**BỘ CÂU HỎI PHỎNG VẤN MODULE 3**

**PHP WEB BACK-END DEVELOPMENT**

|  |  |  |
| --- | --- | --- |
| **ST** | **Câu hỏi** | |
| 1 | **Laravel framework là gì ?** | |
|  | **Laravel** là một PHP framework, có mã nguồn mở và miễn phí được phát triển bởi Taylor Otwell, xây dựng nhằm hỗ trợ phát triển các phần mềm, ứng dụng, theo kiến trúc MVC.  *Những lý do khiến Laravel trở nên rộng rãi:*   * Cú pháp dễ hiểu – rõ ràng * Hệ thống đóng gói modular và quản lý gói phụ thuộc * Nhiều cách khác nhau để truy cập vào các cơ sở dữ liệu quan hệ * Nhiều tiện ích khác nhau hỗ trợ việc triển khai vào bảo trì ứng dụng.   **Framework chính là** một thư viện với các tài nguyên có sẵn cho từng lĩnh vực để lập trình viên sử dụng thay vì phải tự thiết kế | |
| 2 | **Phiên bản mới nhất của laravel là gì?** | |
|  | 8.21.0 | |
| 3 | **Làm thế nào chúng ta có thể cài đặt Laravel bởi composer?** | |
|  | Composer create-project laravel/laravel TenProjectLaravel | |
|  |  | |
| 4 | **Middleware trong Laravel là gì?** | |
|  | Là việc lọc (filter) các request trong ứng dụng giữa client và server. Nếu request hợp lệ thì sẽ cho đi tiếp, còn ko sẽ bị dừng và chuyển sang trang khác.  VD: Khi user muốn xem 1 trang nào đó phải qua middleware thì middleware sẽ xem user này đã đăng nhâpj hay chưa, nếu chưa thì sẽ chuyển tới 1 trang đăng nhập. | |
| 5 | **Các khái niệm cơ bản trong laravel là gì?** | |
|  |  | |
|  | * **Route**:   Định tuyến (route) trong Laravel có nghĩa là chỉ dẫn một request tải một trang đến một method tương ứng để xử lý nằm trong các Controller.  Các route này nằm trong file:   * Laravel phiên bản 5.2 trở về trước route nằm trong app/Http/routes.php * Từ phiên bản Laravel 5.3 trở đi route sẽ nằm trong thư mục routes với hai file là web.php và api.php   Các phương thức dùng cho route:  Route::get($uri, $callback);  Route::post($uri, $callback);  Route::put($uri, $callback);  Route::patch($uri, $callback);  Route::delete($uri, $callback);  Route::options($uri, $callback);  Thông thường ta dùng get với post.  VD:  Route::get('/login', [\App\Http\Controllers\frontend\LoginController::class, 'showLogin'])->name('login.show');  Khi có 1 request gửi tới ứng dụng, sẽ vào trong route kiểm tra xem request là method get hay ko, và uri có phải /login hay ko, nếu đúng cả 2 sẽ chạy vào LoginController và tìm đến method showLogin và thực hiện tiếp đoạn mã trong method này.  Ta có thể truyền tham số   * **Middleware**   Middleware là một thành phần trung gian nữa client và server, dùng để xử lý, lọc các HTTP request đi vào ứng dụng. Nếu request hợp lệ sẽ cho đi tiếp, còn không sẽ bị dừng lại hoặc redirect sang một trang khác. \*\*\* Muốn sử dụng middleware ta cần qua các bước:  + Tạo middleware bằng câu lệnh: php artisan make:middleware CheckUser  + Đăng ký middleware ở App/Http/Kernel phần  protected $routeMiddleware = [  với khai báo tên middleware và class của nó  + Chỉnh sửa class middleware bằng cách kiểm tra xem nó thoả mãn điều kiện để đi tiếp hay ko.   * **CSRF Token**   Laravel sẽ cung cấp một cách đơn giản để bảo về ứng dụng khỏi tấn công [cross-site request forgery](https://en.wikipedia.org/wiki/Cross-site_request_forgery) (CSRF) - Tấn công mạo danh. CSRF là loại tấn công mà hacker đánh lừa trình duyệt, sử dụng phiên làm việc của người dùng để gửi request lên ứng dụng Web.  Muốn sử dụng ta đặt ở các form bằng câu lệnh @csrf, nếu thiếu câu lệnh này khi gửi lên sẽ bị lỗi 419 expired.  Token của @csrf sẽ tự sinh ra khi có 1 request gửi tới ứng dụng.   * **Localization**   Laravel's localization (i18n) là một các tiện lợi để sử dụng đa ngôn ngữ trong ứng dụng. Các ngôn ngữ sẽ được chứa trong folder resources/lang  Ta có thể cài đặt nguôn ngữa tiếng việt tiếng anh bằng tạo file trong  Ressources/lang/vi/languages.php  Ressources/lang/en/languages.php  Trong file ngôn ngữ được định dang theo kiểu key value của 1 mảng  Và gọi nó ra bằng cách {{ \_\_(‘ languages.key’)\_\_}} , \_\_ là 1 hàm helper  **Authentication với Authorization**  **Authentication (xác thưcj)**: Quá trình xác thực, định danh 1 tài khoản đăng nhập vào ứng dụng. VD xem 1 user đã đăng nhập hay chưa.  **Authorization(phân quyền)**: Khi người dùng được xác thực thì bước tiếp theo là user này có quyền làm gì, VD như user này ko thể xoá bài viết của người khác.   * **Helpers**   Là những function có thể sử dụng ở mọi nơi trên ứng dụng như là hàm asset(), view(),  url(), dd(), session(), back(), ...   * **Eloquent với Query Builder**  1. Bảo mật  * Eloquent có tính bảo mật hơn Query Builder trong việc phòng chống SQL Injectio  2. Tính dễ sử dụng  * Eloquent ORM dễ sử dụng hơn trong việc truy xuất, thay đổi cơ sở dữ liệu, cú pháp ngắn gọn, đơn giản hơn QueryBuilder. * Eloquent ORM dễ dàng kết nối với database hơn QueryBuilder. * Về độ phức tạp của một câu SQL mà Eloquent ORM chưa thể đáp ứng thì chúng ta phải sử dụng DB::raw hoặc QueryBuilder.  3. Hiệu suất  * QueryBuilder có hiệu suất truy vấn dữ liệu nhanh hơn Eloquent ORM bởi vì Eloquent phải thêm một lớp trong ứng dụng và yêu cầu nhiều truy vấn SQL. Đối với các database mà có ít bản ghi hiệu suất của chúng không có quá là nhiều sự chênh lệch, vậy nên đối với những database này tôi khuyên các bạn nên sử dụng Eloquent ORM vì cú pháp đơn giản và ngắn gọn của chúng. * **Eager Loading** * **Scope**   Scope được sử dụng để tái sử dụng lại các câu lệnh truy vấn trong một model. Nếu có 1 câu lệnh được dùng đi dùng lại nhiều lần, ta có thể tạo thành scope để tái sử dụng và dễ dàng mở rộng. | |
