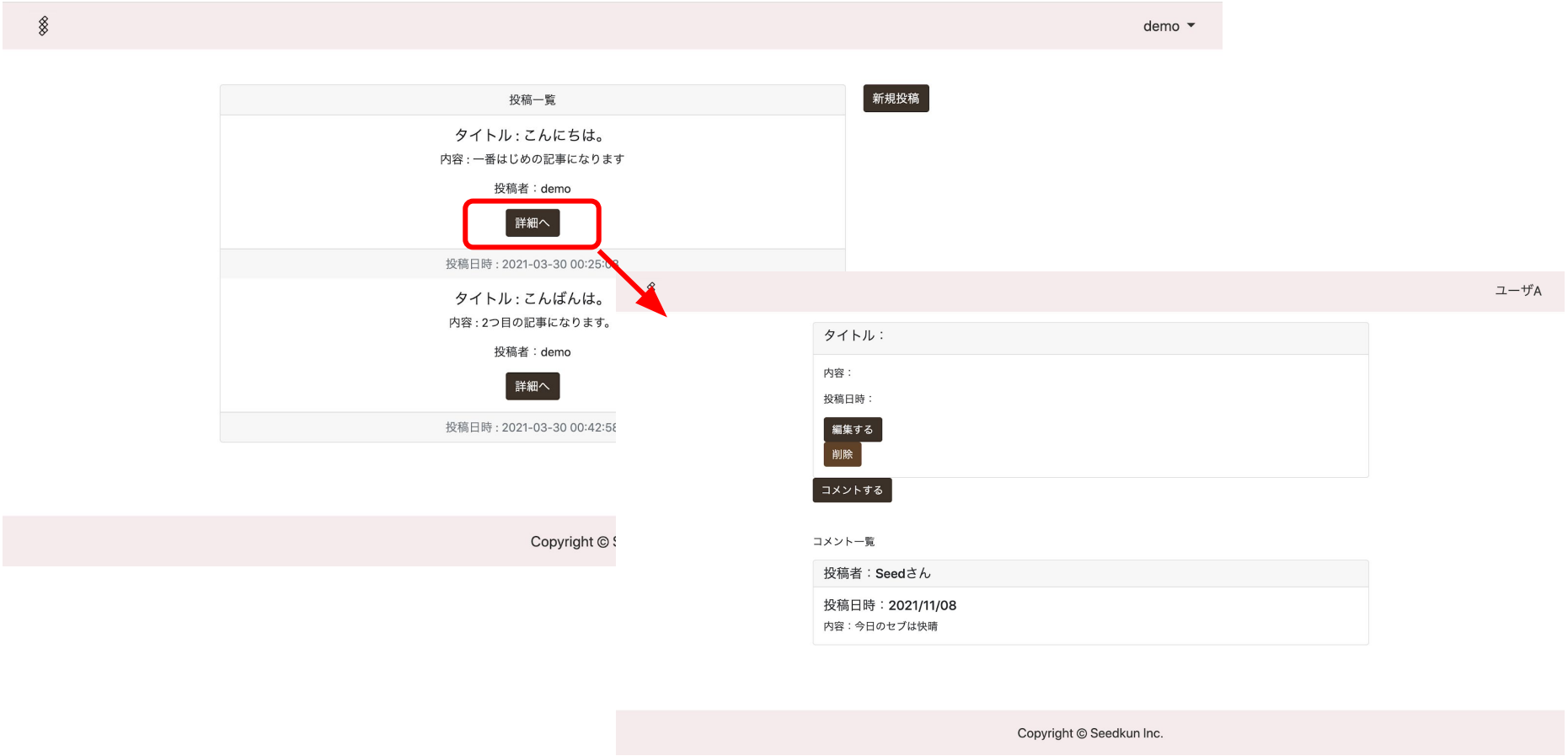
Laravelは、アプリケーションにより管理されているアクティブなユーザーの各セッションごとに、CSRF「トークン」を自動的に生成しています。このトークンを認証済みのユーザーが、実装にアプリケーションに対してリクエストを送信しているのかを確認するために利用します。

