Bài 1: Laravel là gì và các lệnh khởi tạo laravel trong command

* Laravel là 1 framework theo mô hình MVC
* Cài bằng composer.exe
* Windowns + R => Gõ composer –v hoặc vào thử mục htdocts ấn Shift + chuột phải => open command windows here
* Gõ lệnh “composer create-project laravel/laravel-53 –prefer-dist”
* Tiếp theo gõ lệnh “composer.phar install”
* Tiếp đến php artisan key:generate để kích hoạt application key của laravel
* Khởi chạy laravel bằng lệnh “artisan serve”

Bài 2: Cấu trúc của Laravel

* **app:** Chứa các thư mục, các tập tin php, thư viện, models.
  + **Console:** Chứa các tập tin định nghĩa các câu lệnh trên artisan.
  + **Excerption:**  Chứa các tập tin quản lý, điều hướng lỗi.
  + **Http**
    - **Controllers** : Chứa các controllers của project.
    - **Middleware:**Chứa các tập tin lọc và ngăn chặn các requests.
    - **Kernel.php:**Cấu hình, định nghĩa Middleware.
  + **Providers:**Chứ các providers thực hiện việc binding vào service container
  + **User.php:**Là model User mà Laravel tự địn sẵn cho chúng ta.
* **bootstrap:**Chứa tập tin điều hướng hệ thống.
* **config:** Chứa mọi tập tin cấu hình của Laravel.
* **database:** Chứa các thư mục tập tin vềdatabase.
  + **migrations:** Chứa các tập tin định nghĩa,khởi tạo và sửa bảng.
  + **seeds:** Chứa các tập tin định nghĩa dữ liệu insert vào trong database.
  + **factories:** Chứa các tập tin định nghĩa các cột bảng dữ liệu để tạo ra các dữ liệu ảo.
* **public:** Chứa các tập tin css, js, image.
  + **index.php:**Đây là tệp tin root của Laraver
* **resources:** Chứa các views, ngôn ngữ của project.
* **routes:** Chứa các tập tin định nghĩa các router, xử lý điều hướng router bao gồm : web, api và console
* **storage:** Chứa các tập tin hệ thống cache, session, ...
* **tests: ...**
* **vendor:** Chứa các thư viện của composer.
* **.env:** Là tập tin cấu hình chính của laravel như key app,database.
* **.env.example:**Tệp tin cấu hình mẫu của laraver.
* **composer.json:** tập tin của composer.
* **composer.lock:** tập tin của composer.
* **package.js:** Tập tin cấu hình của nodejs
* **gulpfile.js:** Là tập tin gulp builder.
* **phpunit.xml:** Là tập tin xml của phpunit dùng để testing project.
* **server.php:** Là tập tin để artisan trỏ đến tạo server khi gõ lệnh php artisan serve
* **artisan:** Tập tin thực thi lệnh của Laravel.

Bài 3 + 4 + 5: Route trong Laravel

* **Route::get =>** nhận resquest với phương thức GET.
* **Route::post =>** nhận resquest với phương thức POST.
* **Route::put** => nhận resquest với phương thức PUT.
* **Route::delete** => nhận resquest với phương thức DELETE.
* **Route::match** => kết hợp nhiều phương phức như POST,GET,PUT,..
* **Route::any** => nhận tất cả các phương thức.
* **Route::group =>** tạo ra các nhóm route.
* **Route::controller =>** gọi đến controller tương ứng mà chúng ta tự định.
* **Route::resource =>** sử dụng với resource controller
* Cú pháp: **Route::resource(‘url’, ‘tênController’, ‘tùy biến’);**

**Lệnh commandline:**

**Php artisan make:controller tenController --resource**

* **Chú thích các function:**
* index(): Hiển thị một danh sách.
* create(): Thêm mới.
* store(): Lưu mới.
* show($id): Hiển thị một dữ liệu theo tham số truyền vào.
* edit($id): Sửa một dữ liệu theo tham số truyền vào.
* update($id):Cập nhật một dữ liệu theo tham số truyền vào.
* destroy($id):Xóa một dữ liệu theo tham số truyền vào.