|  |  | |
| 6 | **Migration trong laravel là gì? Làm thế nào để sử dụng chúng?** | |
|  | Laravel Migration là cách thức quản lý các phiên bản cơ sở dữ liệu (database version control) giúp cho làm việc nhóm với cơ sở dữ liệu dễ dàng trong việc chia sẻ và thay đổi các kiến trúc cơ sở dữ liệu  Để tạo 1 bảng trong CSDL ta dùng:  php artisan make:migration create\_users\_table --create=users  sẽ tao được 1 migration với tạo bản users  Để thay đổi thông tin của 1 bảng ta dùng;  php artisan make:migration add\_birthday\_column\_to\_users\_table --table=users | |
| 7 | **Service providers trong laravel là gì?** | |
|  | **Service Provider**là thành phần trung tâm của việc khởi tạo tất cả các ứng dụng Laravel bao gồm các thành phần core. Việc đăng kí các liên kết tới service container, event listeners, middleware, và thậm chí các route. **Service provider** chính là nơi để cấu hình ứng dụng của bạn. Đây cũng chính là nơi khai báo các service provider sau này khi muốn viết một service provider  Service providers có một vị trí trung tâm của toàn bộ ứng dụng Laravel. Service providers là một công cụ mạnh mẽ để quản lí các class dependencies và thực hiện quá trình dependency injection. Service provider nói với Laravel bind các thành phần khác nhau vào Service container của laravel. Bạn có thể sử dụng lệnh php artisan make: provider ClientsServiceProvider trên command để tự tạo ra một service provider. Nó sẽ cung cấp cho chúng ta 2 function là: register() và boot(). | |
| 8 | **Lumen là gì?** | |
|  | Lumen là một framework PHP được phát triển bởi cha đẻ của Laravel – Taylor Otwell. Theo tác giả , Lumen không có ý định thay thế Laravel. Trên thực tế Lumen được tạo ra để làm hệ thống microservices – một hệ thống nhỏ hơn để hỗ trợ các hệ thống lớn như Laravel.  Đây là giải pháp hoàn hảo để phát triển các microservices và fast API dựa trên Laravel. Đó là một dự án mới được tạo ra bởi Taylor Otwell. Nó được xây dựng cho microservice, không nhiều cho các ứng dụng giao tiếp với người dùng. Bạn phải sử dụng lệnh này để cài đặt lumem composer global require "laravel/lumen-installer=~1.0". | |
| 9 | **Cấu hình cho Laravel 7x là gì?** | |
|  |  | |
| 10 | **Làm thế nào chúng ta có thể tắt bảo vệ CRSF cho một Route cụ thể?** | |
|  | Chúng ta có thể thêm URL cụ thể hoặc Route trong biến $except trong file app\Http\Middleware\VerifyCsrfToken.php  class VerifyCsrfToken extends BaseVerifier {  protected $except = [  'Pass here your URL',  ];  } | |
| 11 | **Laravel sử dụng template engine nào?** | |
|  | Laravel sử dụng **Blade Template Engine**. Nó là một templating engine đơn giản và mạnh mẽ được cung cấp bởi Laravel. | |
| 12 | **Façade trong laravel là gì ? Làm sao để sử dụng nó ?** | |
|  | Facade là 1 kiểu class, class này cung cấp 1 static interface cho services. Facade giúp truy cập 1 service trực tiếp từ container. Nó được định nghĩa trong Illuminate\Support\Facades, nhờ đó chúng ta có thể dễ dàng sử dụng.  use Illuminate\Support\Facades\Cache;  Route::get('/cache', function () {  return Cache::get('PutkeyNameHere');  }); | |
| 13 | **Làm thế nào để tạo 1 helper trong Laravel ?** | |
|  | 1. Tạo một app/helpers.php trong thư mục ứng dụng 2. Thêm   "files": [  "app/helpers.php"  ]  trong biến autoload của file composer.json.   1. Sau đó update composer bằng composer dump-autoload và composer update. | |
| 14 | **Sử dụng middleware trong laravel như thế nào?** | |
|  | Trong Laravel Middleware như một người trung gian giữa request và response. Middleware là một loại cơ chế lọc HTTP request.   1. Chúng ta có thể tạo 1 middelware với lệnh php artisan make:middleware UsersMiddleware 2. Ở đây UsersMiddleware là tên của Middleware. Sau khi chạy lệnh này nó sẽ tạo ra một file UsersMiddleware.php trong đường dẫn app/Http/Middleware. 3. Sau đó chúng ta phải đăng kí middleware vừa tạo kernel.php trong đường dẫn app/Http.   'Users' => \App\Http\Middleware\UsersMiddleware::class,   1. Bây giờ chúng ta có thể gọi middleware Users ở những chỗ chúng ta cần như controller hoặc route.   //Controller  public function \_\_construct() {  $this->middleware('Users');  }  //Route  Route::group(['middleware' => 'Users'], function () {  Route::get('/', 'HomeController@index');  }); | |
| 15 | **Một Artisan là gì?** | |
|  | Artisan là một kiểu command line interface" sử dụng trong Laravel. Nó cung cấp rất nhiều lệnh hữu ích cho bạn trong khi phát triển ứng dụng của bạn. Một số lệnh artisan phổ biến   * php artisan list * php artisan --version * php artisan help * php artisan make:controller * php artisan make:model * php artisan make:migration * php artisan make:middleware * php artisan make:auth * php artisan make:mail * php artisan make:provider | |
