Cài đặt Laravel

Để cài Laravel, các bạn sẽ đi qua 2 bước :

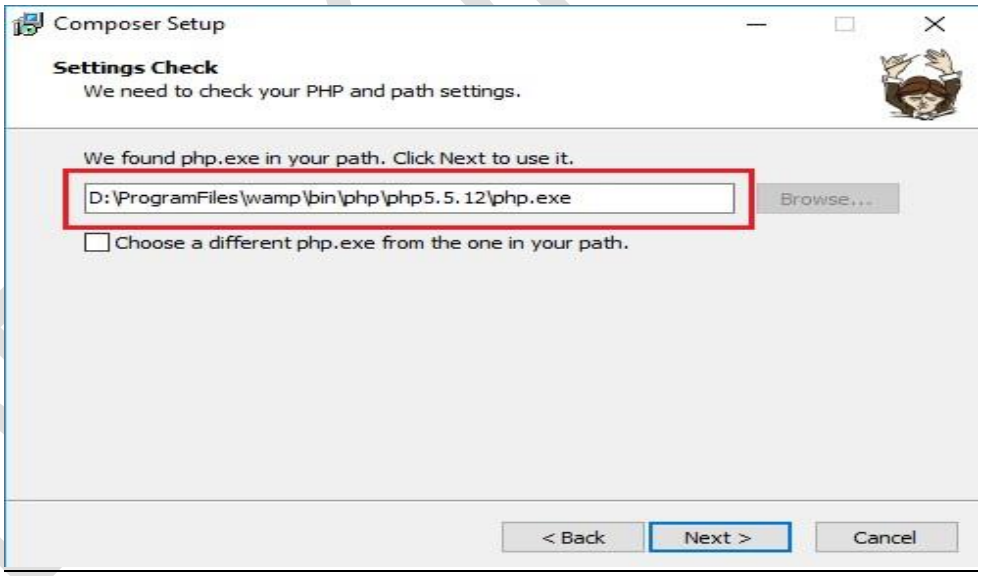
Bước 1: Cài đặt Composer

Bước 2: Cài đặt Laravel

Lưu ý : máy tính của bạn phải được cài sẵn Xampp, Wamp trước khi làm bài này.

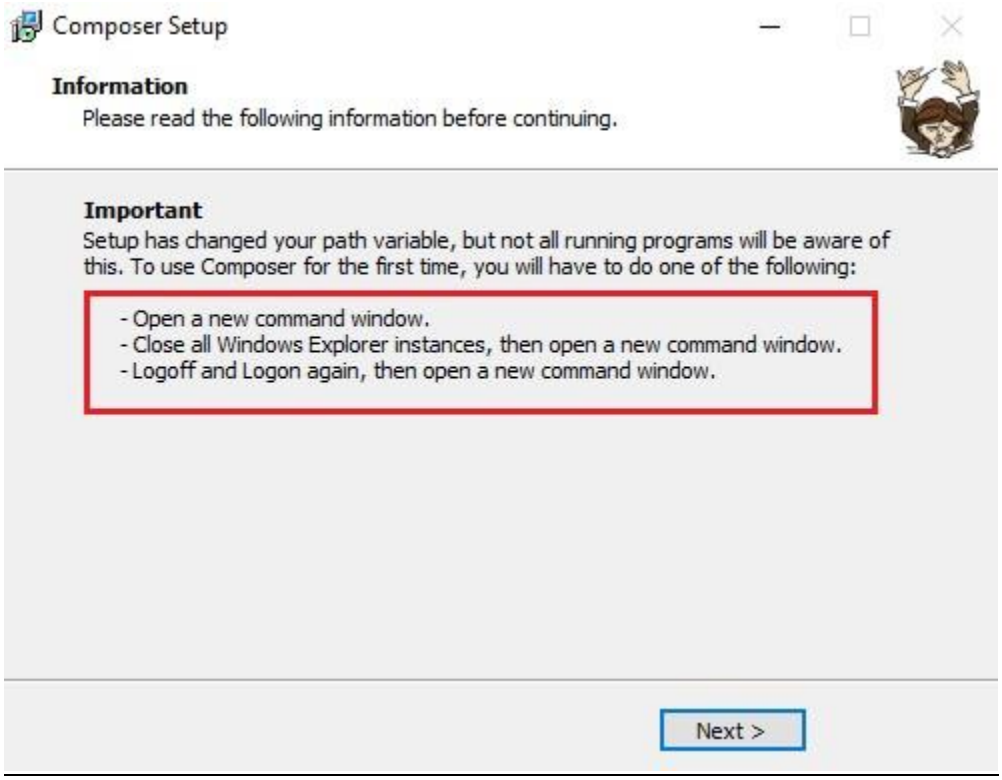
1. Cài đặt Composer

https://getcomposer.org/Composer-Setup.exe



Trong quá trình cài đặt, bạn sẽ được composer báo chọn đến file php.exe trong xampp hoặc

wamp



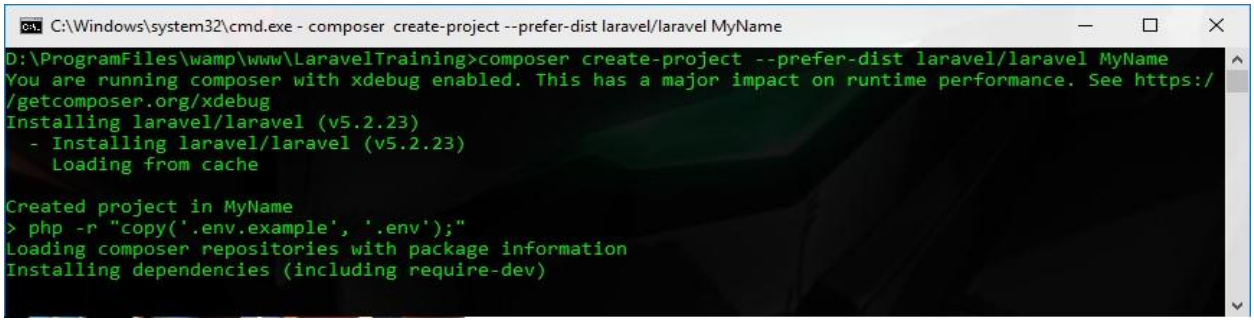
Sau khi cài đặt xong, composer có yêu cầu bạn làm các bước sau:

1. **Mở cmd.exe 2. Đóng tất cả các cửa sổ windows lại (bao gồm cả cmd.exe) 3. Mở lại cmd.exe**
4. **Đóng lại rồi logout ra khỏi windows , sau đó login lại.**
5. **Cuối cùng ta bật cmd.exe lên là xong.**

2. Cài đặt Laravel với composer

Khởi động cmd, truy cập vào thư mục muốn cài đặt Laravel.

composer create-project –prefer-dist laravel/laravel {Tên project Laravel}



3. Giới thiệu cấu trúc thư mục

+ App\Http : Chứa các bộ điều khiển route, controller,…. Ta sẽ phải dùng nhiều tới thư mục này.

+ Config : Chứa các file cấu hình cho hệ thống.

+ Database : Nơi chúng ta cấu hình các bộ dữ liệu mẫu : migrate, seed.

