# Get start

## Chuyển terminal sang git bash

"C:\Program Files\Git\bin\sh.exe" --login -i

## Tạo pj

laravel new new-pj

## Chạy server

php artisan serve

## các func hữu ích

slug: biến chuỗi ‘Đây là chuỗi’ thành ‘day-la-chuoi’, hữu ích cho tạo link

use Illuminate\Support\Str;

…

str::*of*('Đây là chuỗi' )->slug('-')

# Thao tác với db

## kết nối db

trong file .env

**đổi tên db và thông tin đăng nhập đúng với db đã tạo**

DB\_DATABASE=demoProject  
DB\_USERNAME=root  
DB\_PASSWORD=

## **Lưu ý**

Nếu đang dùng MySQL dưới 5.7.7 hoặc MariaDB dưới 10.2.2

Để migrate thành công phải sửa file App\Providers\AppServiceProvider

use Illuminate\Support\Facades\Schema;  
  
class AppServiceProvider extends ServiceProvider  
{  
 public function boot()  
 {  
 //  
 Schema::defaultStringLength(191);  
 }

}

## **Tạo model và update bảng**

php artisan make:model group1 -m

trong database/migrations có 1 file mới được tạo có tên dạng 2021\_05\_29\_163801\_create\_group1\_table.php, chỉnh sửa func để định hình bảng

public function up()  
{  
 Schema::create('group1', function (Blueprint $table) {  
 $table->id();  
 $table->string('Name');  
 $table->integer('Age')->default(0);  
 $table->timestamps();  
 });  
}

xem them syntax tại <https://laravel.com/docs/8.x/migrations>

<https://laravel.com/docs/8.x/migrations#available-column-types>

sau khi chạy xong, chạy lệnh để migrate

**php artisan migrate**

## thêm cột

tạo migration để sửa đổi

php artisan make:migration make\_a\_column --table=table1

trong database/migrations mở file migration mới được tạo, bắt đầu thêm mới cột

Schema::create('table1', function (Blueprint $table) {  
 $table->id();  
 $table->string('row1');  
 $table->integer('row2')->default(0);   
});

# View

## logic

if, for, … viết trực tiếp không cần {{ }}

@if  
@endif  
  
@for($i = 0; $i< 10; $i++)  
@endfor

## Nhúng biến

dạng text

{{ $value }}

dạng html

{!! $value !!}

## nhúng link

### link sort url

{{ route('group1.page1') }}

### link asset

thư mục gốc mặc định sẽ là public

<link rel="stylesheet"  
 href="{{ asset('https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallback') }}">  
<link rel="stylesheet" href="{{ asset('Thumuccha/style.css') }}">

## return view

các view mặc định sẽ nằm trong resources/views, đuôi dạng .blade.php, muốn hiển thị view ta phải return trong file router hoặc controller

### không tham số

return view('thumuccha.view1');

### có tham số: truyền $value cho view1

return view('thumuccha.view1', compact('value'));

### chuyển sang link khác bằng sort url

return redirect()->route('sortUrl');

## tham chiếu

### lấy content view

#### không tham số

@include('view1')

#### có tham số

@include('view1', $value)

### lấy view làm template

có thể có nhiều section được include vào template

khi return sẽ return view chứ không return template

#### template

đánh dấu phần mà view sẽ include vào template

@yield('component1')

#### view

khai báo view làm template

@extends('thumuccha.template1')

đánh dấu phần sẽ include vào template, có thể viết ngắn hoặc dài

ngắn:

@section('component1', 'Hello')

dài

@section('component1')

<div>html ở đây</div>

@endsection

## pagination

tự động tạo thanh pagination từ bootstrap, laravel 8 sẽ bị lỗi do mặc định không dùng bootstrap nên phải sửa đổi ở file app/Providers/AppServiceProvider.php

public function boot()  
{  
 //  
 Paginator::*useBootstrap*();  
}

### controller

// latest lấy dữ liệu và sắp xếp theo mới nhất tới cũ nhất  
// paginate phân trang, mỗi trang 5 item  
$data = $this->category->latest()->paginate(5);  
return view('thumuccha.page1', compact('data'));

### view

<div>  
 {{ $data->links() }}  
</div>

# các form cơ bản

### post form

khi lấy data, attr name sẽ là key trong request

<form action="{{ route('group1.method1') }}" method="post">  
 {{--form phải có @csrf để sinh token--}}  
 @csrf  
 <input name="data1">  
 <select  
 class="form-control"  
 id="add-category-parent"  
 name="data2"  
 >  
 <option value="0">A</option>  
 <option value="1">B</option>  
 <option value="2">C</option>  
 </select>  
 <button type="submit">Post</button>  
</form>

trong app/Models/Category.php cho phép fill vào db

class Category extends Model  
{  
 use HasFactory;  
 protected $fillable= ['name', 'parent\_id', 'slug'];  
}

