**NGÔN NGỮ BLADE TEMPLATE(mang tính kế thừa cao)**

@yield(‘name\_element’)

**Để kế thừa cấu trúc(layout) của cha con sử dụng** @extends(‘name\_child’)

=>**phần tử trong cha để các phần tử con có thể kế thừa**

**Muốn kế thừa sử dụng**

@section(‘name\_element\_child’)

**<Viết các HTML cần hiện thị>**

@show

**Nếu muốn kế thừa một đoạn HTML ở cha thì :**

**Phần tử cha:** @section(‘name’)

@show

**Phần tử con:** empty(‘name’)

@include() => nhập toàn bộ một trang => vào một trang con

@isset($records)

// $records is defined and is not null...

@endisset

* ***Sử dụng kiểm tra một phần tử có tồn tại hay không nếu tồn tại thực thi câu lệnh bên trong***

@empty($records)

// $records is "empty"...

@endempty

* ***Sử dụng kiểm tra một biến có rỗng hay không , nếu có thì thực thi câu lệnh bên trong***

**MAGRATION(công cụ thiết kế database thông qua câu lệnh)**

**Có thể lưu lại lịch sử các lần tương tác trên CSDL=> có thể backup**

**Kết nối: nằm trong file Config**

Vào file .env để chỉnh cấu hình

Lên phpmyAdmin để tạo CSDL tương tự để kết nối

Tạo Bảng : php artisan make:migration name\_table

**Kích hoạt migration để tạo bảng gửi lên phpmyAdmin**

**Php artisan migrate**

* **ROLLBACK**

Dùng: php artisan migrate:rollback => quay lại bước trước

Dùng: php artisan migrate:rollback –step=(‘số bước cần lùi lại’)

* **RESET**

Dùng:php artisan migrate:refresh => chạy lại toàn bộ migrateZ(đưa toàn bộ bước về all bước 1)

Dùng:php artisan migrate:reset => reset toàn bộ thao tác trên migrate

* **ADD Colums**

Dùng: php artisan make:migration add\_<name\_colum>\_to\_<name\_table> --table=users

**-tạo các trường của TABLE**

**Cú pháp: $table->type\_name(‘name\_bien’)**

$table0>id() =>

$table()->biglncrements(“id”)

$table->string(‘name’,100) : kiểu dữ liệu VARCHAR với độ dài 100

$table->text(‘description) : kiểu dữ liệu TEXT số lượng ký tự lớn

$table->json(‘options’) : trường dữ liệu kiểu JSON {“k1”:”value1”,”k2”:”value2”}

$table->timestamps(0) : tạo hai trường creat\_at and update\_at

$table->enum(“level”,[‘easy’,’hard’]) : trường bao gồm các giá trị cho trước

$table->char(‘name’,100) : trường lưu chuỗi ngắn

$table->boolean(‘confirmed’) : trường lưu dữ liệu có 2 giá trị true and false

$table->float(‘amount’,8,2) : lưu trữ dữ liệu FLOAT

$table->integer(‘votes’): lưu trữ dữ liệu INT

$table->dataTime(‘creat\_at’,0) : lưu trữ dữ liệu thời gian

#muốn một trường có thể null **: $table->type\_name(‘name\_bien’) -> nullable();**

* **Các phương thức phụ khi tạo và cập nhật bảng**

**Cú pháp**: $table->type\_name(‘name\_bien’) **-> after(‘name’);** thêm trường ra saucột cho trước

**-**>autolncrement() :thiết lập INTERGER và tự động tăng(KEY)

**-**>nullable(): khai báo trường có thể null

**-**>unsigned(): thiết lập số nguyên không dấu

**-**>useCurrent():lưu trữ dữ liệu cho cột chính là mốc thời gian giện tại

**-**>default($value): thiết lập giá trị mặc định cho trường

**-**>unique(): không được phép trùng

**- KHÓA CHÍNH, KHÓA NGOÀI**

Thêm khóa ngoài vào bảng:

$table->unsignedBigInteger(‘post\_id’); //thêm cột khóa ngoài

**#cấu hình tham chiếu cho khóa ngoài**

$table->foreign(‘post\_id’)->references(‘id’)->on(‘post’)->onDelete(‘cascade’)

OnUpdate()// tự động cập nhật theo

//onDelete

**QUERY BUILDER LARAVER**

**Các lệnh my sql thông thường:**

INSERT INTO table\_name (column1, column2, column3, ...)

VALUES (value1, value2, value3, ...);

UPDATE table\_name

SET column1 = value1, column2 = value2, ...