| 16 | **Service container là gì?** | |
|  | Service Container là một công cụ mạnh mẽ được sử dụng để quản lý các class dependencies và thực hiện dependency injection. Nó còn được gọi là container IoC. | |
| 17 | **Làm thế nào chúng ta có thể cấu hình một Mail trong laravel?** | |
|  | Laravel cung cấp API rõ ràng và đơn giản trên thư viện phổ biến SwiftMailer với các drivers cho SMTP, Mailgun, SparkPost, Amazon SES và gửi email. Laravel đang cho phép chúng tôi gửi Mail quickly thông qua các dịch vụ local hoặc trên nền tảng đám mây.  Nó cho phép chúng ta lưu trữ thông điệp email của chúng ta trong các file view.  public function sendEmail(Request $request, $id)  {  $user = Admin::find($id);  Mail::send('emails.reminder', ['user' => $user], function ($m) use ($user) {  $m->from('info@bestinterviewquestion.com', 'Reminder');  $m->to($user->email, $user->name)->subject('Your Reminder!');  });  } | |
| 18 | **Auth là gì ? Làm sao để sử dụng chúng ?** | |
|  | Laravel Auth là quá trình xác định thông tin đăng nhập của người dùng với cơ sở dữ liệu. Laravel quản lý nó với sự trợ giúp của các sessions, Các sessions lấy tham số đầu vào như tên người dùng và mật khẩu, để nhận dạng người dùng. Nếu các cài đặt khớp nhau thì người dùng được cho là đã được xác thực. Auth là chức năng được xây dựng do Laravel cung cấp; chúng ta phải sử dụng lệnh php artisan make: auth Auth được sử dụng để xác định thông tin đăng nhập của người dùng với cơ sở dữ liệu. | |
| 19 | **Các tính năng của laravel là gì?** | |
|  | 1. Cung cấp một bộ chức năng phong phú như Eloquent ORM, Template Engine, Artisan, Migration system for databases . . . 2. Libraries & Modular (Thư viện và Mô dun) 3. Chúng hỗ trợ kiến trúc MVC 4. Unit Testing 5. Bảo mật 6. Trang web được xây dựng trong Laravel có khả năng mở rộng và bảo mật hơn. 7. Nó bao gồm các namspacevà interface giúp tổ chức tất cả các tài nguyên. 8. Cung cấp API clean | |
| 20 | **Những gì mới trong Laravel 7x?** | |
|  |  | |
| 21 | **Package trong Laravel là gì ?** | |
|  | Package Laravel là một cách chính để thêm chức năng vào Laravel. Packages có thể chứa routes, controllers, views và cấu hình đặc biệt nhằm tăng cường tính ứng dụng ở nhiều project Laravel khác nhau. Có rất nhiều Package có sẵn hiện nay và laravel có một số Package chính thức được đưa ra dưới đây:   * Cashier * Dusk * Envoy * Passport * Socialite * Telescope | |
| 22 | **Validation trong laravel và cách sử dụng ?** | |
|  | Validation là một điều quan trọng nhất trong khi thiết kế một ứng dụng. Nó xác nhận dữ liệu đến. Nó sử dụng ValidatesRequests cung cấp một phương thức thuận tiện để xác thực các yêu cầu HTTP đến với các quy tắc xác thực mạnh mẽ. Dưới đây là một số Quy tắc xác thực có sẵn trong Laravel được liệt kê:   * Alpha * Image * Date Format * IP Address * URL * Numeric * Email * Size * Min , Max * Unique with database | |
| 23 | **Làm sao để kế thừa file view trong Laravel** | |
|  | Với [@extends](https://viblo.asia/u/extends) này ('layouts.master'), chúng ta có thể mở rộng master layout này trong bất kỳ tệp xem nào. Trong ví dụ này, layouts là một thư mục được đặt trong resources/views có sẵn và file master sẽ ở đó. Bây giờ "master.blade.php" là một layout file. | |
| 24 | **Làm thế nào để chuyển hướng từ controller sang view trong laravel ?** | |
|  | return redirect('/')->withErrors('You can type your message here');  return redirect('/')->with('variableName', 'You can type your message here');  return redirect('/')->route('PutRouteNameHere'); | |
| 25 | **Làm thế nào để tạo một hằng số và sử dụng globally?** | |
|  | Bạn có thể tạo một constants.php page/file trong thư mục cấu hình nếu không tồn tại. Bây giờ bạn có thể đặt biến không đổi có giá trị ở đây và có thể sử dụng với Config :: get ('constants.VaribleName');  return [  'ADMINEMAIL' => 'info@bestinterviewquestion.com',  ];  Now we can display with  Config::get('constants.ADMINEMAIL'); | |
| 26 | **Làm sao để bật Log Query trong Laravel ?** | |
|  | * DB::connection()->enableQueryLog(); * $querieslog = DB::getQueryLog(); * dd($querieslog)   DB::connection()->enableQueryLog();  $result = User:where(['status' => 1])->get();  $log = DB::getQueryLog();  dd($log); | |
| 27 | **Làm sao để lấy route name hiện tại** | |
|  | request()->route()->getName() | |
| 28 | **Làm sao để tạo model controller và migration trong một câu lệnh laravel ?** | |
|  | php artisan make:model ModelNameEnter -mcr | |
| 29 | **MVC trong laravel là gì?** | |
|  |  | |
| 30 | **Tinhker là gì?** | |
|  | Laravel Tinker is a powerful REPL tool which is used to interact with Laravel application with the command line in an interactive shell. Tinker came with the release of version 5.4 is extracted into a separate package. How to install tinker composer require laravel/tinker How to execute To execute tinker we can use php artisan tinker command. | |