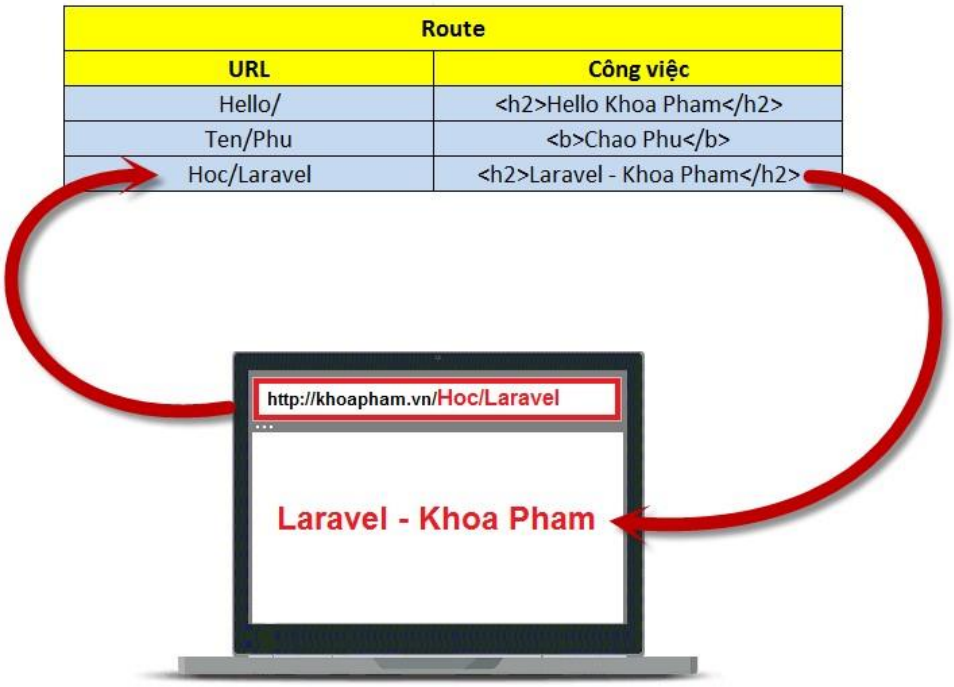
+ Public : Nơi lữu trữ các thư viện CSS, JavaScript, các hình ảnh.

+ Resources\Views : Lưu trữ các file giao diện mã html views.

+File .env: Cài đặt liên kết tới database cho hệ thống.

Tìm hiểu về Route

1. **Cấu trúc của Route**
2. **Truyền tham số trên Route**
3. **Định danh cho Route**
4. **Route Group**



1. Cấu trúc Route

2. Truyền tham số trên Route

2.1 Truyền tham số trên Route

Truyền tham số trên Route	Không truyền giá trị cho tham số

Route::get('myroute/{ten}' , function($ten){ return "Chào bạn" . $ten; });	Route::get('myroute/{ten?}' , function($ten = 'default'){ return "Chào bạn" . $ten; });

2.2 Đặt điều kiện cho tham số với phương thức where();

Điều kiện chữ	Điều kiện số
Route::get('myroute/{ten}' , function($ten){ return "Chào bạn" . $ten; })->where([' ten ' => ' *a-zA-Z++ ']);	Route::get('myroute/{so}' , function($ten){ return "Chào bạn" . $ten; })->where([' so ' => ' *0-9++ ']);

Các trường hợp khác

Tên trường hợp	Regular Expression
Chỉ cho phép số có từ 6 -> 32 số	' so ' => ' *0-9]{6,32} '
Chỉ cho phép số có 5 chữ số	' so ' => ' *0-9]{5}'
Cho phép cả chữ và số	' so ' => ' *0-9a-zA-Z++ '
Cho phpes cả chữ và số , giới hạn 6 ký tự	' so ' => ' *0-9a-zA-Z]{6} '

3. Định danh cho Route

Cách 1 : Khai báo 'as'=> 'Tên Route' trong tham số như sau:

```
Route::get( 'myroute' , [

'as' => 'newname' ,                    function() {

return "Đã đổi tên"; }

] );
```

Cách 2 : Cách này khá ngắn gọn và dễ dùng : thêm phương thức ***name('tên route')*** ở cuối.

```
Route::get( 'myroute' , function() { return "Đã đổi tên"; } )->name('tên route');
```

Gọi Route bằng tên đã đặt, ta sử dụng ***route('ten route');***

```
Route::get( 'myroute' , function(){        return redirect()->route('tên route');
```

```
});
```

4. Route Group

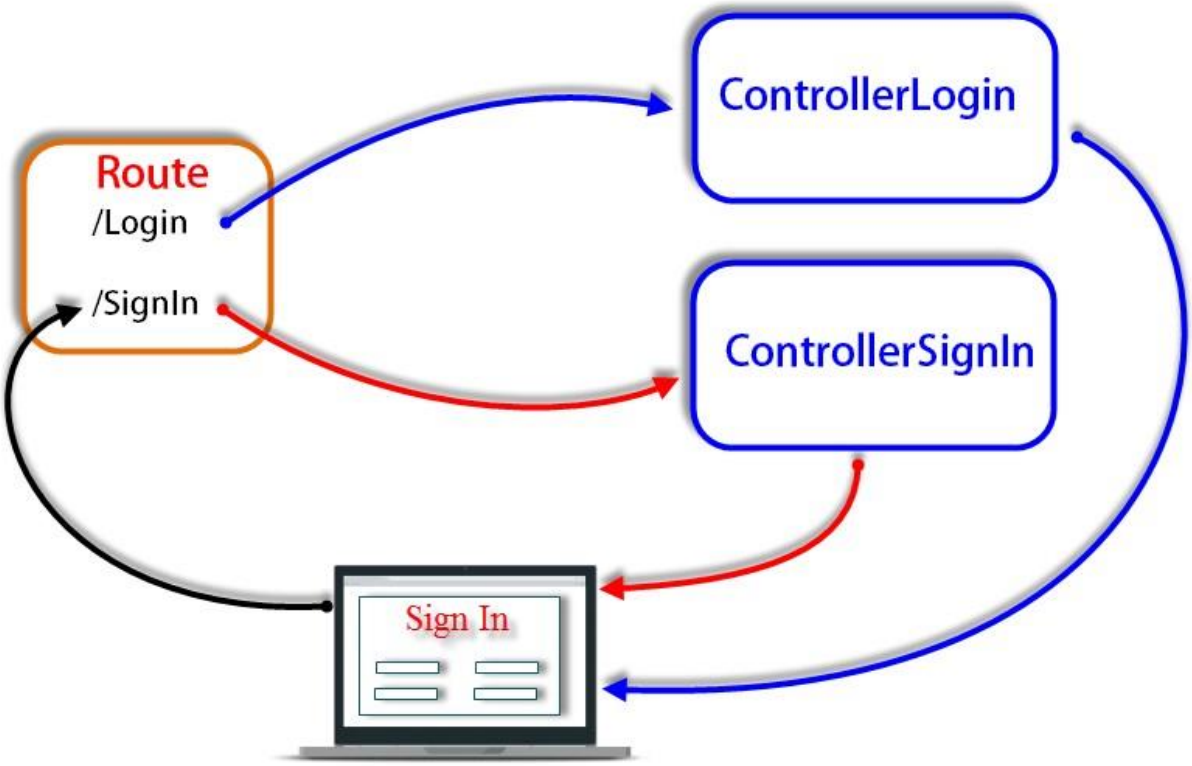
```
Route::group([ 'prefix' => 'MyGroup' ] , function(){
                    //Gọi Route User1:   domain/MyGroup/User1
                    Route::get('User1', function(){  return 'User1'  });

                    //Gọi Route User2:   domain/MyGroup/User2
                    Route::get('User2', function(){  return 'User2'  });

                    //Gọi Route User3:   domain/MyGroup/User3
                    Route::get('User3', function(){  return 'User3'  });
            });
```

CONTROLLER

1. **Cấu trúc Controller**
2. **Tạo Controller**
3. **Gọi Controller từ Route**
4. **Gửi nhận dữ liệu từ Route sang Controller**



1. Cấu trúc Controller

Các Controller sẽ được lưu tại thư mục *App/Http/Controllers* trong Laravel.

```
namespace App\Http\Controllers;

class MyController extends Controller
{
        //Thực hiện các công việc }
```

2. Tạo Controller

Tạo Controller với cmd:

```
php artisan make:controller MyController
```

3. Gọi Controller

Để gọi một hàm trong Controller ta sẽ phải thông qua Route bằng cách khai báo như sau:

Gọi	Nhận dữ liệu bên Controller

Route::get('goiController/{id}', 'MyController@GetData');	class MyController extends Controller
	{
	public function GetData($id)
	{
	echo "Xin Chào !";
	}
	}

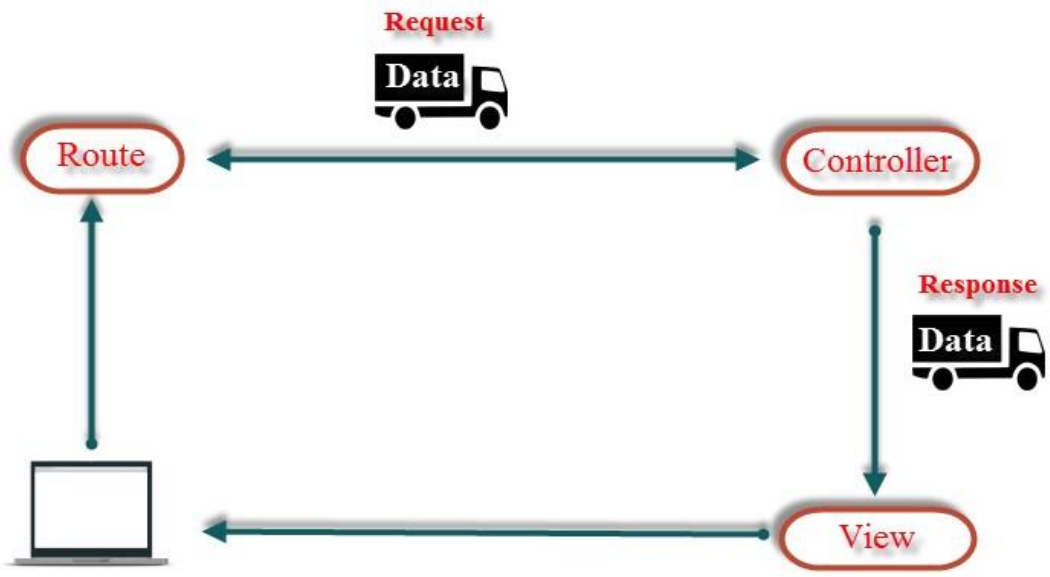
4. Nhận dữ liệu từ Route

Truyền tham số id từ Route	Nhận dữ liệu bên Controller

Route::get('goiController/{id}', 'MyController@GetData');	class MyController extends Controller
	{
	public function GetData($id)
	{
	echo "Đã nhận ". $id;
	}
	}

Gửi nhận dữ liệu với Request và Responses

1. Làm việc với URL
2. Gửi nhận tham số trên Request
3. Sử dụng Cookie với Request và Response
4. Files Upload



1. Làm việc với URL

Cài đặt Route	Làm việc bên Controller

Route::get('goi/Controller', 'MyController@GetData');	use Illuminate\Http\Request;
	use App\Http\Requests;
	
	class MyController extends Controller
	{
	public function GetData(Request $request)
	{
	echo $request->path();
	}
	}

Kết quả : *goi/Controller*

Mở rộng

Phương thức	Chức năng
$request->url();	Trả về một URL đầy đủ.
$request->is('admin/*');	Kiểm tra URL có chứa chuỗi 'admin/' hay không.
$request->isMethod('post');	Kiểm tra phương thức truyền.

2. Gửi nhận tham số trên Request

Cài đặt Route

```
Route::get('getForm',function(){
return view('welcome');
});

Route::post('postForm',[
   'as'=>'postForm',
   'uses'=>'MyController@postForm'
]);
```

Resources/views/welcome.blade.php

```
<form action= "{{route("postForm")}}" method= "post" >
   <input type= "text" name = "name" >
   <input type= "submit" >
</form>
```

Nhận dữ liệu bên Controller

```
use Illuminate\Http\Request;

class MyController extends Controller
{
   //trả về tham số được truyền trên request
public function postForm(Request $request)
   {
         echo $request->name;
   }
}
```

Kết quả : *Hiển thị ra tên bạn vừa nhập vào form*

Mở rộng

Phương thức	Chức năng
$request->has('name');	Kiểm tra tham số name có tồn tại không.
$request->input('id');	Nhận dữ liệu từ thẻ <input name='id' >
$request->input('products.0.name');	Nhận dữ liệu từ mảng
$request->input('user.name');	Nhận dữ liệu từ JSON dạng mảng
$request->all();	Nhận hết dữ liệu, lưu thành dạng mảng.
$request->only('age');	Chỉ nhận tham số age
$request->except('age');	Nhận tất cả tham số ngoại trừ tham số age

3. Sử dụng Cookie với request và response

Cài đặt Route	Nhận dữ liệu bên Controller

//gọi hàm đặt Cookie
Route::get('setCookie','MyController@setCookie');
 	
//gọi hàm hiển thị Cookie
	use Illuminate\Http\Request; use Illuminate\Http\Response; class MyController extends Controller
	{

Route::get('getCookie','MyController@getCookie');	// đặt Cookie
	public function setCookie()
	{
	$response = new Response;
	$response->withCookie(
	'hoten', //tên Cookie
	'Laravel Khoa Pham', //giá trị
	0.1 //minutes - phút
	);
	return $response;
	}
	//hiển thị Cookie
	public function getCookie(Request $request)
	{
	return $request->cookie('hoten');
	}
	}

Kết quả: *Hiển thị ra : Laravel Khoa Pham*

4. Files Uploaded

Cài đặt Route	Nhận dữ liệu bên Controller

Route::get('uploadFile',function()
{
return view('welcome');
});
 	
Route::post('postFile',[
'as'=>'postFile',
'uses'=>'MyController@postFile'
]);
	use Illuminate\Http\Request;
	
	class MyController extends Controller
	{
	public function postFile(Request $request)
	{
	//kiểm tra có tồn tại myFikle ?
	if($request->hasFile('myFile'))
	{
	//lưu file

Resources/views/welcome.blade.php	

```
<form action="{{route("postFile")}}"
method="post"
        enctype="multipart/form-data" >

   <input type="file" name="myFile" id="myFile" >
   <input type="submit" >

</form>
```

```
        $request->file('myFile')->move(
                'images',       //nơi cần lưu
                'Saved.png'     //tên file
                );
    }
else
    {
        echo "Chưa có file";
    }

  }
}
```

Kết quả: *file được lưu tại D:\ProgramFiles\Saved.png*

Mở rộng

Phương thức	Chức năng
getClientSize('myFile')	Trả về dung lượng của file , tính theo bytes
getClientMimeType('myFile')	Trả về kiểu của file : *image/png*
getClientOriginalName('myFile')	Trả về tên của file
getClientOriginalExtension('myFile')	Trả về đuôi của file : *png*
isValid('myFile')	Kiểm tra upload file có thành công hay không

5. Trả về dữ liệu dạng JSON

Cài đặt Route	Nhận dữ liệu bên Controller

//gọi hàm đặt Cookie
Route::get('setJson','MyController@getJson');
	use Illuminate\Http\Response; class MyController extends Controller
	{
	public function getJson()
	{
	return response()->json([
	'name' => 'Khoa Pham',
	'khoahoc' => 'Laravel'
	]);
	}
	}

