






LARAVEL FRAMEWORK

BÀI 8: MAIL, ERRORS & LOGGING

- ⊙ Giới thiệu mail
- ⊙ Gửi mail
- ⊙ Errors & Logging
- ⊙ The Exception Handler



Phần I: Tổng quan Email

-  Giới thiệu email
-  Cấu hình gửi mail
-  Cơ chế mail

Phần II: The Exception Handler

-  Quản lý Error
-  Quản lý Log
-  Exception Handler



- ❑ Việc gửi mail trong laravel đơn giản thông qua các API trong thư viện SwiftMailer.
- ❑ Cơ bản Laravel cung cấp driver cho SMTP, Mailgun, Mandrill, SparkPost, Amazon SES, hàm mail của PHP, và sendmail, cho phép gửi mail qua dịch vụ mail local hoặc cloud



- ❖ Các driver dựa trên API như Mailgun hay Mandrill thường đơn giản và nhanh hơn SMTP server
- ❖ Tất cả các API driver yêu cầu sử dụng thư viện Guzzle HTTP
- ❖ Cài đặt vào project bằng cách thêm dòng dưới đây vào trong file composer.json

```
"guzzlehttp/guzzle": "~5.3|~6.0"
```

☐ Mailgun Driver

- ☐ Để sử dụng Mailgun driver, cần cài Guzzle, rồi set giá trị driver ở trong file config/mail.php thành mailgun
- ☐ Xác nhận thông tin sau trong file config/services.php

```
'mailgun' => [  
    'domain' => 'your-mailgun-domain',  
    'secret' => 'your-mailgun-key',  
],
```

❑ Mandrill Driver

- ❑ Để sử dụng Mandrill driver, cần cài Guzzle, rồi set giá trị driver ở trong file config/mail.php thành mandrill
- ❑ Xác nhận thông tin sau trong file config/services.php

```
'mandrill' => [  
    'secret' => 'your-mandrill-key',  
],
```

❑ SparkPost Driver

- ❑ Để sử dụng Mailgun driver, cần cài Guzzle, rồi set giá trị driver ở trong file config/mail.php thành sparkpost và điều chỉnh file config/services.php

```
'sparkpost' => [  
    'secret' => 'your-sparkpost-key',  
],
```

❑ SES Driver

- ❑ Để sử dụng Amazon SES driver, cài Amazon AWS SDK cho PHP bằng cách thêm dòng dưới đây vào trong file composer.json ở mục require:

```
"aws/aws-sdk-php": "~3.0"
```

- ❑ Set giá trị driver trong file config/mail.php thành ses. Và xác nhận các thông số dưới đây trong file cấu hình config/services.php:

```
'ses' => [  
    'key' => 'your-ses-key',  
    'secret' => 'your-ses-secret',  
    'region' => 'ses-region', // e.g. us-east-1  
],
```


- ❑ Laravel cho phép lưu bản tin email trong views. Ví dụ, để quản lý emails ta có thể tạo thư mục emails bên trong thư mục resources/views.
- ❑ Để gửi một email, sử dụng hàm send trong Mail facade. Hàm send nhận ba tham số.
 - ❑ Tham số đầu tiên là tên của view mà chứa nội dung của email
 - ❑ Tham số thứ hai là một mảng data được truyền vào trong view
 - ❑ Cuối cùng là một Closure trong đó nhận một instance của message, cho phép tùy chỉnh thông tin recipients, subject và các thông tin khác của một bản tin email:

```
namespace App\Http\Controllers;

use Mail;
use App\User;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;

class UserController extends Controller
{
    /**
     * Send an e-mail reminder to the user.
     *
     * @param Request $request
     * @param int $id
     * @return Response
     */
    public function sendEmailReminder(Request $request, $id)
    {
        $user = User::findOrFail($id);

        Mail::send('emails.reminder', ['user' => $user], function ($m) use ($user) {
            $m->from('hello@app.com', 'Your Application');

            $m->to($user->email, $user->name)->subject('Your Reminder!');
        });
    }
}
```

□ Tạo bản tin

- Lưu ý tham số thứ 3 trong hàm send là một Closure cho phép xác định nhiều thông số cho email. Closure này, bạn có thể nhận vào các thuộc tính cho bản tin như CC, BCC...

```
Mail::send('emails.welcome', $data, function ($message) {  
    $message->from('us@example.com', 'Laravel');  
  
    $message->to('foo@example.com')->cc('bar@example.com');  
});
```