| 31 | Namespace trong laravel là gì? | |
|  |  | |
| 32 | ​​Mẫu thiết kế Singleton trong laravel là gì? | |
|  | Singleton pattern thuộc về Creational Design Pattern là một mẫu áp dụng cho việc khởi tạo đối tượng, áp dung pattern này khi ứng dụng của bạn muốn tạo ra một thực thể duy nhất từ một class và dùng chung nó cho nhiều trường hợp. Ví dụ, website cần một đối tượng kết nối đến database nhưng chỉ cần duy nhất một đối tượng cho toàn bộ ứng dụng, sử dụng Singleton Pattern sẽ giải quyết được vấn đề này. Để bắt đầu bạn sử dụng một thuộc tính static để đảm bảo rằng chỉ có một thực thể của lớp này tồn tại. | |
| 33 | **Cách truyền nhiều biến từ controller sang blade** | |
|  | $variable = 'Best';  $variable = 'Interview';  $variable = 'Question';  return view('frontend.index', compact('variable', variable', variable'));  Trong file view bạn có thể hiện thị bằng cú pháp {{ $variable }} hoặc {{ $variable }} hoặc {{ $variable }} | |
| 34 | **Làm thế nào để upload files trong Laravel ?** | |
|  | ...  if($request->hasFile(‘img’)) {  $imageName = time() . '.' . $request->img->getClientOriginalExtension();  $request->img->move(public\_path('images'), $imageName);  } | |
| 35 | **Làm sao để truyền một tên bảng custom trong model ?** | |
|  | Chúng ta truyền vào một biến protected $table = 'TÊN BẢNG CỦA BẠN'; trong Model.  namespace App;  use Illuminate\Database\Eloquent\Model;  class Login extends Model  {  protected $table = 'admin';  static function logout() {  if(session()->flush() || session()->regenerate()) {  return true;  }  }  }  **Nếu chúng ta không truyền biến $table thì Laravel mặc định đặt tên bảng theo quy tắc Tên model + 's'** Ví dụ : Model là Login thì tên bảng là logins | |
| 36 | **Custom validation rule trong Laravel?** | |
|  | 1. Chạy lệnh php artisan make:rule OlympicYear 2. Sau khi chạy lệnh nó sẽ sinh ra cho mình một file có đường dẫn app/Rules/OlympicYear.php 3. Chúng ta có thể viết rule trong hàm passes() của class OlympicYear.php. Nó sẽ trả về true hoặc false tùy điều điện,   public function passes($attribute, $value)  {  return $value >= 1896 && $value <= date('Y') && $value % 4 == 0;  }   1. Tiếp đó chúng ra có thể tùy chỉnh thông báo lỗi với hàm message()   public function message()  {  return ':attribute should be a year of Olympic Games';  }   1. Cuố cùng ở controller ta xử lí như sau   public function store(Request $request)  {  $this->validate($request, ['year' => new OlympicYear]);  } | |
| 37 | **Sử dụng session trong Laravel?** | |
|  | 1. Lấy dữ liệu từ session   session()->get('key');   1. Lấy toàn bộ dữ liệu từ session   session()->all();   1. Xóa dữ liệu từ session   session()->forget('key'); or session()->flush();   1. Lưu dữ liệu vào session   session()->put('key', 'value'); | |
| 38 | **Soft delete trong Laravel sử dụng như thế nào?** | |
|  | **Soft delete** là một tính năng của Laravel giúp khi Model muốn xóa mềm một bản ghi. Tức là bản khi không thực sự bị xóa khỏi cơ sở dữ liệu. Thay vào đó, 1 cột deleted\_at sẽ được thiết lập. Khi bật soft deletes cho 1 model. Chúng ta phải chỉ định thuộc tính softDelete trong model chúng ta sử dụng dụng namespace use Illuminate\Database\Eloquent\SoftDeletes; và chúng ta có thể sử dụng use SoftDeletes;trong model của chúng ta.  Sau đó chúng ta sẽ sử dụng truy vấn delete() thì các bản ghi sẽ không xóa khỏi cơ sở dữ liệu của chúng ta. Sau đó deleted\_at đã được thiết lập trên bản ghi. | |
| 39 | **Thêm nhiều điều kiện AND trong truy vấn Laravel như nào?** | |
|  | Chúng ta có thể thêm nhiều toán tử AND tại một điều kiện where() single cũng như thêm nhiều điện kiện where trong một truy vấn điều kiện  DB::table('client')->where('status', '=', 1)->where('name', '=', 'bestinterviewquestion.com')->get();  DB::table('client')->where(['status' => 1, 'name' => 'bestinterviewquestion.com'])->get(); | |
| 40 | **Cách Sử dụng join trong Laravel?** | |
|  | DB::table('admin')  ->join('contacts', 'admin.id', '=', 'contacts.user\_id')  ->join('orders', 'admin.id', '=', 'orders.user\_id')  ->select('users.id', 'contacts.phone', 'orders.price')  ->get(); | |
| 41 | **Eloquent ORM trong Laravel là gì ?** | |
|  | ORM(Object Relational Mapping) đây là tên gọi chỉ việc ánh xạ các record dữ liệu trong hệ quản trị cơ sở dữ liệu sang dạng đối tượng mà mã nguồn đang định dạng trong class. Đây là tính năng rất quan trọng và được framework laravel hỗ trợ. Laravel cho phép làm việc với các đối tượng và quan hệ trong CSDL thông qua Eloquent. Mỗi bảng sẽ ứng với 1 Model riêng, và việc thao tác tới các bảng đó trong ứng dụng Laravel sẽ thông qua Model tương ứng đó Có nhiều loại quan hệ:   1. Quan hệ 1 - 1 2. Quan hệ 1 - n 3. Quan hệ nhiều nhiều 4. . . . | |
| 42 | **Cách sử dụng cookies trong Laravel** | |
|  | 1. Set giá trị cookie: Cookie::put('key', 'value'); 2. Lấy giá trị cookie: Cookie::get('key'); 3. Xóa giá trị cookie: Cookie::forget('key') 4. Check cookie có tồn tại hay không: Cookie::has('key') | |
