Cài đặt MySQL (mySQL & mySql workbench) -> Tắt autorun mysql trong services của Windows

Cài đặt xampp (tạo webserver giả lập trên localhost)

Chọn config -> httpd -> tìm DocumentRoot -> Chọn lại đường dẫn đến folder đang làm việc

Cài đặt Composer (tương tự như npm)

Cài đặt variable mysql: xampp -> mysql -> bin -> copy path to enviroment

Chạy terminal mysql -uroot -p

Tạo DB: create database tenDB;

Show DB: show databases;

1. **MVC**

Model: deals with database

Tạo model và migrate

php artisan make:model Role -m

tên Model viết Hoa, -m sẽ đồng thời tạo migrate cho Role

View: deals with HTML

Controller: middle View and Model

1. **Route**

Tạo route mới trong routes -> web.php

1. **Controller**

Namespace: để có thể dùng file ở nhiều chỗ khác

Use: dùng để import class,namespace function vào file

Tạo 1 controller: Terminal -> php artisan make:controller –resource ControllerName

Routing Controller:

Route::*get*('/post', '\App\Http\Controllers\PostController@index');

Route::*get*('/post/{id}', '\App\Http\Controllers\PostController@index');

Truyền param vào route, nhận lại param ở index trong Controller

Route::*get*('/contact', '\App\Http\Controllers\PostController@contact');

Tạo 1 function riêng trong controller, nếu muốn truy cập tới chức năng của function đó thì thêm @tên sau tên Controller

a

**Resources, Controller resources**

Route::*resource*('post', '\App\Http\Controllers\PostController');

Trong 1 controller khi return view, view sẽ return theo tên file trong thư mục resources/views

1. **Views**

Blade template: cho phép viết code php trong html ko cần <?php?>, thay bằng {{}}

Truyền data tới views

Controller nhận data từ route -> return lại trong view

*public* function *show\_post*($id)

    {

*return* *view*('post')->*with*('id', $id);

    }

1. **Blade engine**

Tạo layout chung cho các page

Trong folder view tạo 1 app.blade.php làm layout chung

<body>

  <div *class*="container">

    @yield('content')

  </div>

  @yield('footer')

</body>

Tại page cần dùng layout @extends

@extends('layouts.app')

@section('content')

<h1>Contact Page</h1>

@stop

1. **Window OS Migration**

Migration thiết lập các quy định cho table

Khi muốn xóa,hay chỉnh sửa migration = > php artisan migrate:reset => chỉnh sửa => php artisan migrate

(xóa các migration cũ trên db, edit, migrate lại)

Php artisan migrate

* Tạo ra 1 migrate user dựa vào thiết kế schema trong folder database -> migrations
* Các datatype trong laravel (search từ khóa migration trên docs laravel)

**Tạo 1 migration Schema::create**

php artisan make:migration create\_posts\_table –create=”posts” (Tạo ra 1 schema để edit các column)

php artisan migrate (tạo migrate table lên phpmyadmin)

php artisan migrate:refresh (refresh mysql db)

**Tạo column trong 1 table = Schema::table**

php artisan make:migration add\_is\_admin\_column\_to\_posts\_table –table=”posts”



trong function down

$table->dropColumn(‘is\_admin’)

1. **Raw SQL Queries (**insert,reading,updating,deleting)

* **Insert Data**
* use Illuminate\Support\Facades\DB;
* Route::*get*('/insert', function () {
* DB::*insert*('insert into posts(title,content) values (?, ?)',
* ['PHP with laravel', 'Laravel is the best']);
* });
* **Reading data**

Route::*get*('/read', function () {

    $results = DB::*select*('select \* from posts where id = ?', [1]);

    // *var\_dump($results); //log ra results*

*forEach*($results as $result){

*return* $result->title;

    }

});

* **Update Data**

Route::*get*('/update', function () {

    $updated = DB::*update*('update posts set title = "Update title" where id = ?', [1]);

*return* $updated;

});

* **Delete Data**

Route::*get*('/delete',function(){

    $deleted = DB::*delete*('delete from posts where id = ?',[1]);

*return* $deleted;

});

1. **Eloquent/ORM (Tương tác với Database SQL)**

* Reading Data

( Model created in app/Model) tương tác vs database bằng các method của Eloquent

Route::*get*('/read',function(){

    $posts = Post::*all*();

*foreach*($posts as $post){

*return* $post->title;

    }

});

* Find Data

Route::*get*('/find',function(){

    $post = Post::*find*(3);

*return* $post->title;

