Mảng:

$nam = array(

1990,

1991,

1992,

1993,

1994,

1995

);

//Dùng foreach xuất ra các năm trong $nam, gồm cả chỉ số và giá trị

foreach ($nam as $i => $giatri){

echo $i . ' => ' . $giatri . '<br/>';

}

//Dùng foreach xuất ra các năm trong $nam, chỉ có giá trị

foreach ($nam as $giatri){

echo $giatri . '<br/>';

}

**POST và GET:**

<form method="POST">

Username: <input type="text" name="username" value=""/> <br/>

password: <input type="password" name="password" value=""/><br/>

<input type="submit" name="form\_click" value="Gửi Dữ Liệu"/>

<?php

if(isset($\_POST['form\_click'])){

echo 'Tên đăng nhập: '. $\_POST['username'];

echo '<br/>';

echo 'Mật khẩu: ' . $\_POST['password'] ;

}

echo "Nam nói\"Cậu ấy đang ăn tối\" ";

?>

</form>

**Các hàm xử lý chuỗi hay sử dụng**

**addcslashes ($str, $char\_list)**

Hàm này sẽ thêm dấu gạch chéo (\) đằng trước những ký tự trong chuỗi $str mà ta liệt kê ở $char\_list.

**addslashes ( $str )**

Hàm này sẽ thêm dấu gách chéo trước những ký tự (‘, “, \) trong chuỗi $str.

**stripslashes ($str)**

Hàm này ngược với hàm addslashes, nó xóa các ký tự \ trong chuỗi $str.

**explode ( $delimiter , $string)**

Hàm này sẽ chuyển một chuỗi **$string** thành một mảng các phần tử với ký tự tách mảng là $**delimiter.**

**implode($delimiter, $piecesarray);**

Hàm này ngược với hàm explode, nó chuyển một mảng $piecesarray thành chuỗi và mỗi phần tử cách nhau bởi chuỗi $delimiter

**ord ( $string )**

Hàm này trả về mã ASCII của ký tự đầu tiên trong chuỗi $string.

**strlen($string)**

Hàm này đếm số ký tự của chuỗi $string.

**str\_word\_count($str)**

Hàm này trả về số từ trong chuỗi $str.

**str\_repeat(  $str,  int $n  )**

Hàm này lặp chuỗi $str $n lần.

**str\_replace( $chuoi\_tim, $chuoi\_thay\_the, $chuoi\_nguon )**

Hàm này tìm kiếm và thay thế chuỗi.

**md5( $str)**

Hàm này mã hóa chuỗi thành một dãy 32 ký tự (mã hóa md5).

**sha1($string)**

Hàm này mã hóa chuỗi thành một dãy 40 ký tự (mã hóa sha1)

**substr( $string,  $start, $length )**

Hàm này lấy một chuỗi con nằm trong chuỗi $str bắt đầu từ ký tự thứ $start và chiều dài $length.

echo substr( 'freetuts.net',  0, 8);

// Kết quả freetuts

**strstr( $string, $ky\_tu\_cho\_truoc )**

Tách một chuỗi bắt đầu từ  $ky\_tu\_cho\_truoc cho đến hết chuỗi.

echo strstr('freetuts.net Xin Chào', 'Xin');

// Kết quả: Xin Chào

**strpos($str, $chuoi\_tim )**

Tìm vị trí của chuỗi $chuoi\_tim trong chuỗi $str, kết quả trả về false nếu không tìm thấy.

echo strpos('freetuts.net chào các bạn', 'chào');

// kết quả 13

**strtolower($str);**

Chuyển tất cả các ký tự chuỗi $str sang chữ thường

**strtoupper($string );**

Chuyển tất cả các ký tự chuỗi $str sang chữ hoa

**ucfirst( $string )**

Chuyển ký tự đầu tiên chuỗi $string sang chữ hoa

**ucfirst( $string )**

Chuyển ký tự đầu tiên trong chuỗi $string sang chữ thường

**ucwords( $string )**

Chuyên từ đầu tiên trong chuỗi $string sang chữ hoa

**trim($string, $ky\_tu);**

Xóa ký tự $ky\_tu nằm ở đầu và cuối chuỗi $str, nếu ta không nhập $ky\_tu thì mặc định nó hiểu là xóa khoảng trắng.

echo trim(' freetuts.net ');

// kết quả: 'freetuts.net'

echo trim('freetuts.neth', 'th');

// kết quả: 'freetuts.ne'

**ltrim($string, $ky\_tu);**

Tương tự như trim nhưng chỉ xóa bên trái

**rtrim($string, $ky\_tu)**

Tương tự như trim nhưng chỉ xóa bên phải

**nl2br($string)**

Chuyển các ký tự xuống dòng “\n” thành thẻ

**json\_decode($json, $is\_array)**

Dùng để chuyển chuỗi dạng JSON sang các đối tượng mảng hoặc object. Nếu **$is\_array** có giá trị false thì hàm sẽ chuyển một chuỗi $json thành một Class (object),  ngược lại nếu **$is\_array** có giá trị true thì sẽ chuyển chuỗi $json thành một mảng.