| 43 | **Bật tắt chế độ maintenance trong Laravel** | |
|  | // Enable maintenance mode  php artisan down  // Disable maintenance mode  php artisan up | |
| 44 | **Hàm dd() trong Laravel** | |
|  | Đây là một hàm trợ giúp được sử dụng để chuyển nội dung của biến sang trình duyệt và dừng thực thi tập lệnh tiếp theo. Nó là viết tắt của Dump and Die.  dd($array); | |
| 45 | **Thuộc tính fillable trong Model là gì?** | |
|  | Nó là một mảng chứa tất cả các trường của bảng có thể được tạo trực tiếp bản ghi mới trong bảng Cơ sở dữ liệu của bạn.  class User extends Model {  protected $fillable = ['username', 'password', 'phone'];  }  Việc khai báo $fillable sẽ giúp ta Mass Assignment khi tạo một bản ghi mới trong Laravel. | |
| 46 | **Lấy ra thông tin người dùng khi họ đăng nhập với Auth?** | |
|  | use Illuminate\Support\Facades\Auth;  $userinfo = Auth::user();  print\_r($userinfo ); | |
| 47 | **Laravel có hỗ trợ caching?** | |
|  | Có, nó hỗ trợ caching như Memcached và Redis. Theo mặc định, laravel được cấu hình với caching lưu trữ các đối tượng được lưu nối tiếp, được lưu trong các tệp. Thông thường chúng ta có thể sử dụng Memcached hoặc Redis cho các dự án lớn. | |
| 48 | **Sự khác nhau giữa {{ $username }} và {!! $username !!}** | |
|  | {{ $username }} chỉ được sử dụng để hiển thị nội dung văn bản nhưng {!! $username !!} được sử dụng để hiển thị nội dung với các thẻ HTML nếu tồn tại. | |
| 49 | **Làm thế nào để kiểm tra nếu giá trị được gửi trong request?** | |
|  | if($request->has('email')) {       // email value is sent from request  } else {      // email value not sent from request  } | |
| 50 | **Các cơ sở dữ liệu hỗ trợ Laravel** | |
|  | * MySQL 5.6+ ([Version Policy](https://en.wikipedia.org/wiki/MySQL#Release_history)) * PostgreSQL 9.4+ ([Version Policy](https://www.postgresql.org/support/versioning/)) * SQLite 3.8.8+ * SQL Server 2017+ ([Version Policy](https://support.microsoft.com/en-us/lifecycle/search)) | |
| 51 | **Lấy ra địa chỉ IP người dùng trong Laravel** | |
|  | Bạn có thể sử dụng request()->ip(). Hoặc bạn có thể dùng Request::ip() nhưng với trường hợp này bạn phải gọi namespace là Illuminate\Support\Facades\Request. | |
| 52 | **Làm thế nào để tạo khóa ứng dụng trong laravel?** | |
|  | php artisan key:generate | |
| 53 | **Khoá ứng dụng trong Laravel là gì?** | |
|  | APP\_KEY là một chuỗi 32 ký tự ngẫu nhiên được lưu trữ trong biến APP\_KEY trong file .env. Khi tạo 1 project mới, key này sẽ tự động được tạo ra. Nhưng nếu bạn clone hoặc copy 1 project thì sẽ không có key này.  Laravel sử dụng APP\_KEY được sử dụng để giữ cho cookie và dữ liệu được mã hóa khác an toàn , bao gồm session cookie, trước khi chuyển chúng cho trình duyệt của người dùng và sử dụng nó để giải mã cookie đọc từ trình duyệt. | |
| 54 | **Làm thế nào để bảo mật ứng dụng web** | |
|  | Bảo mật đăng nhập ứng dụng web.  + Tạo middleware, đăng ký middleware ở App/Http/Kernel.php, xác thực người dùng bằng cách Auth::check(), nếu như điều này đúng thì sẽ cho đi tiếp, còn ko thì quay sang trang khác  + Auth::user() để lấy thông tin người dùng đăng nhập  + Auth::attempt để kiểm tra thông tin người dùng nhập được gửi lên ứng dụng web.  + Auth::logout(); người dùng sẽ thoát khỏi đưang nhập hiện tại.  + Auth::guard('admin')->attempt kiểm tra thông tin của guard admin  + Sử dụng Hash::make($request->newPassword) để tạo mã băm bảo mật cho mật khẩu người dùng.  Bao mat JRSF | |
| 55 | | **JWT là gì? Cơ chế bảo mật jwt** |
|  | [**JWT**](https://topdev.vn/blog/?s=JSON+Web+Token) là một phương tiện đại diện cho các yêu cầu chuyển giao giữa hai bên Client – Server , các thông tin trong chuỗi **JWT** được định dạng bằng [JSON](https://topdev.vn/blog/json-la-gi/) . Trong đó chuỗi Token phải có 3 phần là header , phần payload và phần signature được ngăn bằng dấu “.” Cấu trúc của JSON Web Token: Như ở trên đã nói JSON Web Token bao gồm 3 phần, được ngăn cách nhau bởi dấu chấm (.):   1. Header 2. Payload 3. Signature (chữ ký)  **Header** Phần **header** sẽ chứa kiểu dữ liệu , và thuật toán sử dụng để mã hóa ra chuỗi JWT  {  "typ": "JWT",  "alg": "HS256"  }   * **“typ” (type)** chỉ ra rằng đối tượng là một **JWT** * **“alg” (algorithm)** xác định thuật toán mã hóa cho chuỗi là **HS256**  **Payload** Phần **payload** sẽ chứa các thông tin mình muốn đặt trong chuỗi  **Token** như **username** , **userId** , **author** , … ví dụ:  {  "user\_name": "admin",  "user\_id": "1513717410",  "authorities": "ADMIN\_USER",  "jti": "474cb37f-2c9c-44e4-8f5c-1ea5e4cc4d18"  } **Signature** Phần chử ký này sẽ được tạo ra bằng cách mã hóa phần **header** , **payload** kèm theo một chuỗi **secret (khóa bí mật)** , ví dụ:  data = base64urlEncode( header ) + "." + base64urlEncode( payload )  signature = Hash( data, secret );   * **base64UrlEncoder :** thuật toán mã hóa **header** và **payload**   Đoạn code trên sau khi mã hóa **header** và **payload** bằng thuật toán **base64UrlEncode** ta sẽ có chuỗi như sau  // header  eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9  // payload  eyJhdWQiOlsidGVzdGp3dHJlc291cmNlaWQiXSwidXNlcl9uYW1lIjoiYWRtaW4iLCJzY29wZSI6WyJyZWFkIiwid3JpdGUiXSwiZXhwIjoxNTEzNzE  Sau đó mã hóa 2 chuỗi trên kèm theo **secret (khóa bí mật)** bằng thuật toán **HS256** ta sẽ có chuỗi **signature** như sau:  9nRhBWiRoryc8fV5xRpTmw9iyJ6EM7WTGTjvCM1e36Q | |