WHERE condition;

DELETE FROM table\_name

WHERE condition;

**Mã hóa bcrypt**

DB:table(‘tên bảng’)->insert(

[ ]

)

**Xử lý trong controller tương tự viết insert trong controller rồi dùng route chuyến hướng đến controller để xử lý**

**Lấy dữ liệu từ DB**

DB::table(‘name\_table’)->select(các trường cần lấy)->get(

);

* Nếu lấy all thì bỏ Đoạn select
* $posts = DB::table('posts')->select('title')->first();

Lấy ra bản ghi đầu tiên

$posts = DB::table('posts')->where('id',1)->first();

Lấy ra bản ghi với điều kiện trong where

Lấy theo id của hàng sử dụng PT find(‘id’)

Đếm số luowngj phần tử: $count = DB::table(‘ten’)->where(điều kiện)->count();

Lấy giá trị lớn nhất : $count = DB::table(‘ten’)->max(‘tên trường’)

Lấy giá trị bé nhất : $count = DB::table(‘ten’)->min(‘tên trường’)

Lấy giá trị trung bình : $count = DB::table(‘ten’)->avg(‘tên trường’)

**Lấy dữ liệu từ nhiều bảng**

$posts = DB::table(‘ten\_bang’)

->join(‘name\_table\_can\_noi’,”key\_phụ” ,”=”,”key\_chinh\_bang\_can\_noi”)

->select(hiện thị những thong tin gì)

->get()

Lấy theo điều kiện DB::table()->where()

#thêm các toán tử so sánh vào giữa (mặc định dấu “=”)

**Thiết lập điều kiện kết hợp AND OR**

AND where([

Viết các điều kiện cách nhau bằng dấu ‘,’

])

OR where(

)

->orWhere(

)

Biểu thức thô: selectRaw, whereRaw, havingRaw, orderByRow, gropByRaw

Sử dụng orderBy() để sắp xếp

Lấy số lượng data nhất định

* Limit(so lượng phần tử)

**ELOQUENT ORM**

**Tạo model:**

C1: php artisan make:model Post

C2:php artisan make: model Post -m(tạo cả CSDL )

Post::all() lấy tất cả

Post::where()->get(); lấy dựa vào điều kiện

Post::where()->First():lấy phần tử đầu tiên dựa vào điều kiện

Post:: find():lấy phần tử dựa vào id

Post::find([1,2,3]): lấy nhiều data dựa vào các id

Post::orderBy()->get();

Post::selectRaw()->groupBy()->get();

**Thêm dữ liệu bằng phương thức save();**

Giống con trỏ c++

$post = new <name\_model>

$post-><ten phan tu> = <giá trị>

……………………….

$post->save();

Thêm bằng phương thức created()

Post::created([

*Nhập các trường dữ liệu*

])

\*\*\*\*\*cần phải khai báo một mảng cần xử lý bào trong model

 protected $fillable =['title','user\_id','votes'];

$fillable khbai báo các trường được thay đổi

Update

Post::where('id',$id)

        ->update([

            'title' => 'Bài viết mới',

            'user\_id' => 1,

            'votes' => 10,

        ]);

Tương tự delete

$post = Post::find($id)

        ->delete();

Post::destroy( , , ,) dùng để xóa một danh sách các phần tử bằng id

Tạo thùng rác cho bảng

Cú pháp: php artisan make:migration add\_softdelete\_to\_<name\_table>\_table –table=’<name\_table>’

Vào controller tương ứng : trong function up:

**$table->softDeletes();**

Trong function dow:

**$table->dropSoftDeletes();**

**Muốn lấy toàn bộ bản ghi kể cả đã xóa:**

**$post = Post::withTrashed()->get();**

Khôi phục dữ liệu đã xóa tamk thời:

$post = Post::onlyTrashad()->where(điệu klieenj)->restore();

Xóa vĩnh viễn một đoạn ghi:

Post::onlyTrashad()->where()->forceDelete();

**CSDL (mối quan hệ giữa các bảng)**

* **Tạo mối quan hệ 1 – 1:**

**-tạo bảng 2**

function FeaturedImages(){

        return $this->hasOne('App/Models/FeaturedImages');

    }

**Viết hàm vào bảng 1 để liên kết 1-1 với bảng 2(hasOne)**

**ở bảng 2 khai báo**

**function name\_table1(){**

**return $this->belongTos(modul1::class);**

**}**

**Các mối quan hệ khác tương tự**