**Route::group()**

**Cú pháp: Route::group($attr, $handle);**

**Trong đó: $attr là các mảng thành phần điều kiện**

**$handle là các đoạn sử lý thường là hàm**

**Bài 6: View trong Laravel**

**Tạo view trong Laravel phải có đuôi là .php hoặc blade.php**

**Tạo thư mục cho view nằm trong resource**

**Gọi view:**

**view(‘duongdantoiview’, ‘dulieutruyenvaoview’);**

**Gọi view trong route:**

**Route::get(‘duongdanroute’, function(){**

**Return view(‘tenview’, ‘data’);**

**});**

**Gọi view trong Controller:**

**Public function index()**

**{**

**Return view(‘view’, ‘data’);**

**}**

**Cách truyền dữ liệu cho view**

**Dùng Compact:**

**Compact ('tenbientruyenvao')**

**Dùng With:**

**View ('tenview')->with ('key', 'value');**

**Dùng Mảng:**

**View ('tenview', ['key'=>'value']);**

**Bài 7 + 8: Blade template engine trong Laravel**

**- Tạo blade template trong Laravel (huynguyen.blade.php)**

**- Sử dụng blade template {{}}**

**VD:**

**Vòng lặp for**

**@for ($i = 0; $i < 10; $i++)**

**In ra giá trị {{ $i }} <br/>**

**@endfor**

**Vòng lặp foreach**

**@foreach ($users as $user)**

**<p>This is user {{ $user->id }}</p>**

**@endforeach**

**Vòng lặp forelse**

**@forelse($users as $user)**

**<li>{{ $user->name }}</li>**

**@empty**

**<p>No users</p>**

**@endforelse**

**Vòng lặp while**

**@while (true)**

**<p>I'm looping forever.</p>**

**@endwhile**

**Câu điều kiện if - else**

**@if (count($records) === 1)**

**I have one record!**

**@elseif (count($records) > 1)**

**I have multiple records!**

**@else**

**I don't have any records!**

**@endif**

\* Tính kế thừa giao diện

* @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.
* @extends(): Có tác dụng khai báo cho blade template biết là file hiện tại đang kế thừa từ file nào
* @Parent có thể đặt trước nội dung muốn thêm vào (nếu muốn phần được thêm sẽ ở sau phần khởi tạo) và ngược lại.

**Bài 9: Controller trong Laravel**

**Tạo controller trong Laravel.**

**-Cũng giống như view, để tạo controller trong Laravel cũng phải có một số ràn buộc như sau:**

* **Controller phải được đặt trong đường dẫn App\Http\controllers**
  + **Tên của controller phải giống với tên class trong file controller đó.**
  + **Class controller phải extends(kế thừa) từ Controller (viết hoa chữ C).**

**Command**

**Php artisan make:controller controllername**

Hoặc

**Php artisan make:controller controllername –resource**

**Sử dụng dụng các action trong Controllers.**

**-Để tạo một action trong controllers thì chúng ta chỉ việc tạo một hàm ở trong controllers và gọi nó ở trong Route là được.**

