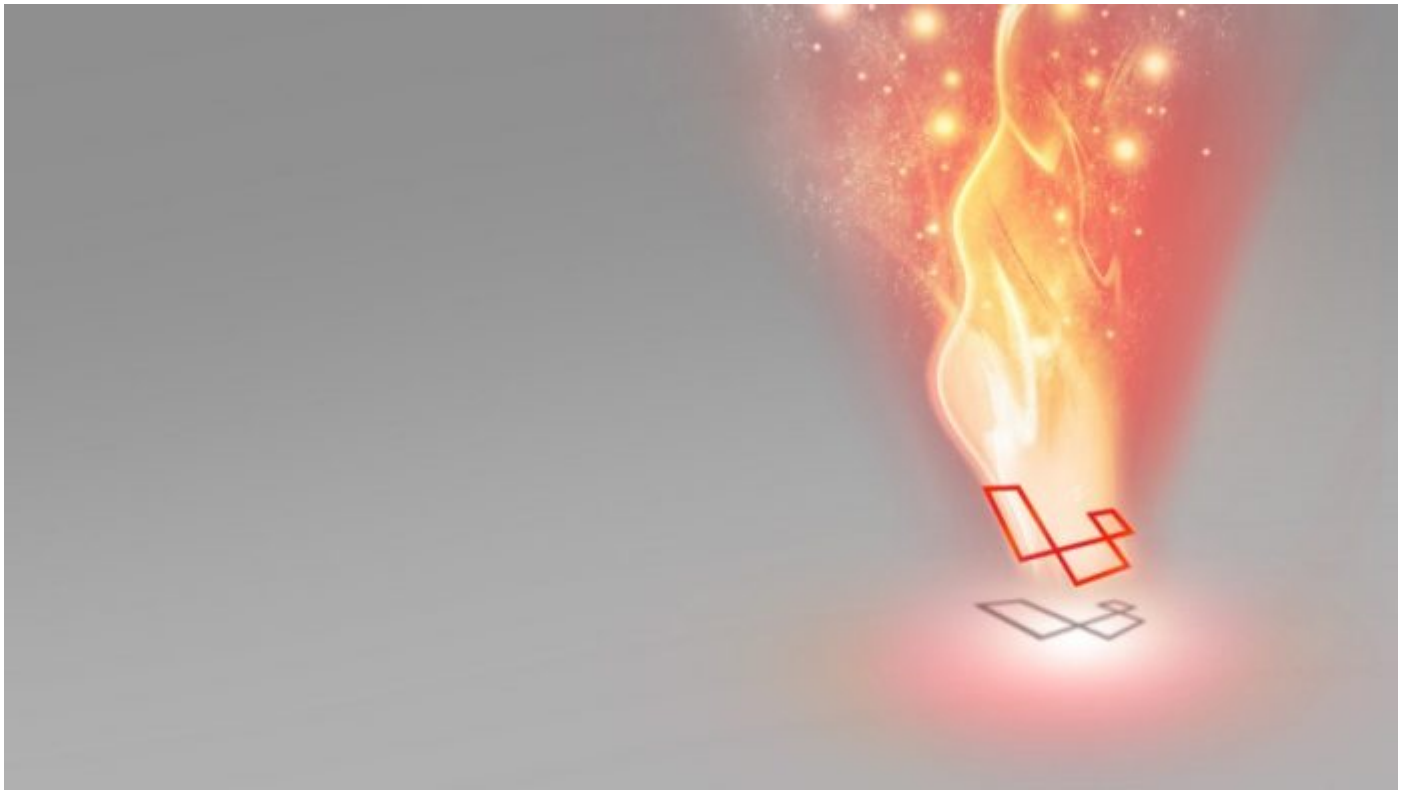
LARAVEL bạn đã viết đúng?

July 20, 2017



Chào các bạn trong bài trước mình đã chia sẽ cho các bạn những lời khuyên để học nhanh Laravel rồi phải không nào.

Qua nhiều lần khảo sát cũng như trao đổi với nhiều bạn làm về Laravel mình nhận thấy đa phần các bạn đều viết sai cách. Sai ở đây mình không nói về Logic mà mình nói về cách viết trong OOP (Hướng đối tượng).

Hôm nay mình viết bài này để chia sẽ cho các bạn code Laravel sao cho đúng (ý kiến cá nhân mình ^^) nhằm tăng performance website cũng như tính thẩm mỹ trong code.

Nói đến hướng đối tượng thì không riêng gì Laravel mà các Framework cung như thế. MVC là gì mình không cần phải nói ai cũng biết phải không nào ?. Nhưng liệu chúng ta có hiểu được mỗi thành phần trong đó xử lý những gì không nào. Đây là cái mình sẽ nói chính trong bài viết này.

1. Sử Dụng MVC theo đúng nghĩa !

Qua nhiều lần trao đổi với nhiều bạn. đa phần các bạn viết sai công dụng của các thành trên. Controller nhiều bạn gọi Models và viết cả câu query để truy vấn dữ liệu như vậy là sai nhé các bạn. viết như thế sẽ không còn mô hình MVC nữa nhé.

Vậy viết sao cho đúng nghĩa MVC ?

Với Models chúng ta nên viết function xử lý trong chính Class bạn muốn lấy dử liệu. Sau khi có fucntion mong muốn Controller chỉ cần đều hướng tới Models gọi function can thiết và trả về dữ liệu. Các bạn hãy xem qua ví dụ sau.

```
1  //Models
2
3  Class User extends Model
4  {
5       protected $table = 'users';
6
7
8       public function getListUser()
9       {
10         return self::get();
11      }
12 }
13
14 //Controllers
15
```

```php
use App\User;

Class UserController extends Controller
{
    function __construct(User $user)
    {
        $this->user = $user;
    }

    public function index()
    {
        $listUser = $this->user->getListUser();
        return view('viewName', compact('listUser'));
    }
}
```

Khá là đơn giản phải không nào. Trong ví vụ trên tôi có dùng __construct trong việc khởi tạo models. Nhiều bạn vẫn thường dùng New ModelClass đều này không sai nhưng việc cứ lặp đi lặp lại mỗi function bạn phải gọi new Model nhin code không đẹp cho lắm. Thì __construct sẽ giúp các bạn trong việc này.