| 56 | **web service là gì?** | |
|  | Một cách ngắn gọn thì web services (hay "dịch vụ web") là một mô-đun phần mềm được thiết kế để thực hiện một nhóm các tác vụ nhất định. Các web services có thể được truy cập và sử dụng thông qua mạng internet dưới dạng dịch vụ. Khi đó, web service sẽ cung cấp các chức năng của nó cho máy khách để người dùng đạt được các mục tiêu sử dụng nhất định. | |
| 57 | **API là gì? cách tạo API** | |
|  | **API** là các phương thức, giao thức kết nối với các thư viện và ứng dụng khác. Nó là viết tắt của **Application Programming Interface** – giao diện lập trình ứng dụng. API cung cấp khả năng cung cấp khả năng truy xuất đến một tập các hàm hay dùng. Và từ đó có thể trao đổi dữ liệu giữa các ứng dụng. | |
| 58 | **Trình bày cơ chế dependency injection** | |
|  | 1. Dependency injection là gì? | |
|  | * Dependency injection là một kĩ thuật khi mà một Object phụ thuộc vào một Object khác. Dependecy trong lập trình có thể hiểu như Khi class A sử dụng một phương thức của class B, khi đó người ta nói Class A phụ thuộc vào class B. Vì vậy khi cần 1 Object có dependency với các object khác, Object đó sẽ gọi 1 service khác, service đó có nhiệm vụ khỏi tạo các Object phụ thuộc, và đó là cơ chế của Dependency Injection.  3. Các loại Dependency injection:  * Constructor Injection: Sử dụng Dependency thông qua hàm khởi tạo Constructor * Setter injection: Sử dụng phương thức Setter để chèn sự phục thuộc. * Interface injection: Dependency được cung câp bởi phương thức sẽ chèn Dependency vào bất kì class nào được nó truyền vào, class phải thực thi một giao diện(Interface) để chấp nhận dependency đó.   Nếu có bất kì thay đổi object phụ thuộc nào trong class, thì DI sẽ có trách nhiệm cung cấp các đối tượng phù hợp cho lớp đó. | |
| 59 | **Kiểm thử là gì?** | |
|  | Kiểm thử hay còn gọi là testing, là quá trình đánh giá một hệ thống hay các thành phần của nó với mục đích tìm xem liệu hệ thống có đáp ứng các yêu cầu được đã được chỉ định hay không. Nói một cách đơn giản, kiểm thử được thực hiện trên một hệ thống để xác định bất kỳ lỗ hổng, các lỗi hoặc các yêu cầu đang bị thiếu hay trái ngược với các yêu cầu thực tế đã được đề ra. | |
| 60 | **Tại sao cần kiểm thử phần mềm?** | |
|  | Kiểm thử phần mềm là cần thiết vì không có gì là hoàn hảo , tất cả chúng ta đều có thể mắc lỗi ở những mức độ khác nhau . Một số những sai lầm là không quan trọng, nhưng một số khác là rất nghiêm trọng bạn sẽ phải mất rất nhiều tiền hoặc nguy hiểm bởi lỗi lầm đấy. Chúng ta cần phải kiểm tra mọi thứ và bất cứ thứ gì chúng ta sản xuất ra bởi vì mọi thứ luôn có thể sai - con người luôn phạm sai lầm.  Vì vậy tôi cho rằng công việc của chúng ta dù ít hay nhiều cũng có thể có những sai lầm, do đó tất cả chúng ta đều cần phải kiểm tra công việc của mình. Thường một số sai lầm đến từ các giả định xấu và các điểm mù, vì vậy chúng ta có thể mắc phải những sai lầm tương tự khi chúng ta không thực hiện kiểm thử, hoặc tự mình kiểm thử công việc của mình. Việc tự kiểm tra sẽ làm cho chúng ta không thể nhận thấy những sai sót, vì mặc định chúng ta đã làm đúng các yêu cầu đúng không ? | |
| 61 | **TDD là gì?** | |
|  | TDD (Test Driven Development- Phát triển hướng kiểm thử TDD ) là một phương thức làm việc, hay một quy trình viết mã hiện đại. Lập trình viên sẽ thực hiện thông qua các bước nhỏ (BabyStep) và tiến độ được đảm bảo liên tục bằng cách viết và chạy các bài test tự động (automated tests). Quá trình lập trình trong TDD cực kỳ chú trọng vào các bước liên tục sau: Viết 1 test cho hàm mới. Đảm bảo rằng test sẽ fail. Chuyển qua viết code sơ khai nhất cho hàm đó để test có thể pass. Tối ưu hóa đoạn code của hàm vừa viết sao cho đảm bảo test vẫn pass và tối ưu nhất cho việc lập trình kế tiếp Lặp lại cho các hàm khác từ bước 1 Thực tế, nên sử dụng UnitTestFramework cho TDD (như JUnit trong Java), chúng ta có thể có được môi trường hiệu quả vì các test được thông báo rõ rang thông qua màu sắc: Đỏ: test fail, chuyển sang viết function cho test pass Xanh lá: viết một test mới hoặc tối ưu code đã viết trong màu đỏ. | |
| 62 | **Validation trong laravel và cách sử dụng ?** | |
|  | - Validation là tiến trình kiểm tra cái gì đó có tuân theo một qui tắc đã cho không. Và cụ thể đối với form nó là các ràng bộc về dữ liệu của form nhằm đảm bảo tính chặt chẽ của ứng dụng.  Tạo request form validation  - php artisan make:request StoreBlogPost  public function rules()  {  return [  'title' => 'required|unique:posts|max:255',  'body' => 'required',  ];  }  Ta viết trong hàm rules các quy tắc và tiêm StoreBlogPost vào phương cân ta cần validation.  public function authorize()  {  return true;  }  Để cho quyền truy cập vào phương thức này. | |