**json\_encode($array\_or\_object)**

Chuyển một mảng hoặc mội đối tượng (classs) sang chuỗi dạng JSON

**LARAVEL**

1. **Tạo dự án laravel**

Chạy lệnh cmd:

composer create-project --prefer-dist laravel/laravel demo01

**Chạy laravel:** php artisan serve

Nếu có lỗi tiến hành update composer: composer update

**Chạy vue js: npm run watch**

# Bài 3: Cấu trúc thư mục của Laravel framework 5.x

Laravel sử dụng mô hình MVC làm nền tảng phát triển. Trong bài này chúng ta sẽ tìm hiểu chức năng và nhiệm vụ của từng thư mục, điều này giúp bạn tiếp cận Laravel 1 cách dễ hiểu hơn là đi ngay vào việc code

#### ****1. Thư mục app****

Thư mục này rất quan trọng và bạn thường xuyên phải thao tác trong thư mục này: Nó chứa các Models, Controller, Providers ...

#### ****2. Thư mục bootstrap****

Chứa các file khởi tạo và một số file cấu hình nạp tự động

#### ****3. Thư mục config****

Chứa các file cấu hình của Laravel Framework: cấu hình database, app, mail, session ...

#### ****4. Thư mục public****

Các file tĩnh như file HTML, CSS, Javascript hay ảnh sẽ được đưa vào thư mục này.

#### ****5 Thư mục resources****

Chứa trang views (V của mô hình MVC), chứa các file khác như (SASS, LESS, CoffeeScript) và các file hỗ trợ cho đa ngôn ngữ..

#### ****6. Thư mục routes (từ laravel 5.2 mới có)****

Chứa file web.php và api.php có vai trò chỉ đường cho yêu cầu (request) đi đến đâu

#### ****7 Thư mục storage****

Chứa các file đã biên dịch từ các file view xài Blade template, chứa file sessions, caches và các file được sinh ra tự động của framework

#### ****8. Thư mục tests****

Chứa các file test của Framework

#### ****9. Thư mục vendor****

Chứa các thư viện mà Composer cài đặt cho Framework

#### Một số file con khác:

**.gitignore:** Git (phần mềm để quản lý sự thay đổi của source code) sử dụng 2 file này để định nghĩa thuộc tính và danh sách các file không theo dõi.

**composer.json:** File này chứa danh sách các dependencies (thư viện, framework) sử dụng trong app. Khi bạn chạy câu lệnh composer install hoặc composer update thì file này sẽ là cơ sở để xác định phiên bản nào cần tải về của các thư viện, framework được liệt kê trong file này.

**artisan**: File này được dùng để thực thi các lệnh CLI (command line interface) để hỗ trợ phát triển ứng dụng.

# Bài 4: Route trong Laravel 5.x

Trong Laravel  Route là nơi nhận request của người dùng (phương thức GET hoặc POST) sau đó có thể chỉ định (định tuyến) 1 action của 1 Controller thực hiện request đó. Khi tiếp cận Laravel ta không thể bỏ qua Route vì nó là tiền đề cho việc xử lý yêu cầu

#### ****Vị trí route trong laravel****

Từ phiên bản 5.1 trở về trước: Route được định nghĩa trong app/routes.php.