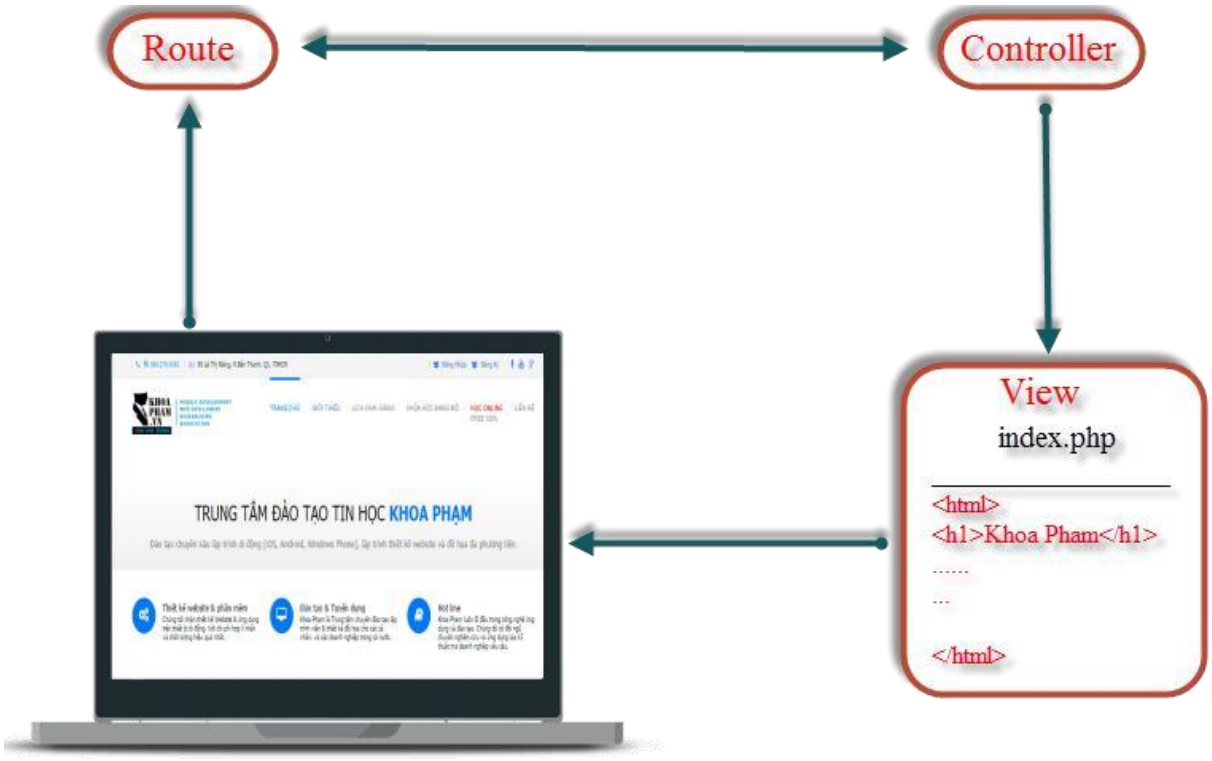
Kết quả: *{"name":"Khoa Pham","khoahoc":"Laravel"}*

Views

1. **View là gì ?**
2. **Gọi view**
3. **Truyền tham số trên view 4. Dùng chung dữ liệu trên view**

1. View là gì ?

Là các file có đuôi .php, chứa mã nguồn html, hiển thị dữ liệu cho người dùng xem và được lưu tại thư mục *resources/views* trong Laravel.



2. Gọi view

Cài đặt Route	Resources/views/myView.php

Route::get('myView', function(){
return view('myView');
});
	Here is my view

Kết quả : Here is my view

Kiểm tra View có tồn tại không ta dùng view()->exists('TenView');

```
If( view()->exists('Layouts.Home') )

{

        // resource/views/Layouts/Home.php có tồn tại

}
```

3. Truyền tham số sang view

Cài đặt Route	Resources/views/myView.php

Route::get('myView/{ten}', function($ten){
return view('myView',['ten'=>$ten]);
});
	<?php
	echo $ten;
	?>

Kết quả : giá trị truyền trên route

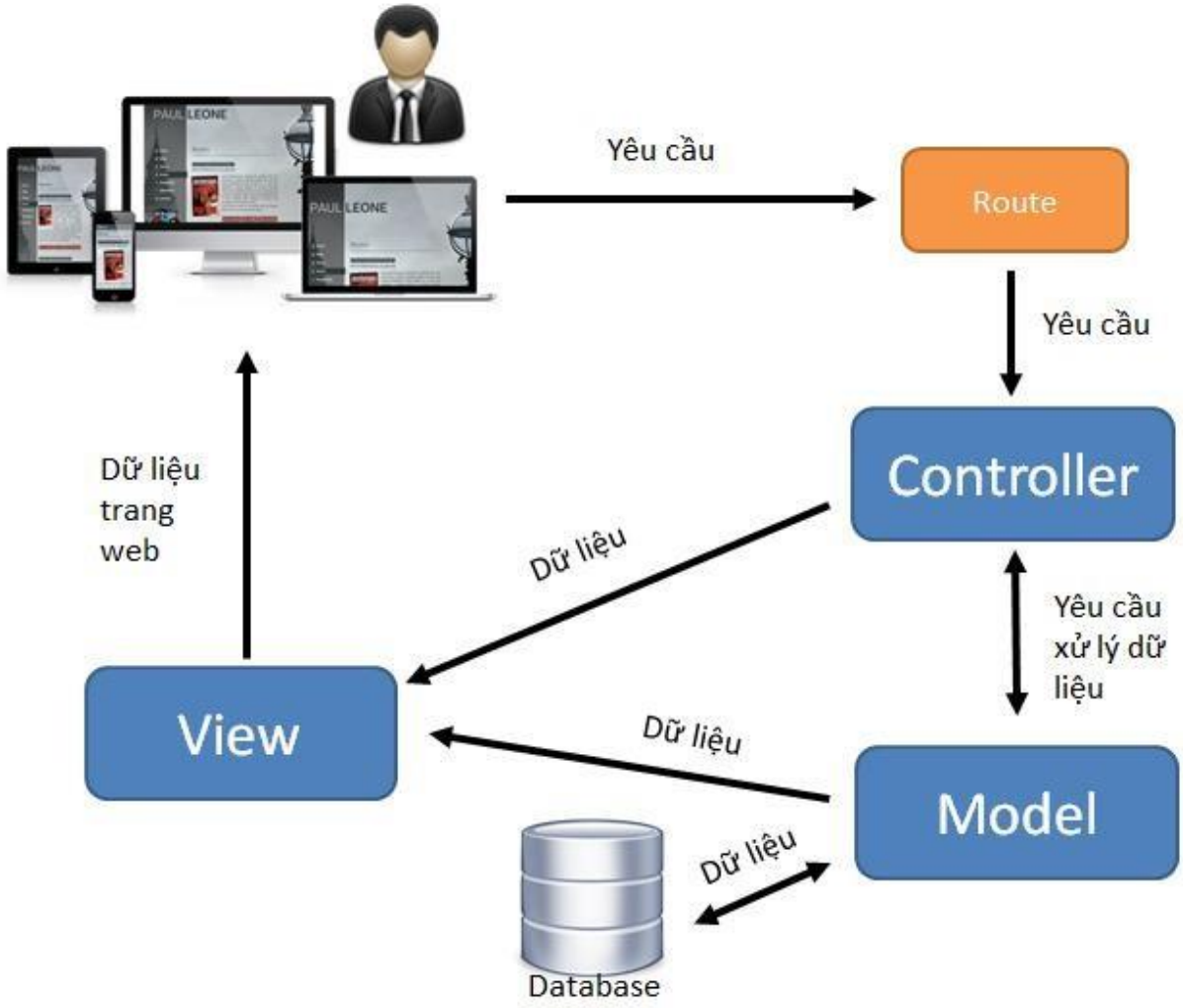
4. Dùng chung dữ liệu trên views

Cài đặt Route	Resources/views/myView.php

View::share('name','Laravel-KhoaPham');
Route::get('myView', function(){
return view('myView');
});
	<?php
	echo $name; ?>

Kết quả : hiển thị dòng chữ Laravel-KhoaPham

Mô hình : Model View Controller



MVC - viết tắt của Model View Controller. Đây là một mô hình, cho phép chúng ta tách biệt các thành phần xử lý trong hệ thống. Để từ đó chúng ta có thể giải quyết các công việc một các nhanh gọn và dễ dàng hơn.

1. **Controller** : có chức năng điều khiển, sắp xếp và xử lý các yêu cầu của người dùng.
2. **Model** : sẽ đảm nhiệm các công việc trao đổi dữ liệu với database.
3. **View** : là thành phần giao diện, hiển thị dữ liệu cho người dùng.