| 63 | **Làm sao để chạy được TestCase trong Laravel ?** | |
|  | **Sử dụng phpunit để TestCase**  **composer require --dev phpunit/phpunit ^7** | |
| 64 | **Laravel Passport là gì?** | |
|  |  | |
| 65 | **Làm thế nào để gọi Middleware trong Laravel ?** | |
|  | **Middleware** như  là một cơ chế cho phép bạn tham gia vào luồng xử lý request của một ứng dụng Laravel. Trong một quá trình xử lý route điển hình của Laravel khi thực thi việc xử lý yêu cầu và middleware là một trong những class mà ứng dụng phải thông qua.  php artisan make:middleware MobileRedirect  Đăng ký middleware vào App/Http/Kernel.php | |
| 66 | **Làm thế để clear cache trong Laravel ?** | |
|  | **Please run below artisan commands step wise step.**   * php artisan config:clear * php artisan cache:clear * composer dump-autoload * php artisan view:clear * php artisan route:clear | |
| 67 | **Làm thế nào để ký service provider trong Laravel?** | |
|  |  | |
| 68 | **API RESTful là gì và tại sao nó được sử dụng?** | |
|  |  | |
| 69 | **Làm thế nào để sử dụng lại một file view trong Laravel?** | |
|  |  | |
| 70 | **Chạy composer dump-autoload thì điều gì xảy ra?** | |
|  | Tự động load tất cả các class để sau này ta dùng các class mà ko cần include các class này. | |
| 71 | **JQuery là gì?** | |
|  | **jQuery** là thư viện javascript nổi bật nhất, phổ biến nhất trong lập trình web. Ra đời vào năm 2006 bởi John Resig. Nó đã trở thành một thành phần không thể thiếu trong các website có sử dụng Javascript. Với slogan “Write less – Do more” (viết ít hơn – làm nhiều hơn) nó đã giúp lập trình viên tiết kiệm được rất nhiều thời gian và công sức trong việc thiết kế website. | |
| 72 | **Ajax là gì? Cách hoạt động** | |
|  | AJAX là chữ viết tắt của cụm từ **Asynchronous Javascript and XML**. AJAX là phương thức trao đổi dữ liệu với máy chủ và cập nhật một hay nhiều phần của trang web, hoàn toàn không reload lại toàn bộ trang.  Ajax được viết bằng Javascript chạy trên client, tức là mỗi browser sẽ chạy độc lập hoàn toàn không ảnh hưởng lẫn nhau. Về mặt kỹ thuật, nó đề cập đến việc sử dụng các đối tượng XmlHttpRequest để tương tác với một máy chủ web thông qua Javascript.  **Cách hoạt động**     1. Từ trình duyệt của client, ta có một sự kiện để gọi ajax. Khi đó javascript sẽ tạo nên một đối tượng XMLHttpRequest. Và đối tượng này sẽ được gửi một request đến server ví dụ: Khi người dùng click vào ô input và chọn skill tìm việc làm của người dùng. Khi đó ta sẽ lấy thông tin đó và gửi đến server và cần trả về các việc làm tương ứng phù hợp với người dùng. 2. Khi server nhận được HttpRequest từ đó sẽ xử lý request và trả về response cho web. Server xử lí lấy ra các việc làm javascript chẳng hạn thuộc skill mà người dùng cần và trả về dữ liệu của các việc làm. 3. Sau khi nhận được một response từ server, Javascript sẽ xử lý và cập nhật vào trang web cho chúng ta. | |
| 73 | **Phân biệt session và cookie?** | |
|  | | **Cookie** | **Session** | | --- | --- | | **Cookie** được lưu trữ trên trình duyệt của người dùng. | **Session** không được lưu trữ trong trình  duyệt của người dùng. | | Dữ liệu **cookie** được lưu trữ ở phía máy khách. | Dữ liệu **session** được lưu trữ ở  phía máy chủ. | | Dữ liệu **cookie** dễ dàng sửa đổi khi chúng được lưu trữ ở phía khách hàng. | Dữ liệu **session** không dễ dàng sửa đổi vì  chúng được lưu trữ ở phía máy chủ. | | Dữ liệu **cookie** có sẵn trong trình duyệt của chúng ta đến khi hết hạn. | Dữ liệu **session** có sẵn cho trình duyệt chạy.  Sau khi đóng trình duyệt sẽ mất thông tin **session**. | | |
| 74 | **So sánh Eloquent ORM và Query Builder?** | |
|  | **1, Tính Bảo mật:** Query builder và Eloquent đều sử dụng : 'PDO parameter binding' nên sẽ giúp chúng ta tránh được lỗi sql injection. **2, Tính tương tác:** Bạn có thể sử dụng tất cả các function của Query Builder trong Eloquent nhưng không thể sử dụng các funcation của Eloquent trong Query Builder. **3, Tính dễ sử dụng:**   * Eloquent có các hàm được laravel xây dựng sãn nên nó ngắn gọn và dễ hiểu hơn so với Query builder * Eloquent dễ dàng liên kết giữa các bảng trong database hơn Query builder * Eloquent chỉ thực hiện được những truy vấn không quá phức tạp, để thực hiện những truy vấn phức tạp cần dùng Query builder   **4, Hiệu suất:** Eloquent thực hiện các thao tác với database châmj hơn so với Query builder | |
| 75 | **Queue là gì ? Stack là gì?** | |
|  | Queue trong Laravel thực chất là một hàng đợi chứa các tác vụ cần xử lý, tác vụ nào đưa vào trước sẽ được xử lý trước, tác vụ nào đưa vào sau sẽ được xử lý sau. Và quan trọng nhất là Queue trong Laravel có thể chạy ngầm. | |
| 76 | **JSON là gì?** | |
|  | **JSON** là viết tắt của **J**ava**S**cript **O**bject **N**otation, là một kiểu định dạng dữ liệu tuân theo một quy luật nhất định mà hầu hết các ngôn ngữ lập trình hiện nay đều có thể đọc được. **JSON**là một tiêu chuẩn mở để trao đổi dữ liệu trên web. | |