Từ phiên bản 5.2: Route được định nghĩa trong thư mục routes, phiên bản 5.3 chia ra 2 file web.php và api.php giúp bạn thêm tùy chọn phát triển web và api riêng

#### ****Một số phương thức có sẵn của Route trong Laravel****

Route::get($uri, $callback);

Route::post($uri, $callback);

Route::put($uri, $callback);

Route::patch($uri, $callback);

Route::delete($uri, $callback);

Route::options($uri, $callback);

Nếu bạn muốn sử dụng chung nhiều phương thức Laravel hỗ trợ bạn phương thức **match**và **any**

Route::match(['get', 'post'], '/', **function** () {

// xử lý phương thức GET và POST

});

Route::any('foo', **function** () {

// xử lý tất cả

});

#### ****Truyền tham số (Route Parameters)****

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

**return** 'User '.$id;

// Tham số được biểu diễn trong dấu ngoặc nhọn

// Ví dụ lấy id của user và show ra màn hình

});

Nếu bạn muốn truyền nhiều tham số:

Route::get('posts/{post}/comments/{comment}', **function** ($postId, $commentId) {

// Truyền 2 tham số {post} và {comment}

});

#### ****Route với Tham số không bắt buộc****

Nếu bạn muốn tham số truyền vào có hay không đều được, bạn có thể đặt dấu ( ? ) ngay sau tên tham số để đánh dấu nó là tham số không bắt buộc:

Route::get('user/{name?}', **function** ($name = null) {

**return** $name;

});

Route::get('user/{name?}', **function** ($name = 'John') {

**return** $name;

});

#### ****Đặt điều kiện cho Route****

Bạn  có thể dùng **Regex**để qui định format cho các tham số ở cùng một vị trí trong URL, bằng cách dùng hàm**where()**  để khai báo.

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

// Route có tham số nhận giá trị chuỗi

})

->where('name', '[A-Za-z]+');

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

// Route có tham số nhận giá trị số

})

->where('id', '[0-9]+');

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

// Route có tham số nhận giá trị chuỗi và số

})

->where(['id' => '[0-9]+', 'name' => '[a-z]+']);

Toi da 3 chu so:

where(['id'=> '[0-9]{3}+']);

#### ****Truyền data từ Router cho view:****

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

$name = "chienbui";

$age = 20;

return view('demo', compact('name', 'age'));

});

#### ****Router gọi Controller:****

Route::get('/home', 'HomeController@showInfo');

Có Controller: HomeController và action: showInfo().

#### ****Đặt tên route trong Laravel****

Route::get('user/profile', **function** () {

// Route có tên là profile

})->name('profile');

Bạn cũng có thể đặt tên và chỉ định 1 action trong Controller

Route::get('user/profile', 'UserController@showProfile')->name('profile');

#### ****Gom nhóm trong Route****

Nếu bạn muốn gom nhiều route vào thành 1 nhóm các thuộc tính chung như middleware hay namespaces, ban có thể sử dụng Route::group

Route::group(['middleware' => 'auth'], **function** () {

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

// Nhóm các user đã được chứng thực

});

Route::get('user/profile', **function** () {

// Nhóm các user đã được chứng thực

});

});

#### ****Tạo tiền tố cho Group****

Nếu bạn muốn nhóm Route có thêm 1 tiền tố phía trước như admin chẳng hạn bạn có thể viết

Route::group(['prefix' => 'admin'], **function** () {

Route::get('users', **function** () {

// Matches The "/admin/users" URL

});

});

## Route::controller trong laravel

Route::controller($uri, $controller, $names);

Route::controller('/controller','HomeController',['getFirst'=>'first','getSecond'=>'second','postThird'=>'third']);

## Route::resource() trong laravel

Route này được laravel tạo ra nhằm mục đích chính là xây dựng RESTful (xây dựng Webservice), route này sẽ đi kèm với một RESTful controller.

**Để tạo 1 RESTful controller thì có 2 cách**:

* Cách 1 là tự tạo bằng tay file controller
* Cách 2: sử dụng dòng lệnh của laravel hỗ trợ
* //Cú pháp sử dụng:
* php artisan controller:make "têncontroller" [tùy chọn]
* **Các action của RESTful controller mà Route::resource hỗ trợ**:

| **Method** | **URI** | **Action** | **Route name** |
| --- | --- | --- | --- |
| GET | /resource | index | resource.index |
| GET | /resource/create | create | resource.create |
| POST | /resource | store | resource.store |
| GET | /resource/{id} | show | resource.show |
| GET | /resource/{id}/edit | edit | resource.edit |
| PUT/PATCH | /resource/{id} | update | resource.update |
| PUT/PATCH | /resource/{id} | destroy | resource.destroy |

Route::resource('photos', 'PhotoController')->except([

'create', 'store', 'update', 'destroy'

]);

## Ghi đè name routes

Mặc định tất cả các route trong resource controller sẽ có tên như trong bảng bên trên. Tuy nhiên bạn có thể ghi đè tên route bằng cách sau đây:

Route::resource('photos', 'PhotoController')->names([

'create' => 'photos.build'

]);

## Route::filter trong laravel

Route::filter($name, $callback);

Ta sẽ thực hiện ví dụ mà đã đề cập ở trên đó là thiết lập thời gian hoạt động cho phép của một Controller.Đẻ thực hiện chức năng này ta sẽ có hai cách  như sau:

* Sử dụng tham số $callback trong Filter dạng hàm trả về kết quả
* Sử dụng tham số $callback trong Filter dạng chuỗi

## Route::group trong laravel

Route::group($attributes, $callback);

Route::group(['namespace'=>'Indexs'],function(){

    //nếu không sử dụng option namespace thì bạn phải viết

    //Route::get('/post','Indexs\PostController@showPost');

    Route::get('/post','PostController@showPost');

});

# Bài 5: Controller trong laravel 5.x

1. Cách tạo 1 Controller trong laravel 5.x

**Cách 1:** Tạo trực tiếp bằng cách tạo 1 file có tên trùng với tên Class của Controller, bạn nên đặt tên file và tên class theo kiểu PascalCase ( viết hoa các chữ cái đầu tiên ) và thêm chữ Controller phía sau để chúng ta phân biên với Model

Bạn vào thư mục app/Http/Controllers và tạo 1 file có  tên là DemoController.php, và thêm đoạn code sau:

**namespace** **App**\**Http**\**Controllers**;

**class** **DemoController** **extends** **Controller**

{

//

}

Ở đây mình tạo 1 Class tên DemoController, class này sẽ kế thừa từ Lớp Controller của Laravel và khai báo namespace là App\Http\Controllers

**Cách 2: Tạo bằng command**

Bạn vào thư mục gốc của ứng dụng mở command line tại thư mục gốc và gõ lệnh

php artisan make:controller DemoController

Lệnh này sẽ tự động tạo cho bạn 1 controller với tên DemoController

Thật đơn giản đúng ko bạn.

2. Tạo action trong Controller

Trong laravel tên các action thường đặt theo dạng camelCase (viết hoa chữ cái đầu tiên của mỗi từ trừ từ đầu tiên) ví dụ getInfoUser

Mỗi action có thể có hay không có tham số truyền vào tùy trường hợp bạn sử dụng.

Mình lấy ví dụ cho bạn dễ hiểu hơn.

Tại route (file routes/web.php) bạn khai báo 1 route nhận tham số là id của user và chỉ định 1 action của Controller xử lý

Route::get('get-user/{id}', 'DemoController@getInfoUser' );

Ở ví vụ này khi bạn gõ <http://localhost/congnghe5s-laravel/get-user/1> ( congnghe5s-laravel là thư mục project của bạn ) Laravel sẽ chỉ định action getInfoUser của Controller DemoController xử lý

Tại controller DemoController bạn cần khai báo 1 action tên getInfoUser và nhận tham số là id của user

**public** **function** **getInfoUser**($idUser){

  // Nhận tham số truyền vào là id user

**echo** $idUser;

}

Trên màn hình của bạn sẽ hiện thị số 1

Trong Controller có thể lấy dữ liệu từ Model và trả về View:

public function showInfo(){

echo "xin chao";

return view("welcome");

}

3. Resource Controllers trong Laravel

Bằng việc khai báo 1 Route Resource trong route, Laravel sẽ tự hiểu trong Controller chứa các phương thức phụ vụ cho CURD (creat - update - read - delete) dữ liệu.

Trong route bạn khai báo như sau

Route::resource('photos', 'PhotoController');

Với 1 câu lệnh ngắn trên Laravel sẽ hiểu là trong Controller PhotoController có các phương thức sau:

| **Verb** | **URI** | **Action** | **Route Name** |
| --- | --- | --- | --- |
| GET | /photos | index | photos.index |
| GET | /photos/create | create | photos.create |
| POST | /photos | store | photos.store |
| GET | /photos/{photo} | show | photos.show |
| GET | /photos/{photo}/edit | edit | photos.edit |
| PUT/PATCH | /photos/{photo} | update | photos.update |
| DELETE | /photos/{photo} | destroy | photos.destroy |

Laravel cung cấp cho bạn cách tạo 1 Controller với các phương thức trên rất đơn giản bằng dòng lệnh

php artisan make:controller PhotoController --resource

# Bài 6: View trong laravel 5.x

## 1. Cách tạo 1 View trong Laravel 5.x

Trong Laravel 5.x, Views được đặt trong thư mục Resources > Views, bạn chỉ cần tạo 1 file .php hoặc .blade.php(blade template) để sử dụng, Ví dụ mình tạo 1 file có tên congnghe5s.blade.php. Trong file này bạn có thể viết code html, css,javascript, php...