**-VD:**

**<?php**

**namespace App\Http\Controllers;**

**class homeController extends Controller**

**{**

**public function index(){**

**echo "Đây là index trong homeController";**

**}**

**}**

**Route::get('call-controller','homeController@index');**

**Truyền tham số trong action Controllers.**

**-VD:**

**<?php**

**namespace App\Http\Controllers;**

**class homeController extends Controller**

**{**

**public function index($name,$age){**

**echo "Xin chào $name, $age tuổi";**

**}**

**}**

**Route::get('a/{name}/{age}','homeController@index')->where(['name'=>'[a-zA-Z]+','age'=>'[0-9]+']);**

**Bài 10: Query Builder trong Laravel**

**Cấu hình database.**

**-Trước tiên để sử dụng được Query builder cần phải cấu hình database kết nối đến cơ sở dữ liệu.**

**-Để cấu hình Database các bạn cần mở file .env lên và cấu hình như sau.**

**APP\_ENV=local**

**APP\_KEY=base64:w6dZccXCvdehAax2NYUt6f/igtsyw7QTQCx2eGEQBgI=**

**APP\_DEBUG=true**

**APP\_LOG\_LEVEL=debug**

**APP\_URL=http://localhost**

**DB\_CONNECTION=mysql**

**DB\_HOST=127.0.0.1**

**DB\_PORT=3306**

**DB\_USERNAME=root**

**DB\_PASSWORD=**

**BROADCAST\_DRIVER=log**

**CACHE\_DRIVER=file**

**SESSION\_DRIVER=file**

**QUEUE\_DRIVER=sync**

**REDIS\_HOST=127.0.0.1**

**REDIS\_PASSWORD=null**

**REDIS\_PORT=6379**

**MAIL\_DRIVER=smtp**

**MAIL\_HOST=mailtrap.io**

**MAIL\_PORT=2525**

**MAIL\_USERNAME=null**

**MAIL\_PASSWORD=null**

**MAIL\_ENCRYPTION=null**

**PUSHER\_APP\_ID=**

**PUSHER\_KEY=**

**PUSHER\_SECRET=**

**-Chú thích:**

**DB\_HOST : Là server chứa mysql**

**DB\_PORT: Là cổng post của server Mysql (thường để mặc định là 3306).**

**DB\_DATABASE: Là tên database**

**DB\_USERNAME: Là username để truy cập vào database**

**DB\_PASSWORD: Là password để truy cập vào database**

**Sử dụng Query Builder.**

**Lấy tất cả dữ liệu trong bảng.**

**-Cú Pháp:**

**DB::table('tablename')->get()**

**Select cột trong bảng.**

**-Cú pháp:**

**DB::table('tablename')->select('columnfirst','columnsecond')->get();**

**Select có điều kiện.**

**-Cú pháp:**

**+Điều kiện == :**

**DB::table('tablename')->where('column','filter')->get();**

**+Điều kiện >:**

**DB::table('tablename')->where('column','>','filter')->get();**

**+Điều kiện <:**

**DB::table('tablename')->where('column','<','filter')->get();**

**+Điều kiện !=:**

**DB::table('tablename')->where('column','<>','filter')->get();**

**+Điều kiện lồng:**

**VD:**

**$users = DB::table('users')->where('votes', '>', 100)->orWhere('name', 'John')->get();**

**Truy vấn tìm kiếm.**

**DB::table('tablename')->where('column','like','filter')->get();**

**// vd : DB::table('product')->where('name','like','%Sách%')->get();**

**Join bảng.**

**$users = DB::table('users')->join('contacts', 'users.id', '=', 'contacts.user\_id')->get();**

**+Left join:**

**$users = DB::table('users')->leftjoin('contacts', 'users.id', '=', 'contacts.user\_id')->get();**

**Unions.**

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

**->whereNull('first\_name');**

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

**->whereNull('last\_name')**

**->union($first)**

**->get();**

**OrderBy.**

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

**->orderBy('name', 'desc')**

**->get();**

**Random.**

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

**->inRandomOrder()**

**->first();**

**GroupBy/having.**

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

**->groupBy('account\_id')**

**->having('account\_id', '>', 100)**

**->get();**

**Thêm (insert).**

**DB::table('users')->insert(**

**['email' => 'john@example.com', 'votes' => 0]**

**);**

**Sửa (update).**

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

**->where('id', 1)**

**->update(['votes' => 1]);**

**Xóa (delete).**

**DB::table('users')->delete();**

**//hoặc**

**DB::table('users')->where('votes', '>', 100)->delete();**

**3, Chú ý.**

**- Để sử dụng được các câu truy vấn trên bắt buộc phải:**

**Kết nối với cơ sở dữ liệu.**

**Nếu truy vấn trong controllers thì cần phải khai báo**

**use Illuminate\Support\Facades\DB; còn trong Route thì không cần**