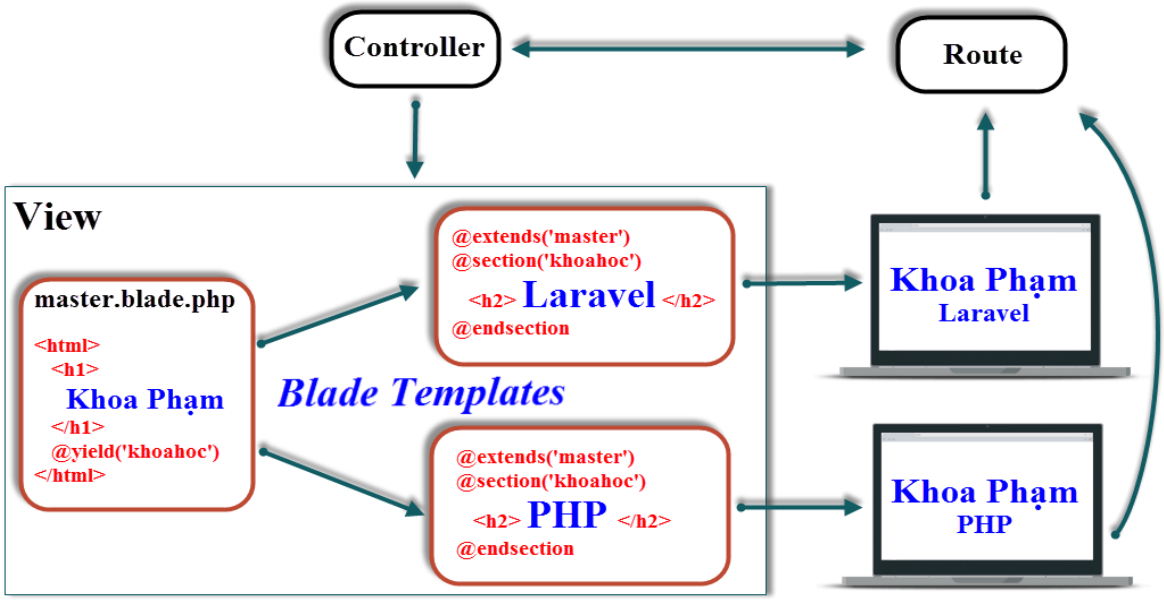
Như chúng ta thấy ở trên hình vẽ, khi người dùng gửi một yêu cầu lên hệ thống, hệ thống sẽ gửi về cho phía Controller xử lý các yêu cầu của người dùng. Trong quá trình làm việc, Controller sẽ phải thông

qua lớp Model để làm việc với CSDL. Sau khi xử lý xong công việc, Controller sẽ đưa sang Views để hiển thị cho người dùng.

Blade Template

1. **Blade Templates là gì ?**
2. **Sử dụng Blade Templates.**
3. **Các câu lệnh điều kiện.**

1. Blade Templates là gì ?



Muốn sử dụng Blade Template thì các tên file phải có chứa .blade đằng trước .php

MyFile.**blade**.php

2. Sử dụng Blade Templates

2.1 Kế thừa

Master.blade.php	Home.blade.php

<table>
<tr><td>

```
<html>
  <h1> HỌC LẬP TRÌNH </h1>
  @yield('NoiDung')

  @include('Footer')
</html>
```

</td><td rowspan="3">

```
@extends('Master')

@section('NoiDung')

<a href="khoapham.vn" > Laravel </a>

@endsection
```

</td></tr>
<tr><td>Footer.blade.php</td></tr>
<tr><td>

```
<h2> Khoa Phạm </h2>
```

</td></tr>
</table>

2.2 Hiển thị dữ liệu

Chức năng	Mã lệnh	Kết quả
In giá trị của biến	<?php $str = 'Học lập trình Laravel'; ?> {!! $str !!}	*Học lập trình Laravel*
In giá trị của biến	<?php $str = 'Học lập trình Laravel'; ?> {{ $str }}	Học lập trình Laravel
Đánh dấu Comments	{-- đánh dấu Comment --}	

3. Các câu lệnh điều kiện

Lệnh	Mã lệnh
Kiểm tra điều kiện với if() - else	**@if($dieukien)** **@elseif($dieukien2)** **@else** **@endif**
Vòng lặp for()	**@for($i = 0; $i < 10 ; $i++)** **@endfor**
Vòng lặp foreach()	**@foreach($users as $user)** **@endforeach**

Vòng lặp forelse()	@forelse($users as $user) //$users không rỗng @empty //$users rỗng @endforelse
Vòng lặp while()	@while($dieukien) @endwhile
Bỏ qua vòng lặp với continute	@continue @continue($dieukien) //thực hiện khi có điều kiện
Thoát khỏi vòng lặp với break	@break @break($dieukien) //thực hiện khi có điều kiện

Làm việc với Database

1. **Schema**
2. **Migrate**
3. **Seed**
4. **Query Builder**
5. **Eloquent - Model**
6. **Liên kết dữ liệu trong Laravel**

Kết nối với cơ sở dữ liệu trong laravel

Mở file .env
DB_HOST=localhost DB_DATABASE= Ten CSDL DB_USERNAME= Ten nguoi dung DB_PASSWORD= Mat khau

1. Schema

1.1 Tạo bảng

```
Schema::create('SanPham', function ($table) {
        $table->increments('id');          //Tự tăng, khóa chính
        $table->string('TenSanPham');  //Kiểu chuỗi
        $table->integer('Gia');              //Kiểu int
        $table->timestamps();              //Tự cập nhật thời gian
    });
```

Mở rộng

Câu lệnh	Mô tả
$table->primary('TenKhoaChinh');	Tạo khóa chính
$table->foreign('KhoaPhu')->references('KhoaChinh')->on('Bang');	Tạo khóa phụ
$table->unique('TênCột');	Rang buộc unique
$table->time();	Kiểu giờ
$table->dateTime();	Kiểu ngày, giờ
$table->date();	Kiểu ngày
$table->text();	Kiểu text
$table->float();	Kiểu float
$table->boolean();	Kiểu logic
$table->rememberToken();	Tạo Token

Điều kiện

Câu lệnh	Mô tả
->nullable();	Cho phép giá trị null
->default($value);	Gán giá trị mặc định cho cột
->unsigned();	Đặt unsigned cho integer