Trong file congnghe5s.blade.php mình thêm đoạn html sau:

Cong nghe 5s

## 2. Gọi View trong Laravel 5.x

Trong Route bạn thêm đoạn code

Route::get('congnghe5s', 'DemoController@congNghe5s' );

Nếu bạn đọc bài trước bạn sẽ hiểu rằng khi thêm URL từ congnghe5s,Laravel sẽ chỉ định action congNghe5s trong Controller DemoController xử lý request này

Tại DemoController bạn thêm action sau

**public** **function** **congNghe5s**(){

**return** view('congnghe5s');

}

Bây giờ bạn vào trình duyệt và gõ <http://localhost/congnghe5s-laravel/congnghe5s> màn hình sẽ hiển thị

View nhiều cấp độ: layout > sub > user.php:

// Views

Route::get('quantri', function(){

return view('layout.sub.user');

});

## 3. Truyền dữ liệu qua Views trong Laravel

Để truyền dữ liệu từ Controller qua Views bạn chỉ cần thêm vào tham số thứ 2 của hàm view 1 mảng dữ liệu gồm cặp key => value,

bạn cập nhật lại action congNghe5s:

**public** **function** **congNghe5s**(){

**return** view('congnghe5s', ['name' => 'CongNghe5s', 'email' => 'congnghe5s@abc.xyz']);

}

trong view congnghe5s.blade.php bạn cập nhật lại đoạn code sau

Cong nghe 5s

{{ $name }} - {{ $email }}

Trong Blade Template ký hiệu {{  }} thay thế cho echo trong php. Ở bài sau mình sẽ nói rõ hơn cú pháp Blade Template

**return** view(**'infomation'**, [**'name'**=> $name]);

Refesh lại trình duyệt bạn sẽ nhận được kết quả là:

## 4. Kiểm tra Views có tồn tại hay không

Laravel 5.x hỗ trợ bạn kiểm tra sự tồn tại của file view.

**if** (View::exists('customer')) {

// Nếu tồn tại view tên là customer

}

**Khi gửi form lên theo phương thức Post gửi token:**

<**input type="hidden" name="\_token" value="**{{csrf\_token()}}**"**>

# Tạo file template kế thừa đơn giản

-Như hình ở trên mình có sử dụng các từ khóa lạ như @yield(),@section(),@extends().

–**@yield(), @section-@show**: có tác dụng như một biến nó được tạo ra nhằm báo cho blade template  biết vị trí ý sẽ dùng để chèn nội dung cho biến đó.

–**@include(): Có tác dụng khai báo cho blade template biết vị trí chèn code từ 1 file xác định**

–**@extends()**: Có tác dụng khai báo cho blade template biết là file hiện tại đang kế thừa từ file nào

Rồi bây giờ chúng ta sẽ tạo 1 thư mục **master** trong view. và tạo 3 view blade .

và 1 view tên**index.blade.php** ngang hàng với thư mục **master**

* **master**
  + **master.blade.php**
  + **header.blade.php**
  + **footer.blade.php**

**Trong file index.blade.php**

**@extends('master.master')**

Và khai báo:

**@section('content')**

# Làm việc với database và model trong Laravel

Kết nối database

đầu tiên bạn nên tạo 1 database trong mysql

sau đó vào project sửa phần **.env**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | DB\_CONNECTION=mysql  DB\_HOST=127.0.0.1  DB\_PORT=3306  DB\_DATABASE=vietpro  DB\_USERNAME=root  DB\_PASSWORD= |

-Điều kiện để sử dụng **Schema Builder**:

* Phải có kết nối với database .
* Muốn sử dụng được Schema Builder thì phải gọi namespace của nó **use** **Illuminate\Support\Facades\Schema (web.php và migration đã tích hợp không cần gọi.). Nên gọi khi ta sử dụng câu lệnh của Schema buidle trong controller**

**web.php () –> như mình nói ở trên thì trong web.php không cần gọi : use**

Illuminate\Support\Facades\Schema Route::get('schema/create-table', function () {

Schema::create('users', function($table)

{

$table->increments('id'); // id là tên trường muốn tạo trong bảng.