< 補足：csrf_token 使い方 />

headタグ内	<meta name="csrf-token" content="{{ csrf_token() }}">

formタグ内	@csrf

< 投稿CRUD処理 詳細画面 />

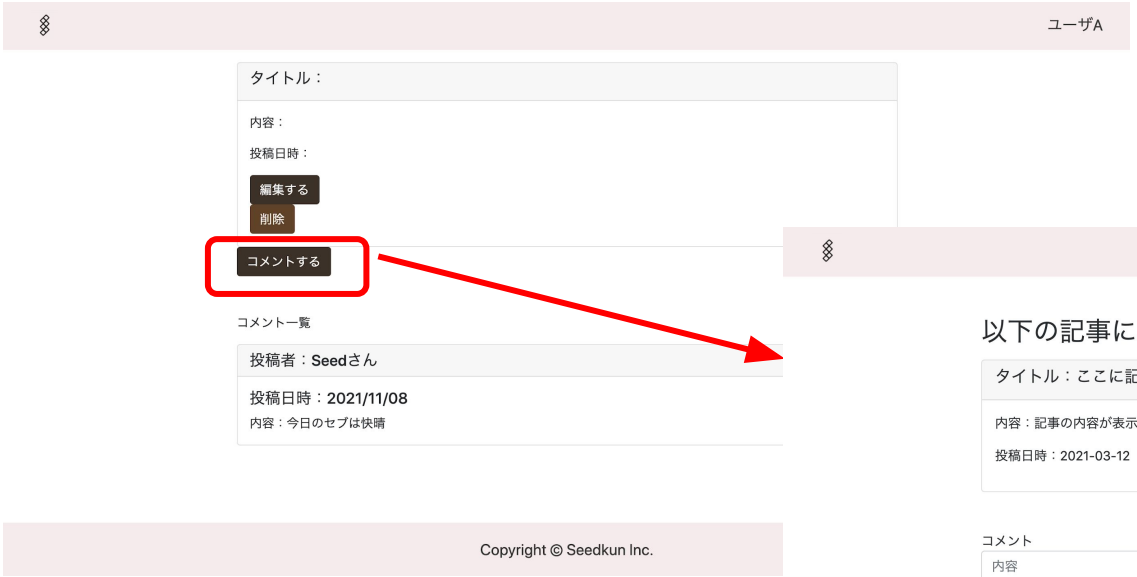


< 投稿CRUD処理 編集画面 />

< 投稿CRUD処理 削除/>


ユーザA
タイトル：
内容：
投稿日時：
削除
コメントする
コメント一覧
投稿者：Seedさん
投稿日時：2021/11/08
内容：今日のセブは快晴
demo
投稿一覧
新規投稿
タイトル：こんにちは。
内容：一番はじめの記事になります
投稿者：demo
詳細へ
投稿日時：2021-03-30 00:25:08
タイトル：こんばんは。
内容：2つ目の記事になります。
投稿者：demo
詳細へ
投稿日時：2021-03-30 00:42:58
Copyright © Seedkun Inc.

< コメント追加処理 />


ユーザA
タイトル：
内容：
投稿日時：
編集する
削除
コメントする
コメント一覧
投稿者：Seedさん
投稿日時：2021/11/08
内容：今日のセブは快晴
Copyright © Seedkun Inc.


ユーザA
以下の記事にコメントします
タイトル：ここに記事タイトルが表示されます
内容：記事の内容が表示されます。記事の内容が表示されます。記事の内...
投稿日時：2021-03-12
コメント
内容
コメントする
Copyright © Seedkun Inc.