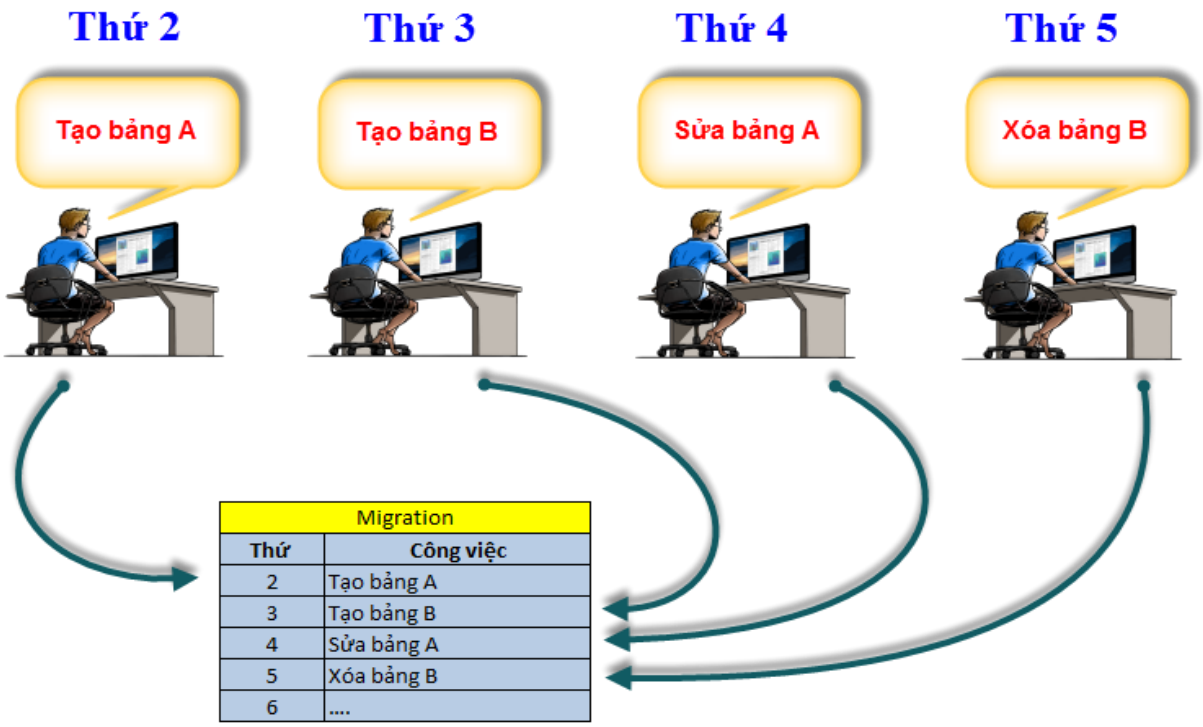
1.2 Sửa bảng

Câu lệnh	Mô tả
$table->dropColumn('TenCot');	Xóa cột trong bảng
Schema::rename($from, $to);	Đổi tên bảng

1.3 Xóa bảngss

Câu lệnh	Mô tả
Schema::drop('users');	Xóa bảng users
Schema::dropIfExists('users');	Xóa bảng users nếu bảng tồn tại

2. Migrate



Migrate dùng để tạo lên cấu trúc các bảng trong cơ sở dữ liệu. Ta có thể sử dụng migrate để tạo ra các bảng cũng như back up, restore lại theo ý muốn.

Các file migrate sẽ được lưu tại **database/migrations/**

Sử dụng migrate với cửa sổ cmd

php artisan make:migration TenMigrate	Tạo file migrate với artisan
php artisan migrate	Thực thi file migrate
php artisan migrate:rollback	Hủy bỏ việc thực thi của migrate trước
php artisan migrate:reset	Hủy bỏ hết công việc của migrate

Option

--create=TenBang	Migrate tạo bảng
--table=TenBang	Migrate chỉnh sửa bảng

Cấu trúc migration

```
use Illuminate\Database\Schema\Blueprint; use Illuminate\Database\Migrations\Migration;
class CreateTable extends Migration
{
   public function up()
   {
       //đoạn lệnh khi thực hiện migrate
   }

   public function down()
   {
       //đoạn lệnh thực hiện khi Rollback.
   }
}
```

Tạo bảng với Schema

```
public function up()
{
        Schema::create('SanPham', function (Blueprint $table) {
           $table->increments('id');          //Tự tăng, khóa chính
           $table->string('TenSanPham');  //Kiểu chuỗi
           $table->integer('Gia');             //Kiểu int
           $table->timestamps();               //Tự cập nhật thời gian
        }); }
```

3. Seed

Seed là bộ dữ dữ liệu mẫu, nó giúp chúng ta quản lý dữ liệu trong bảng một cách thuận tiện, dễ dàng khôi phục lại khi cần thiết.

Các file seed được lưu tại thư mục **database/seeds/**

Tạo dữ liệu mẫu trong Seed.	Thực thi Seed.

use Illuminate\Database\Seeder; use Illuminate\Database\Eloquent\Model;
class DatabaseSeeder extends Seeder
{
public function run()
{
DB::table('users')->insert([
'name' => str_random(10),
'email' => str_random(10).'@gmail.com',
'password' => bcrypt('secret'),
]);
}
}
	Mở cửa sổ cmd : php artisan db:seed

4. Query Builder

Có tác dụng thay thế cho các câu lệnh truy vấn thông thường bằng các phương trong lớp DB.

Ví dụ : **$users = DB::table('users')->get();** sẽ lấy toàn bộ dữ liệu trong bảng users ra và lưu vào $users

Lệnh này sẽ tương đương với lệnh truy vấn thông thường : ***SELECT * FROM users***

Các lệnh truy vấn

Lệnh truy vấn	Mô tả	Ví dụ
DB::table('users')	Chọn bảng trong cơ sở dữ liệu	DB::table('users')->get();
get()	Lấy dữ liệu trong bảng	DB::table('users')->get();
first()	Lấy một dòng dữ liệu đầu tiên từ kết quả truy vấn	DB::table('users')-> where('name', 'John')->first();
value('tên cột')	Trả về dữ liệu của cột đã khai báo	DB::table('users')-> where('name', 'Joh')-> value('email');
select('tên cột 1')	Chọn tên cột cần truy vấn	DB::table('users')-> select('name', 'email')->get();

addSelect('tên cột')	Thêm cột vào truy vấn trước đó với addSelect()	$query = DB::table('users') ->select('name'); $users = $query-> addSelect('age')->get();
DB::raw('Truy vấn')	Thêm lệnh truy vấn vào select()	DB::table('users')-> select(DB::raw('count(*) as userCount, status'))
join('bảng liên kết', 'cột liên kết 1', 'điều kiện', 'cột liên kết 2')	Lệnh Join bảng trong truy vấn	DB::table('users')-> join('contacts', 'users.id', '=', 'contacts.user_id')-> select('contacts.phone')->get();
where('cột 1', 'điều kiện' , giá trị)	Điều kiện where	DB::table('users')-> where('votes', '=', 100)->get();
orwhere('cột 1', 'điều kiện' , giá trị)	Điều kiện hoặc	DB::table('users')-> where('votes', '=', 100)-> orwhere('age', '>=', '18')->get();
orderBy('tên cột', 'điều kiện')	Lệnh orderBy	DB::table('users')-> orderBy('name', 'desc')->get();
groupBy('tên cột')-> having(điều kiện)	Lệnh groupBy	DB::table('users')-> groupBy('account_id')-> having('account_id', '>', 100)->get();
skip(vị trí)-> take(số lượng)	Giới hạn kết quả truy vấn Tương đương với LIMIT	DB::table('users')->skip(10)->take(5) ->get();
avg('tên cột');	Lấy giá trị trung bình	DB::table('orders')-> where('finalized', 1)->avg('price');
max('price');	Lấy giá trị max	DB::table('orders')->max('price');
count();	Lệnh đếm	DB::table('users')->count();

Lệnh update

Lệnh truy vấn	Mô tả	Ví dụ
update(['tên cột' => giá trị]);	Lệnh update	DB::table('users')->where('id', 1)-> update(['votes' => 1]);
increment('tên cột',giá trị) decrement('tên cột',giá trị)	Tăng/giảm giá trị cột	DB::table('users')-> increment('votes',4);