});});

**Nếu trong controller thì ta phải gọi use Illuminate\Support\Facades\Schema nhé!**

<?php

namespace App\Http\Controllers;

use Illuminate\Support\Facades\Schema;

use Illuminate\Http\Request;

class CreateTable extends Controller

{

public function CreateTable()

{

Schema::create('users', function($table)

{

$table->increments('id'); // id là tên trường muốn tạo trong bảng.

});

}}

**kiểu dữ liệu cho các trường trong bảng:**

| **Kiểu dữ liệu** | **Miêu tả** |
| --- | --- |
| $table->bigIncrements('id'); | Thuộc kiểu “BIGINT” và là khoá chính |
| $table->bigInteger('votes'); | Thuộc kiểu “BIGINT” |
| $table->binary('data'); | Thuộc kiểu BLOB |
| $table->boolean('confirmed'); | Thuộc kiểu BOOLESE |
| $table->char('name', 4); | Thuôc kiểu CHAR(4) |
| $table->date('created\_at'); | Thuôc kiểu Date |
| $table->dateTime('created\_at'); | Thuôc kiểu DateTime |
| $table->decimal('amount', 5, 2); | Thuôc kiểu DECIMAL |
| $table->double('column', 15, 8); | Thuôc  double kiểu tương đương với độ chính xác, tổng cộng 15 chữ số và 8 chữ số sau dấu thập phân |
| $table->float('amount'); | Thuôc kiểu FLOAT |
| $table->increments('id'); | Tăng ID vào bảng (khóa chính) |
| $table->integer('votes'); | Thuôc kiểu INT |
| $table->json('options'); | Thuôc kiểu JSON |
| $table->jsonb('options'); | Thuôc kiểu JSONB |
| $table->longText('description'); | Thuôc kiểu LONGTEXT |
| $table->mediumInteger('numbers'); | Thuôc kiểu MEDIUMINT |
| $table->mediumText('description'); | Thuôc kiểu MEDIUMTEXT |
| $table->smallInteger('votes'); | Thuôc kiểu SMALLINT |
| $table->tinyInteger('numbers'); | Thuôc kiểu TINYINT |
| $table->string('email'); | Cột tương đương VARCHAR |
| $table->string('name', 100); | Thuôc kiểu VARCHAR (100) |
| $table->text('description'); | Thuôc kiểu Text |
| $table->time('sunrise'); | Thuôc kiểu Time |
| $table->timestamp('added\_on'); | Thuôc kiểu TIMESTAMP |
| $table->timestamps(); | Thêm các 2 cột **created\_at** và **update\_at** |
| $table->rememberToken(); | Thêm cột remember\_tokendưới dạng VARCHAR (100) NULL |

## Đổi tên bảng

Schema::rename('users','admin');

## Xóa bảng

Schema::dropIfExists('tableName');

## Đổi tên cột

Schema::table('users', function ($table) {

    $table->renameColumn('email', 'user\_mail');

});

## Xóa cột

Schema::table('tableName', function ($table) {

    $table->dropColumn('columnName');

    //hoặc xóa nhiều cột cùng lúc.

    $table->dropColumn(['columnName1','columnName2','columnName3']);

});

# Quản lý bảng bằng Migration trong laravel

# Tạo migrations

Để tạo một Laravel Migration chúng ta sử dụng câu lệnh artisan:

cách tạo migration thông thường:



|  |  |
| --- | --- |
| 1 | php artisan make:migration create\_users\_table |

* **php artisan make:migration TenMigrate --create=TableName**  : Tạo migrations thêm  bảng. – nên dùng cách này
* **php artisan make:migration TenMigrate --table=TableName**  : Tạo migrations chỉnh sửa bảng (Thêm cột,xoá cột).

Khi đó một file migration sẽ được tạo ra trong thư mục database/migrations, trong tên các file migration có chứa thông tin thời gian tạo ra giúp cho Laravel sắp xếp và xác định được các file mirgration tại các thời điểm cần thiết.

Chúng ta lên đặt tên sao cho khi nhìn vào 1 migration phải biết được nó dùng để tạo bảng tên gì : cấu trúc đặt tên: create\_tên bảng\_table.

ví dụ ta muốn dùng migration để tạo bảng tên là comment ta gõ lệnh artisan như sau:



|  |  |
| --- | --- |
| 1 | php artisan make:migration create\_comment\_table |

# Cấu trúc của Migrations

File migration chứa Migration class với hai phương thức up() và down(), phương thức up() sử dụng để thêm các bảng, cột hoặc tạo một index cho cột nào đó trong CSDL. Phương thức down() sử dụng để làm những việc ngược lại với phương thức up().

túm lại:

-Hàm up trong Migrations có tác dụng thực thi. khi gõ lệnh **migrate** (xuống dưới là biết).