| 77 | **CSRF là gì?** | |
|  | **CSRF** hay còn gọi là kỹ thuật tấn công “**Cross-site Request Forgery**“, nghĩa là kỹ **thuật tấn công giả mạo** chính chủ thể của nó. **CSRF** nói đến việc tấn công vào chứng thực request trên web thông qua việc sử dụng Cookies. Đây là nơi mà các hacker có khả năng sử dụng thủ thuật để tạo request mà bạn không hề biết. Vì vậy, một CSRF là hacker lạm dụng sự tin tưởng của một ứng dụng web trên trình duyệt của nạn nhân. | |
| 78 | **Trình các mối quan hệ trong ORM? mối quan hệ n-n thì ta làm như thế nào ?** | |
|  | Quan hệ n-n (còn gọi là quan hệ nhiều - nhiều, hay quan hệ many to many) là mối quan hệ giữa hai tập thực thể mà mỗi thực thể thuộc tập này có thể có quan hệ với nhiều thực thể của tập kia và **ngược lại**. Ví dụ, một Product có thể nằm trong nhiều Order khác nhau và một Order có thể có nhiều Product khác nhau, do đó mối quan hệ Product-Order là quan hệ n-n. Định nghĩa quan hệ này trong Eloquent sử dụng phương thức belongsToMany(). | |
| 79 | **Collections trong Laravel là gì ?** | |
|  | Collection trong Laravel là một class được tích hợp sẵn các phương thức thường xuyên được sử dụng để xử lý dữ liệu nhàm giảm thiểu tối đa thời gian cho các lập trình viên. Đặc biệt là khi làm việc với API kết nối với database vì dữ liệu từ database trả về sẵn kiểu là Collection.  Collection trong Laravel là một lớp trong lập trình hướng đối tượng, nó thay thế cho các mảng truyền thống. Giống như mảng, Collection có nhiều các item thành phần, tuy nhiên các item này có thể là các đối tượng thay cho các kiểu dữ liệu đơn giản như String, Integer | |
| 80 | **SSH là gì? Làm thế nào để cấu hình SSH?** | |
|  | SSH, hoặc được gọi là Secure Shell, là một giao thức điều khiển từ xa cho phép người dùng kiểm soát và chỉnh sửa server từ xa qua Internet.  ssh-keygen -t rsa  Tạo key public and private. | |

|  |  |
| --- | --- |
| app | Thư mục app, chứa tất cả các project được tạo, hầu hết các class trong project được tạo đều ở trong đây. Không giống các framwork khác, các file model không được chứa trong một thư mục riêng biệt, mà được chứa ngay tại thư mục app này. |
| app/Console | Thư mục Console, chứa các tập tin định nghĩa các câu lệnh trên artisan. |
| app/Exceptions | Thư mục Exceptions, chứa các tập tin quản lý, điều hướng lỗi. |
| app/Http/Controllers | Thư mục Controllers, chứa các controller của project. |
| app/Http/Middleware | Thư mục Middleware, chứa các tập tin lọc và ngăn chặn các requests. |
| app/Providers | Thư mục Providers, chứa các file thực hiện việc khai báo service và bind vào trong Service Container. |
| bootstrap | Thư mục bootstrap, chứa những file khởi động của framework và những file cấu hình auto loading, route, và file cache. |
| config | Thư mục config, chứa tất cả những file cấu hình. |
| database | Thư mục database, chứa 2 thư mục migration (tạo và thao tác database) và seeds (tạo dữ liệu mẫu), tiện lợi để lưu trữ dữ liệu sau này. |
| database/factories | Thư mục factories, chứa các file định nghĩa các cột bảng dữ liệu để tạo ra các dữ liệu mẫu. |
| database/migrations | Thư mục migrations, chứa các file tạo và chỉnh sửa dữ liệu. |
| database/seeds | Thư mục seeds, chứa các file tạo dữ liệu thêm vào CSDL. |
| public | Thư mục public, chứa file index.php giống như cổng cho tất cả các request vào project, bên trong thư mục còn chứa file JavaScript, và CSS. |
| resources | Thư mục resources, chứa những file view và raw, các file biên soạn như LESS, SASS, hoặc JavaScript. Ngoài ra còn chứa tất cả các file lang trong project. |
| resources/views | Thư mục views, chứa các file view xuất giao diện người dùng. |
| routes | Thư mục routes, chứa tất cả các điều khiển route (đường dẫn) trong project. Chứa các file route sẵn có: web.php, channels.php, api.php, và console.php. |
| routes/api.php | file api.php, điều khiển các route của ứng dụng, như route của ứng dụng User (đăng ký, đăng nhập, ...). |
| routes/web.php | file web.php, điều khiển các route của view, như route của trang top, sản phẩm, ... |
| storage | Thư mục storage, chứa các file biên soạn blade templates của bạn, file based sessions, file caches, và những file sinh ra từ project.   * Thư mục app, dùng để chứa những file sinh ra từ project. * Thư mục framework, chứa những file sinh ra từ framework và caches. * Thư mục logs, chứa những file logs. * Thư mục /storage/app/public, lưu những file người dùng tạo ra như hình ảnh. |
| tests | Thư mục tests, chứa những file tests, như PHPUnit test. |
| vendor | Thư mục vendor, chứa các thư viện của Composer. |
| .env | file .env, chứa các config chính của Laravel. |
| artisan | file thực hiện lệnh của Laravel. |
| .gitattributes .gitignore | File dành cho xử lý git. |
| composer.json composer.lock composer-setup.php | File của Composer. |
| package.js | file package.js, chứa các package cần dùng cho projects. |
| phpunit.xml | file phpunit.xml, xml của phpunit dùng để testing project. |
| webpack.mix.js | file webpack.mix.js, file dùng để build các webpack. |