# Router và xử lý data

tài liệu:

<https://laravel.com/docs/5.0/eloquent#insert-update-delete>

## lưu ý

để trong router có thể gọi thẳng controller, phải uncomment 1 dòng trong file **To App/Providers/RouteServiceProvider.php**

protected $namespace = 'App\\Http\\Controllers';

để có thể create data ta cần cho phép fill vào cột

trong app/Models/Category.php cho phép fill vào 2 cột

class group1 extends Model  
{  
 use HasFactory;  
 protected $fillable= ['Name', 'Age'];  
}

## Router

các router được định nghĩa trong file routers/web.php

### get

#### cơ bản

trả về view khi trỏ đến link

Route::get('/', function () {  
 return view('home');  
});

#### kết nối tới controller

gọi đến 1 func của controller khi trỏ tới link, đồng thời định nghĩa sort url cho link

// nhóm link vào group  
Route::prefix('groupLink1')->group(function () {  
 Route::get('/', [  
 // sort url  
 'as' => 'group1.page1',  
 //trỏ về controller và gọi func page1()  
 'uses' => 'group1Controller@page1'  
 ]);  
 Route::get('/page2', [  
 'as' => 'gourp1.page2',  
 'uses' => 'group1Controller@page2'  
 ]);

});

#### get có tham số

// trỏ tới edit đồng thời truyền vào func biến $id  
// khi khai báo edit trong controller sẽ có dạng edit($id)  
Route::get('/edit/{id}', [  
 'as' => 'categories.edit',  
 'uses' => 'CategoryController@edit'  
]);

### post

cũng như get nhưng thay get thành post, các thao tác truyền tham số cũng tương tự

Route::post('/action1', [  
 'as' => 'group1.action1',  
 'uses' => 'group1Controller@action1'  
]);

## Controller

### Tạo controller

tạo mới model có link categories/create

Tạo controller group1Controller

**php artisan make:controller group1Controller**

### khởi tạo constructer cho model để truy cập database

use App\Models\group1Model;

…

private $group1;  
  
 public function \_\_construct(group1Model $group1Model)  
 {  
 $this->group1 = $group1Model;  
 }

…

### thao tác db

#### get all data

$data = $this->group1->all();

$data sẽ có dạng mảng:

[

{"id":1,

"Name":"ABC",

"Age":12,

"created\_at":null,

"updated\_at":null},

{"id":2,

"Name":"DEF",

"Age":15,

"created\_at":null,

"updated\_at":null}

]

#### find

tìm theo id

$data = $this-> group1->find($id);

#### create

$this->group1->create([  
 'Name' => 'Name1',  
 'Age' => 16

]);

#### update

$this->category->find($id)->update([  
 'Name' => 'Name1',  
 'Age' => 16  
]);

#### delete

có 2 cách delete, delete thẳng luôn và delete tạm thời, delete thẳng luôn thì chỉ cần delete mà không phải cài gì:

$this->category->find($id)->delete();

nếu muốn delete tạm thời thì phải migrate cho bảng thêm cột deleted\_at

public function up()  
{  
 Schema::table('categories', function (Blueprint $table) {  
 //  
 $table->softDeletes();  
 });  
}  
  
*/\*\*  
 \* Reverse the migrations.  
 \*  
 \** ***@return*** *void  
 \*/*public function down()  
{  
 Schema::table('categories', function (Blueprint $table) {  
 $table->dropSoftDeletes();  
 });  
}

kế đó phải khai báo thêm trong model

class Table1 extends Model  
{  
 use SoftDeletes;  
}

xem thêm tại: [***https://laravel.com/docs/8.x/eloquent#soft-deleting***](https://laravel.com/docs/8.x/eloquent#soft-deleting)

### tách component

thay vì làm tất cả trong controller thì nên chia ra component cho hệ thống không bị phình

#### component

trong app tạo thư mục Components, tạo mới component1.php

<?php  
  
namespace App\Components;  
  
class Component1  
{  
// lấy data gưi từ controller về, lưu vào $data  
 private $data;  
  
 public function \_\_construct($data)  
 {  
 $this->data = $data;  
 }  
  
 function action1()  
 {  
 // do something  
   
 // lấy data  
 $this->data;   
 return;  
 }  
}

#### controller

use App\Components\Component1;

…  
public function method1()  
{

// khởi tạo component và gửi data vào  
 $component1 = new Component1($data);

// gọi action1 từ component1  
 $component1->action1();

}

### nhận data từ post

router trỏ post về method1

// $request chứa data post về  
public function store(Request $request)  
{  
 // lấy data1 từ data trong request  
 $request->data1;  
}