-Hàm down trong Migrations có tác dụng thực thi . khi gõ lệnh **rollback** (xuống dưới là biết).

Chúng ta cùng **xem ví dụ sau:**



|  |  |
| --- | --- |
| 1 |  |

# Thực thi các Migrations

|  |  |
| --- | --- |
| php artisan migrate | chạy migration |
| php artisan migrate:resest | resest lại migration |
| php artisan migrate:refesh | chạy lại migration |
| php artisan migrate:status | xem trạng thái của migration |
| php artisan migrate:install | cài đặt migration |

Để chạy tất cả các migration có trong database/migraions sử dụng lệnh artisan như sau:



|  |  |
| --- | --- |
| 1 | php artisan migrate |

Các migration file có thể ảnh hưởng đến cấu trúc database hiện tại, do đó khi chạy lệnh trên có thể nó sẽ thông báo xác nhận chạy hay không, để ép buộc chạy migration mà không hiển thị thông báo sử dụng thêm tùy chọn –force



|  |  |
| --- | --- |
| 1 | php artisan migrate --force |

Trong quá trình migration có thể bạn muốn quay trở lại cấu trúc CSDL trước lúc thực thi các migration, hoàn toàn có thể được bằng cách sử dụng rollback và reset. Rollback thì cần cung cấp thêm tùy chọn –step là rollback lại bao nhiêu lần , không có —-step mặc định là 1 , còn **reset sẽ đưa cấu trúc CSDL về thời điểm chưa có bất kỳ một migration** nào.



|  |  |
| --- | --- |
| 1  2 | php artisan migrate:rollback --step=3 // nếu không ghi --step=3 mặc định là 1 nhé  php artisan migrate:reset |

Ngoài ra, Laravel còn cung cấp migrate:refresh, đúng như tên gọi của nó, lệnh ra rollback toàn bộ các migrate và sau đó chạy lại lệnh migrate, tức là tương đương hai lệnh sau chạy lần lượt:



|  |  |
| --- | --- |
| 1 | php artisan  migrate:refresh |



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | <?php    use Illuminate\Support\Facades\Schema;  use Illuminate\Database\Schema\Blueprint;  use Illuminate\Database\Migrations\Migration;    class CreateUsers extends Migration  {      /\*\*       \* Run the migrations.       \*       \* @return void       \*/      public function up()      {          Schema::create('users', function (Blueprint $table) {              $table->increments('id');              $table->string('email',50);              $table->string('password');              $table->string('full')->nullable();              $table->string('address')->nullable();              $table->string('phone')->nullable();              $table->tinyInteger('level');              $table->string('remember\_token')->nullable();          });    }        /\*\*       \* Reverse the migrations.       \*       \* @return void       \*/      public function down()      {          Schema::dropIfExists('users');      }  } |

Với migration này sau khi chạy câu lệnh :**php artisan migrate** sẽ tạo cho chúng ta bảng users sau:

# Query builder thực thi truy vấn CSDL

DB::**table**(**'sinhvien'**)->insert([  
 [**'full'** => **'Nguyễn Văn A'**, **'password'** => **'123456'**, **'email'** => **'anv@gmail.com'**, **'level'** => 0],  
 [**'full'** => **'Nguyễn Văn B'**, **'password'** => **'123456'**, **'email'** => **'bnv@gmail.com'**, **'level'** => 0],  
 [**'full'** => **'Nguyễn Văn C'**, **'password'** => **'123456'**, **'email'** => **'cnv@gmail.com'**, **'level'** => 0],  
 [**'full'** => **'Nguyễn Văn D'**, **'password'** => **'123456'**, **'email'** => **'dnv@gmail.com'**, **'level'** => 0],  
]);

<http://vietpro.net.vn/laravel-17-query-builder-thuc-thi-truy-van-csdl.html>

# Tạo dữ liệu mẫu với Seeding

<http://vietpro.net.vn/laravel-18-tao-du-lieu-mau-voi-seeding.html>

### Tạo seeder

-Để tạo Seeder trong Laravel chúng ta dùng lệnh:

 php artisan make:seed seedName

-Với seedName: là tên của seed các bạn muốn tạo (nên đặt tên có ý nghĩa để sau này dễ quản lý).