Lệnh insert

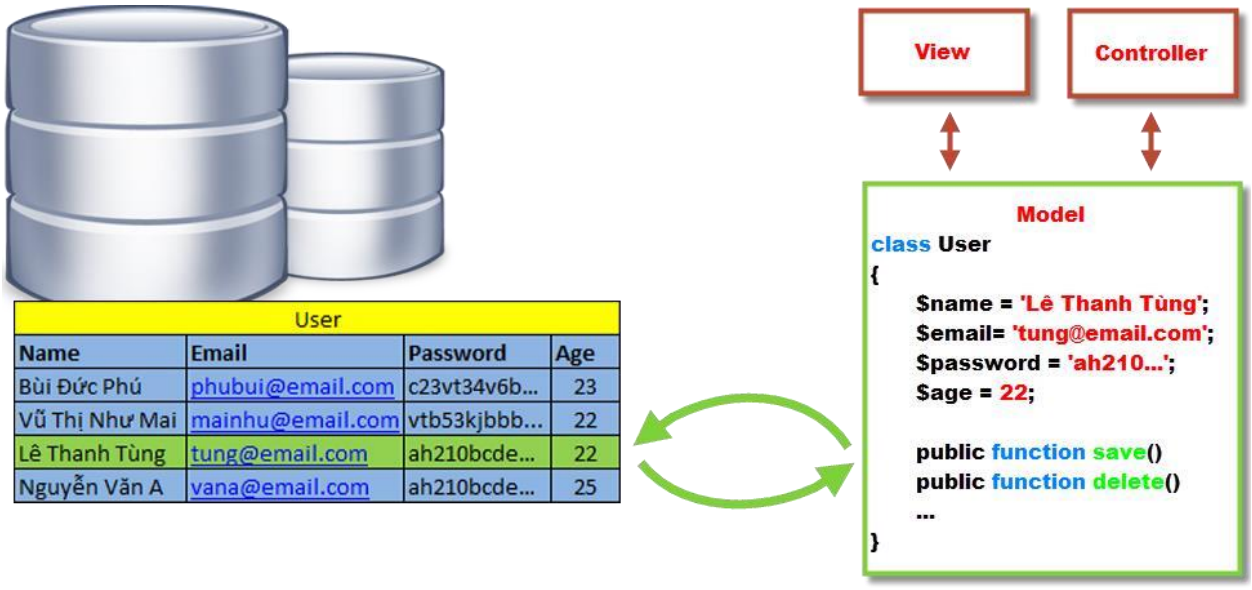
Lệnh truy vấn	Mô tả	Ví dụ
insert([mảng các bản ghi]);	Lệnh insert	DB::table('users')->insert(['email' => 'john@example.com', 'votes' => 0]);

Lệnh delete

Lệnh truy vấn	Mô tả	Ví dụ
delete();	Xóa dữ liệu	DB::table('users')->where('votes', '<', 100)->delete();
truncate();	Xóa tất cả dữ liệu trong bảng và đặt chỉ số tự tăng về 0	DB::table('users')->truncate();

5. Eloquent – Model

Model là một lớp dữ liệu, có cấu trúc giống với bảng trong cơ sở dữ liệu, dùng để xử lý dữ liệu ra vào trong bảng.



User			
Name	Email	Password	Age
Bùi Đức Phú	phubui@email.com	c23vt34v6b...	23
Vũ Thị Như Mai	mainhu@email.com	vtb53kjbbb...	22
Lê Thanh Tùng	tung@email.com	ah210bcde...	22
Nguyễn Văn A	vana@email.com	ah210bcde...	25

5.1 Tạo model

Các file model sẽ được lưu tại thư mục App/

Tạo một model :

```
php artisan make:model TenModel
```

Tạo một model và migrate tương ứng với nó :

```
Php artisan make:model TenModel -m
```

Kết nối Model tới bảng trong cơ sở dữ liệu

Mã lệnh	Mô tả
protected $table = 'tên bảng';	Kết nối model với bảng trong cơ sở dữ liệu
public $timestamps = false;	Tắt/bật chế độ tự động quản lý 'created_at' và 'update_at'

Ví dụ

```
namespace App;
use Illuminate\Database\Eloquent\Model;

class User extends Model
{
    protected $table = 'user';
public $timestamps = false;
}
```

5.2 Các phương thức trong model

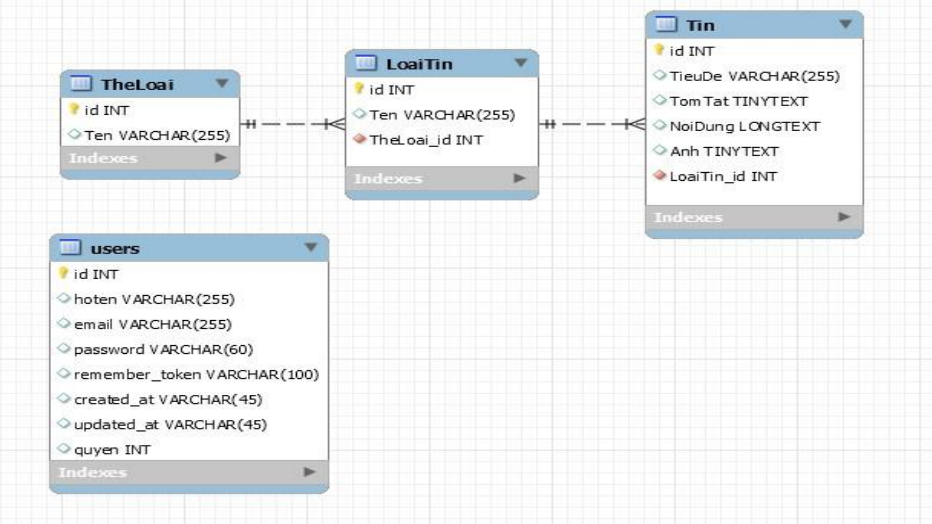
Một số phương thức hay sử dụng trong model

Mã lệnh	Mô tả
$user = new User(); echo $user->name;	Lấy giá trị thuộc tính của model
$user = User::all();	Lấy toàn bộ dữ liệu trong bảng
$user = User::find(giá trị khóa chính);	Tìm user theo khóa chính
$user->toJson();	Trả dữ liệu kiểu JSON
$user->save();	Lưu dữ liệu từ model vào bảng
$user->delete();	Xóa dữ liệu trong bảng
User::destroy(giá trị khóa chính);	Xóa dữ liệu bằng khóa chính trong bảng

Kết hợp model với query builder

```
$user = User::where('active', 1)->orderBy('name', 'desc')->take(10)->get();
```

6. Liên kết dữ liệu



Model là đại diện cho các bảng trong cơ sở dữ liệu, chính vì thế mà nó cũng có các liên kết với nhau.

Khai báo các liên kết tới các model khác.

Ví dụ : Liên kết một nhiều. Ta khai báo hàm TenLienKet() trong class model.

Khai báo	Sử dụng

`public function TenLienKet()`
`{`
`  return $this->hasMany( 'TenModel' , 'KhoaPhu', 'KhoaChinh' );`
`}`
	`TenModel::TenLienKet()`

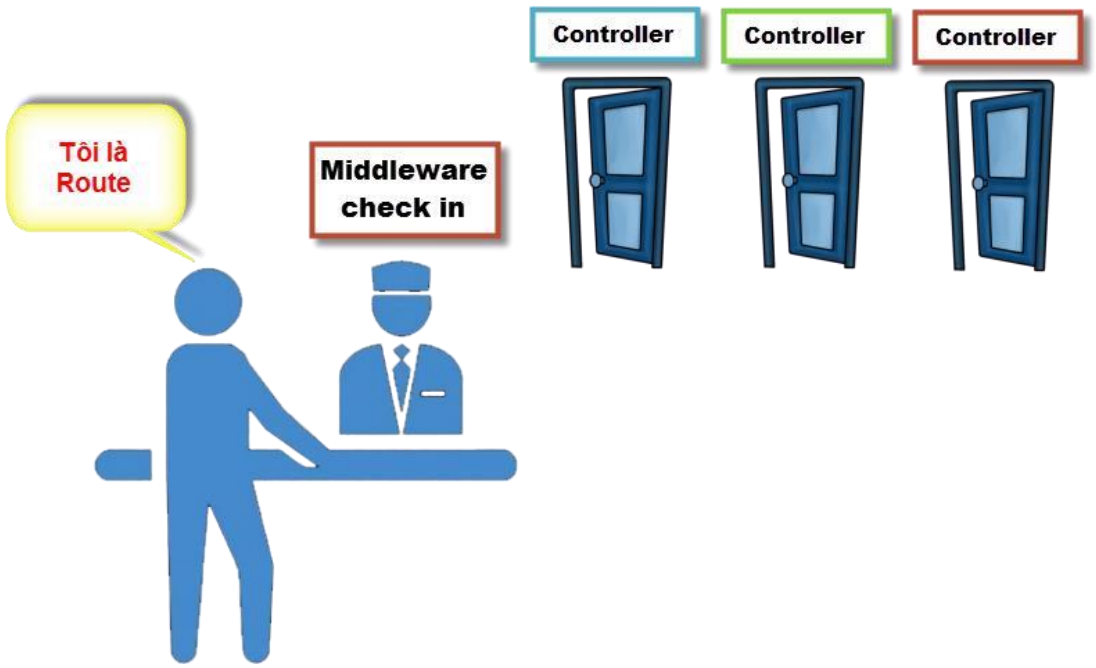
Bảng liên kết

Liên kết	Hàm liên kết
Một – Một , Liên kết từ bảng cha tới bảng con.	**hasOne();**
Một – Một , Liên kết từ bảng con tới bảng cha.	**belongsTo();**
Một – Nhiều	**hasMany();**
Nhiều – nhiều	**belongsToMany();**