Qua ví vụ trên các bạn thấy code có ngắn gọn hơn không nếu code bạn cũng đang mắt phải tình trạng mình vừa nói thì hãy thử theo cách của mình nhé.

2. Request hãy để nó làm việc theo đúng những gì nó có !

Cũng như Models. Request sẽ quản lý các xữ lý liên quan gữi và nhận yêu cầu từ phía người dùng (Views). Nhiều bạn vẫn dùng nó ngay tại controller đều này không sai. những nó sẽ dẫn tới việc lặp đi lặp lại làm cho function chức năng của chúng ta như một mớ hỗn độn khó mà quản lý.

Cách giải quyết hợp lý nhất là ta nên xây dựng một nơi để quản lý vấn đề này bằng cách tạo một Request riêng. Với Laravel chúng đã tích hợp sẵn lệnh tạo một Request thông qua Artisan Console các bạn có thể thực hiện như sau:

```
php artisan make:request ten_request
```

Đây là nội dung Request ta vừa tạo. Tôi đã tạo một file Request có tên là UserRequest:

```php
//app/Http/Requests/UserRequest;

namespace App\Http\Requests;

use App\Http\Requests\Request;

class UserRequest extends Request
```

```
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return false;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
            //
        ];
    }
}
```

Chúng ta sẽ làm gì với nó. Nhiều bạn chua sử dụng qua chắc chắn sẽ bối rối phải không nào. tại đây chúng ta sẽ xử lý Validate input từ form được gửi lên từ người dùng. mà những Rules validate này đã có sẵn trong Laravel Doc. Các bạn có thể xem qua ví vụ sau:

```
/**
 * Get the validation rules that apply to the request.
 *
 * @return array
 */
public function rules()
{
    return [
        'username' => 'required'
    ];
}
```

Khá quen thuộc phải không nào. Không dừng ở đó tại đây các bạn có thể custom được cả thuộc tính Attribute của input và custom được message sẽ hiện thị khi bắt lỗi

```
/**
 * Get the validation attributes that apply to the request.
 *
 * @return array
 */
public function attributes()
{
```

```
8       return [
9           'username' => 'Tên người dùng'
10      ];
11  }
12  /**
13   * Get the validation custom message that apply to the request.
14   *
15   * @return array
16   */
17  public function customMessages()
18  {
19      return [
20          'username.required' => 'Tên người dùng bắt buộc nhập',
21      ];
22  }
```

Vậy sử dụng Request này như thế nào. Đơn giản thôi tại controller ban muốn sử dụng các bạn hãy gọi NameSpace của nó ra để khái báo rằng bạn sẽ sử dụng nó.

```
1  use App\Http\Requests\UserRequest;
2
3  Class UserController extends Controller
4  {
5      //
6  }
```

Tiếp theo chúng ta cần làm gì. Truyền tham chiếu nó vào action bạn cần xử lý Request. Ví vụ tôi có action đăng ký người dùng như sau:

```
1  use App\Http\Requests\UserRequest;
2
3  Class UserController extends Controller
4  {
5      //
6
7      public function userRegister(UserRequest $request)
8      {
9          // your code here.
10     }
11 }
```

Ở function userRegister các bạn sẽ không cần phải gọi Validator rồi make rules các kiểu nữa vì UserRequest đã thay bạn làm cả rồi.^^

3. Tạo Helper hãy vận dụng OOP thông qua Namespace

Nhiều bạn mắc lỗi nho nhỏ khi tạo thư viên helper cua mình nhằm giúp xử lý các tác vụ nhỏ khi cần thiết nhu xử lý chuỗi, mảng ,...Mọi người thường xử lý theo dạng php thuần trong khi các bạn đang

viết OOP.

```
1 funtion getString($string)
2 {
3     return $string;
4 }
```

Đây cũng là một cách. nhưng thật sự không tối ưu khi chỉnh sửa và thường gặp lỗi với autoload khi các bạn thêm vào composer.jon autoload. ý kiến cá nhân của mình các bạn nên tận dụng cái mà OOP đang làm cách mà Laravel đang sử dụng. mọi thứ đều viết với dạng các lớp , thuộc tính và phương thức.

```
1 namespace App\Helpers;
2 class Helper
3 {
4     public function getString($string)
5     {
6         return $string
7     }
8 }
```

Việc sử dụng cũng khá dễ dàng các bạn chỉ cần khai báo namespace tại controller và gọi phương thức của nó ra.

```
1  use App\Helpers\Helper;
2
3  Class UserController extends Controller
4  {
5      function __construct(Helper $helper)
6      {
7          $this->helper = $helper;
8      }
9
10     public function userRegister(UserRequest $request)
11     {
12         $this->helper->getString($string);
13     // your code here.
14     }
15 }
```

Lời kết:

trên chỉ là ý kiến cá nhân của mình. mọi góp ý các bạn có thể comment bên dưới. Hy vọng bài viết giúp ích cho các bạn trong việc học tập cũng như rèn luyện code của mình.

Còn khá nhiều thủ thuật khác mình sẽ đề cập trong bài viết tiếp theo nhé!. Chúc các bạn thành công. Nếu thấy bổ ích các bạn hãy share cho bạn bè nhé.

Techtalk Via codevivu