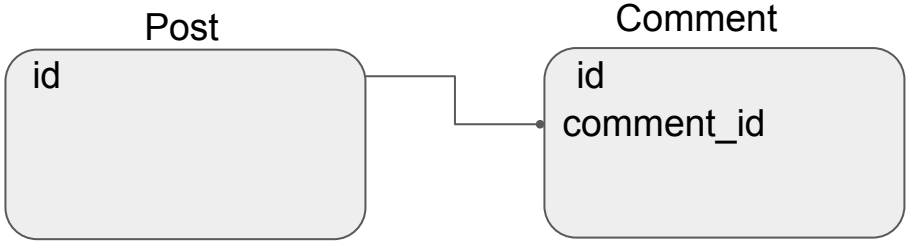
< コメント一覧表示 />


demo
タイトル：おはようございます。
内容：一番はじめの記事になります
投稿日時：2021-03-30 00:25:08
編集する
削除
コメントする
コメント一覧
投稿者：Seedさん
投稿日時：2021/11/08
内容：今日のセブは快晴
Copyright © Seedkun Inc.

< リレーションとは />

リレーション:
データベースの考え方に基づいたもので、モデルとモデル間でのデータの関連性を表したものをリレーションと呼びます。

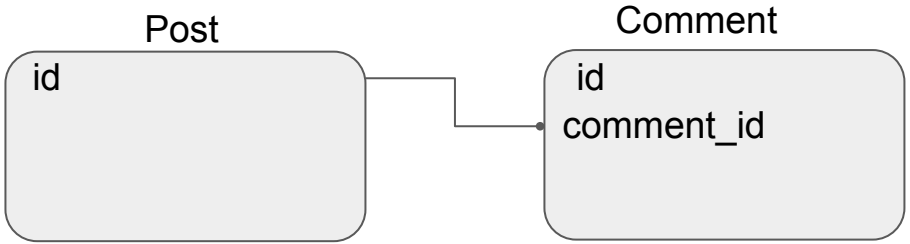
例:
Postという投稿データのモデルが有り、記事情報を持っているとします。
その記事情報に対してコメントを扱うCommentというモデルが有った場合、どのPostのCommentなのかを知るためにCommentにPostのid(識別子)であるpost_idという情報を持つことで、モデルPostとモデルCommentの関係性を表すことができます。これがリレーションです。



< リレーションでの1対多とは />

1データに対し、リレーションしているデータが複数存在している関係を「**1対多**」といいます。

投稿1つ分のデータに対し、Commentのデータは複数存在できるため、**1対多の関係性**になっていると言えます。



記事一つに対し寄せられるコメントは多数

< テーブル構造 />

postsテーブル

カラム名	論理名	データ型	説明
id	ID	bigInt	テーブル内データの通し番号
title	タイトル	text	投稿のタイトル
body	投稿内容	text	投稿内容
user_id	投稿ユーザー	bigInt	投稿ユーザーの ID
created_at	作成日時	timestamp	初めて投稿された日時
updated_at	更新日時	timestamp	投稿が更新された日時

< テーブル構造 />

commentsテーブル

カラム名	論理名	データ型	説明
id	ID	int	テーブル内データの通し番号
body	コメント内容	text	コメント内容
user_id	ユーザID	bigInt	コメント記入したユーザの ID
post_id	投稿ID	bigInt	コメント対象記事の ID
created_at	作成日時	timestamp	初めてコメントされた日時
updated_at	更新日時	timestamp	コメントが更新された日時

< ビューファイルの準備 />

- 下記よりファイルをダウンロード

  https://github.com/NexSeed00/new_practice/blob/master/Laravel/comment_view.zip?raw=true

- resources/viewsフォルダにcommentsフォルダを作成

- create.htmlを移動、blade.phpに拡張子を変更

- 内容の確認

< 補足 : 命名規則 />

種類	記法	複数or単数	例
Migration名	スネーク	複数	日付_create_comments_table
テーブル名	スネーク	複数	comments
モデル名	キャメル	単数	Comment or commentData
コントローラ名	キャメル	単数	CommentController
ビュー名	スネーク	複数	comments or comment_data