Liên kết qua bảng trung gian	**hasManyThrough();**

Kiểm soát Route với Middleware

1. Kiểm soát Route với middleware



2. Làm việc với middleware

2.1 Cấu trúc của middleware

```
public function handle($request, Closure $next)
   {
      if ($request->input('age') <= 200) {          return redirect('home');
      }
      return $next($request);
   }
```

2.2 Tạo middleware mới cửa sổ cmd

```
php artisan make:middleware TenMiddleware
```

2.3 Đăng ký Middleware cho routes

Mở app/Http/Kernel.php

```
protected $routeMiddleware = [
    'auth' => \App\Http\Middleware\Authenticate::class,
    'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
    'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
    'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,

    'Định danh Middleware' => \App\Http\Middleware\TenMiddleware::class,
];
```

2.4 Gán middleware cho route

Cách 1

```
Route::get('user',[
                    'middleware' => 'Định danh',
function () { }  ]);
```

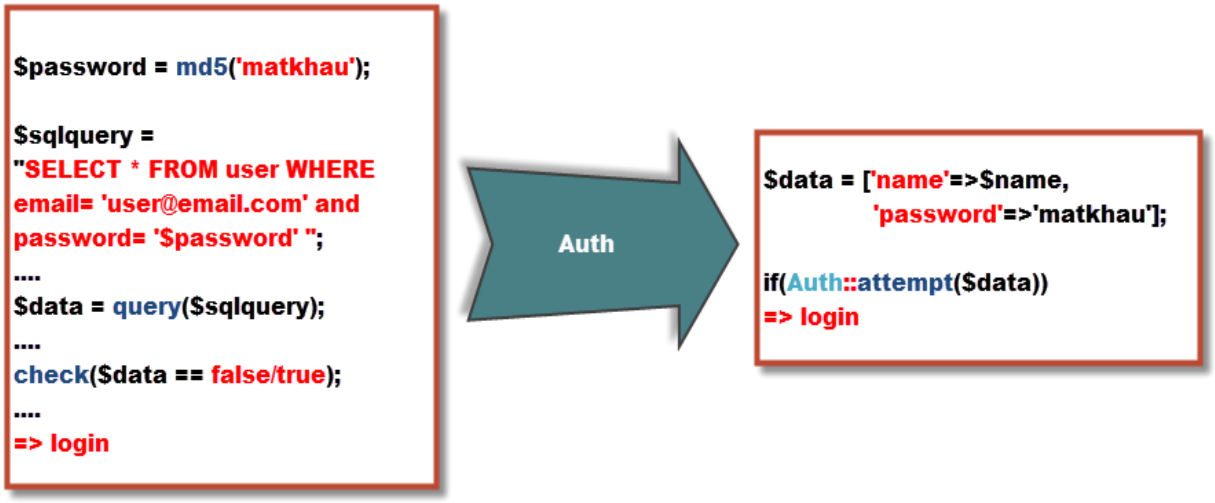
Cách 2

```
Route::get('/', function () {

})->middleware(['Định danh1', 'Định danh2']);
```

Auth - Authenticate

1. Auth là gì



2. Các hàm trong Auth

Để sử dụng Auth bạn phải thêm thư viện

```
use Illuminate\Support\Facades\Auth;
```

Sử dụng Auth

```
Auth::login($user);
```

Phương thức	Chức năng
attempt(array())	Đăng nhập với thông tin đăng nhập
login(model)	Đăng nhập với đối tượng
logout()	Hủy đăng nhập
user()	Lấy thông tin người đang đăng nhập
check()	Kiểm tra đăng nhập chưa

Session

1. **Cấu hình session**
2. **Các hàm trong session**

1. Cấu hình

Cấu hình session trong **config/session.php**

Tùy chỉnh trong config/session.php	Mô tả
'lifetime' => 120,	Thời gian tồn tại của session tính theo phút
'expire_on_close' => false,	true : mất session khi đóng trình duyệt, false : ngược lại
'driver' => env('SESSION_DRIVER', 'file'),	Chọn nơi lưu trữ session

Để sử dụng được session thì ta phải đặt routes trong middleware web

```
Route::group(['middleware' => 'web'], function () {
Route::get('Route-Session', function(){          session()->put('KhoaHoc','Laravel');
        echo session('KhoaHoc');
  });
});
```

2. Các hàm trong Session

Sử dụng Session

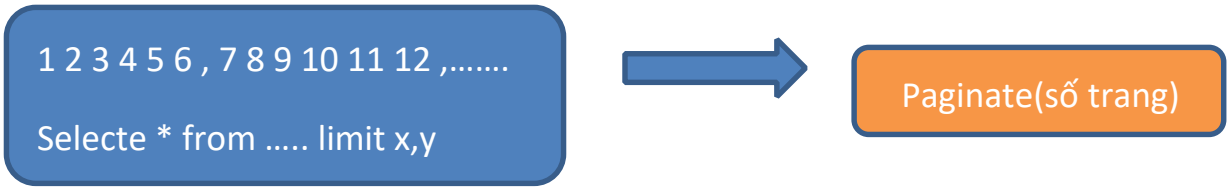
```
Session::PhuongThuc;
session()->PhuongThuc;
```

Phương thức trong Session

Phương thức	Chức năng
put('Ten','GiaTri');	Khai báo session
flash('flasha','Đây là session flash');	Khai báo flash
session('Ten');	Lấy dữ liệu từ session
has('Ten')	Kiểm tra session tồn tại không
forget('Ten');	Xóa bỏ 1 session
flush();	Xóa hết các session

Pagination

1. Sử dụng phân trang
2. Cấu trúc thẻ phân trang



1. Khai báo

ControllerPhanTrang	PhanTrang.blade.php
$sanpham = sanpham::paginate(5); return view('phantrang',['sanpham'=>$sanpham]);	{!! $sanpham->links() !!}

Sử dụng Paginate

```
sanpham::paginate(5);
```

Phương thức	Chức năng
paginate(Số tin trong 1 trang);	Phân trang
simplePaginate(Số tin trong 1 trang);	Phân trang không có số
paginate(5)->setPath('myurl/sanpham');	Chỉnh đường dẫn phân trang
{!! $sanpham->links() !!}	Hiển thị phân trang trên view
{!! $sanpham->appends(['sort' => 'votes'])->links() !!}	Thêm biến vào đường dẫn phân trang
{!! $sanpham->fragment('foo')->links() !!}	Thêm fragment vào sau đường dẫn

2. Cấu trúc thẻ phân trang

Cấu trúc thẻ phân trang paginate();

```
<ul class = "pagination" >
        <li class= "disabled" ><span>«</span></li>
        <li class= "active" ><span>1</span></li>
        <li><a href= "view?page=2" >2</a></li>
        <li><a href= "view?page=3" >3</a></li>
        <li><a href= "view?page=2" rel="next">»</a></li>
</ul>
```

Cấu trúc thẻ phân trang simplePaginate();

```
<ul class = "pager" >
        <li><a href = "http://yourhost/view?page=4" rel= "prev">«</a>li>
        <li><a href = "http://yourhost/ view?page=6" rel= "next">»</a></li> </ul>
```