    // *foreach($posts as $post){*

    // *return $post->title;*

    // *}*

});

* Find Data with condition

Route::*get*('/findwhere', function () {

    $posts = Post::*where*('id', 3)->*orderBy*('id', 'desc')->*take*(1)->*get*();

*return* $posts;

});

Route::*get*('/findmore',function(){

    // *$posts = Post::findOrFail(2);*

    // *return $posts;*

    $posts = Post::*where*('users\_count','<',50)->*firstOrFail*();

})

* Inserting Data using Eloquent

Route::*get*('/basicinsert',function(){

    //*create new post*

    $post = new Post;

    //*add props*

    $post->title = 'new Eloquent title';

    $post->content = 'Eloquent is coold';

    //*save*

    $post->*save*();

});

* Updating Data using Eloquent

Route::*get*('/basicinsert2',function(){

    //*findpost*

    $post = Post::*find*(2);

    //*update props*

    $post->title = 'new Eloquent title 2';

    $post->content = 'Eloquent is cool 2';

    //*save*

    $post->*save*();

});

* Creating data and configuring mass assignment

Route::*get*('/create', function () {

    Post::*create*(['title' => 'the Create method', 'content' => 'I\'m learning PHP']);

});

* Updating with Eloquent

Route::*get*('/update', function () {

    Post::*where*('id', 2)->*where*('is\_admin', 0)->*update*(['title' => 'New PHP title', 'content' => "New content"]);

});

* Delete with Eloquent

Route::*get*('/delete',function(){

    $post = Post::*find*(2);

    $post->*delete*();

});

Route::*get*('/delete2', function () {

    Post::*destroy*([3,4,5]); //*Delete multi id*

    // *Post::where('is\_admin',0)->delete(); //Delete with condition*

});

* Soft Delete / Trash

Tạo migrate add column deleted\_at



Config migrate SoftDelete

Text

Description automatically generated

Config Model SoftDelete

Text

Description automatically generated

Route::*get*('/softdelete',function(){

    Post::*find*(1)->*delete*();

});

* Read soft deleted

Route::*get*('/readsoftdelete',function(){

    // *$post = Post::find(1);*

    // *return $post;*

    // *Trả về tất cả record*

    $post = Post::*withTrashed*()->*where*('is\_admin',0)->*get*();

*return* $post;

    // *Chỉ trả về trashed record*

    // *$post = Post::onlyTrashed()->where('is\_admin',0)->get();*

    // *return $post;*

});

* Restored Data

Route::*get*('/restore',function(){

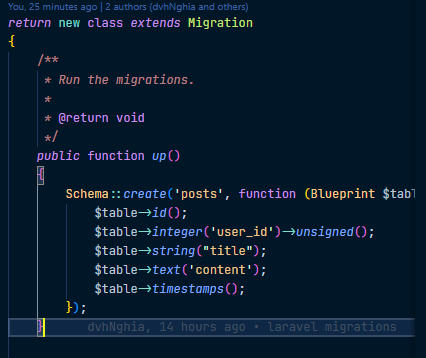
    Post::*withTrashed*()->*where*('is\_admin',0)->*restore*();

});

1. **Eloquent Relationships**

* **One to One**

Tạo user\_id trong Migrate của Post



Tạo 1 function trong User model, sử dụng hasOne method

*public* function *post*(){

*return* $this->*hasOne*('\App\Models\Post');

    }

Tại Route

Route::*get*('/user/{id}/post',function($id){

*return* User::*find*($id)->post;

});

* **Inverse Relation**

Model Post

*public* function *user*()

    {

*return* $this->*belongsTo*('App\Models\User');

    }

Route

Route::*get*('/post/{id}/user',function($id){

*return* Post::*find*($id)->user->name;

});

* **One to Many Relationship**

Tạo thêm 1 Post trg database

Tại model User

*public* function *posts*(){

*return* $this->*hasMany*('App\Models\Post');

    }

Tại Route

Route::*get*('/posts',function(){

    $user = User::*find*(1);

*foreach*($user->posts as $post){

        echo $post->title . "<br>";

    }

});

* **Many to Many**

Tạo 1 Model Role và tạo table user\_id,role\_id liên kết 2 table user và roles

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

Nhúng model Role vào Model User

Text

Description automatically generated

Tại Route

Route::*get*('/user/{id}/role', function ($id) {

    $user = User::*find*($id)->*roles*()->*orderBy*('id','desc')->*get*();

*return* $user;

    // *foreach ($user->roles as $role) {*

    // *return $role->name;*

    // *}*

});