# Insert dữ liệu bằng Seeder

Mình sẽ thêm dữ liệu cho bảng sinhvien như sau:

**class** SinhViens **extends** Seeder  
{  
 */\*\*  
 \* Run the database seeds.  
 \*  
 \** ***@return*** *void  
 \*/* **public function** run()  
 {  
 *// Tạo dữ liệu test khi chạy lệnh db:seed* DB::*table*(**'sinhvien'**)->insert([  
 **'full'** =>**'Bùi Văn Chiến'**,  
 **'email'** => **'Chienbv@gmail.com'**,  
 **'level'** => 0,  
 **'password'** => bcrypt(**'123456'**),  
 ]);  
  
 }  
}

Chạy riêng file seeder: **php artisan db:seed --class=Users**

**Nếu chạy tất cả file seeder thì gọi chung vào** DatabaseSeeder.php và chạy:

php artisan db:seed // mặc định chạy file DatabaseSeeder.php

*// Tạo dữ liệu test khi chạy lệnh db:seed*DB::*table*(**'sinhvien'**)->insert([  
 **'full'** =>**'Bùi Văn Chiến'**,  
 **'email'** => **'Chienbv@gmail.com'**,  
 **'level'** => 0,  
 **'password'** => bcrypt(**'123456'**),  
]);

# Tìm Hiểu Middleware trong laravel

<http://vietpro.net.vn/laravel-23-tim-hieu-middleware-trong-laravel.html>

# Tìm hiểu về model trong laravel

# Tạo bằng lệnh Laravel cung cấp sẵn.

-Mình khuyến khích các bạn dùng cách này nhé.

-Tạo model bằng cách này thì khá là đơn giản. Các bạn chỉ cần mở cmd(commandline) lên và trỏ đến thư mục chứa project Laravel của các bạn và gõ lệnh.

php artisan make:model News

* News là tên model, các bạn có thể chỉnh thành tên model các bạn muốn tạo.
* Ở dòng thứ hai thì chúng ta cũng tạo được model nhưng đồng thời chúng ta sẽ được tạo luôn một file migration( nói ở bài sau).

-Sau đó enter và vào trong thư mục App/ bạn sẽ thấy có một file News.php trong đó và bên trong sẵn có đoạn code.

### Khai báo table cần sử dụng trong Model.

-Ở trong Laravel mỗi một model ứng với một bảng(table) dữ liệu trong CSDL và để khai báo model sử dụng bảng dữ liệu nào trong database thì mọi người khai báo dòng sau trong class model.

protected $table='tableName';

Trong đó : tableName là tên table các bạn muốn sử dụng

# Lọc cột dữ liệu trong model.

-Trong laravel chẳng hạn như bảng user tôi chỉ muốn truy vấn cột username,useremail mà không muốn sử dụng cột passwork thì sau. Chính vì điều đó Laravel cũng đã cung cấp cho chúng ta một thông số là **fillable** để điều chỉnh các cột cần sử dụng.

protected $fillable = ['column1','column2',..,'columnn'];

# Query builder thực thi truy vấn CSDL

<http://vietpro.net.vn/laravel-17-query-builder-thuc-thi-truy-van-csdl.html>

*$users = DB::table('users')*

*->where('vote', '>', 50)*

*->orWhere('comment', '>', 50)*

*->get();*

*// Ví dụ về whereBetween*

*$users = DB::table('users')*

*->whereBetween('age', [18, 35])->get();*

*// Ví dụ về whereIn*

*$users = DB::table('users')*

*->whereIn('id', [1, 2, 3])*

*->get();*

*// Ví dụ về whereNull*

*$users = DB::table('users')*

*->whereNull('vote')*

*->get();*

*// Ví dụ về whereDate*

*$users = DB::table('users')*

*->whereDate('join\_date', '2017-03-27')*

*->get();*

*//Ví dụ về whereDay*

*$users = DB::table('users')*

*->whereDay('join\_date', '27')*

*->get();*

*// Ví dụ về whereMonth*

*$users = DB::table('users')*

*->whereMonth('join\_date', '03')*

*->get();*

*//Ví dụ về whereYear*

*$users = DB::table('users')*

*->whereYear('join\_date', '2017')*

*->get();*

**Get sinhvien theo id trả về Json:**

$sinhvien = SinhVien::*where*(**'id'**, **'='**, $id)->get();  
**return** response()->json($sinhvien, 200);