❑ Tạo bản tin

- ❑ Danh sách các hàm có thể sử dụng trong biến \$message:

```
$message->from($address, $name = null);  
$message->sender($address, $name = null);  
$message->to($address, $name = null);  
$message->cc($address, $name = null);  
$message->bcc($address, $name = null);  
$message->replyTo($address, $name = null);  
$message->subject($subject);  
$message->priority($level);  
$message->attach($pathToFile, array $options = []);  
  
// Attach a file from a raw $data string...  
$message->attachData($data, $name, array $options = []);  
  
// Get the underlying SwiftMailer message instance...  
$message->getSwiftMessage();
```

❑ Gửi mail dạng text

- ❑ Mặc định, view được truyền vào hàm send là dạng HTML. Tuy nhiên, bằng cách truyền tham số đầu tiên của hàm send là một mảng text, bạn có thể chỉ định một view text để gửi bổ sung với HTML view:

```
Mail::send(['html.view', 'text.view'], $data, $callback);
```

- ❑ Sử dụng từ khóa “text” khi cần mail dạng text

```
Mail::send(['text' => 'view'], $data, $callback);
```

☐ Gửi mail dạng chuỗi

- ☐ Sử dụng hàm raw nếu muốn gửi một email dạng chuỗi trực tiếp

```
Mail::raw('Text to e-mail', function ($message) {  
    //  
});
```

☐ Các file đính kèm

- ☐ Để đính kèm tài liệu trong email, sử dụng hàm attach của \$message truyền trong Closure. Tham số đầu tiên của hàm attach chứa đường dẫn đầy đủ tới file

```
Mail::send('emails.welcome', $data, function ($message) {  
    //  
    $message->attach($pathToFile);  
});
```

❑ Đính kèm inline

- ❑ Biến \$message có thể sử dụng được trong tất cả các view của email
- ❑ Để nhúng vào một ảnh inline, sử dụng hàm embed của biến \$message bên trong email view

```
<body>
    Here is an image:

    
</body>
```

- ❑ Nhúng dữ liệu thô vào trong email view: sử dụng hàm embedData cho biến

```
<body>
    Here is an image from raw data:

    
</body>
```

☐ Sử dụng queue gửi mail

- ☐ Khi ứng dụng gửi nhiều mail sẽ làm chậm thời gian response của ứng dụng
- ☐ Việc sử dụng queue cho bản tin email để gửi ở background là cần thiết
- ☐ Để queue một bản tin mail, sử dụng hàm queue trong Mail facade:

```
Mail::queue('emails.welcome', $data, function ($message) {  
    //  
});
```




DEMO

Demo cấu hình và gửi email





LARAVEL FRAMEWORK

BÀI 8: MAIL, ERRORS & LOGGING (PHẦN 2)

- ❑ Laravel cung cấp việc xử lý error và exception thông qua việc tích hợp thư viện Monolog



- ❑ Nội dung chi tiết lỗi trong ứng dụng được điều khiển bởi cấu hình debug trong file cấu hình config/app.php. Mặc định, cấu hình này thiết lập dựa trên giá trị biến môi trường APP_DEBUG, lưu trong file .env.
- ❑ Trong môi trường phát triển nội bộ nên set giá trị APP_DEBUG thành true. Trong môi trường production, giá trị này luôn luôn phải là false

❑ Các chế độ log

- ❖ Laravel hỗ trợ các chế độ log: single, daily, syslog, và errorlog. Ví dụ, nếu muốn ghi log file hàng ngày, thay vì ghi vào một file, chỉ cần thiết lập giá trị log trong file config/app.php

```
'log' => 'daily'
```

- ❖ Khi sử dụng chế độ daily, Laravel sẽ chỉ lưu trữ log files của 5 ngày. Muốn điều chỉnh số lượng file lưu trữ thì thêm vào cấu hình log_max_files vào trong app.php

```
'log_max_files' => 30
```

❑ Log Severity Levels

- ❖ Laravel hỗ trợ nhiều mức độ log khác nhau, mặc định tất cả các mức độ log được gán cho ứng dụng.
- ❖ Có thể tùy chỉnh một số mức độ log thông qua `log_level` bên trong file `app.php`
- ❖ Các mức độ log bao gồm: `debug`, `info`, `notice`, `warning`, `error`, `critical`, `alert`, `emergency`

```
'log_level' => env('APP_LOG_LEVEL', 'error'),
```

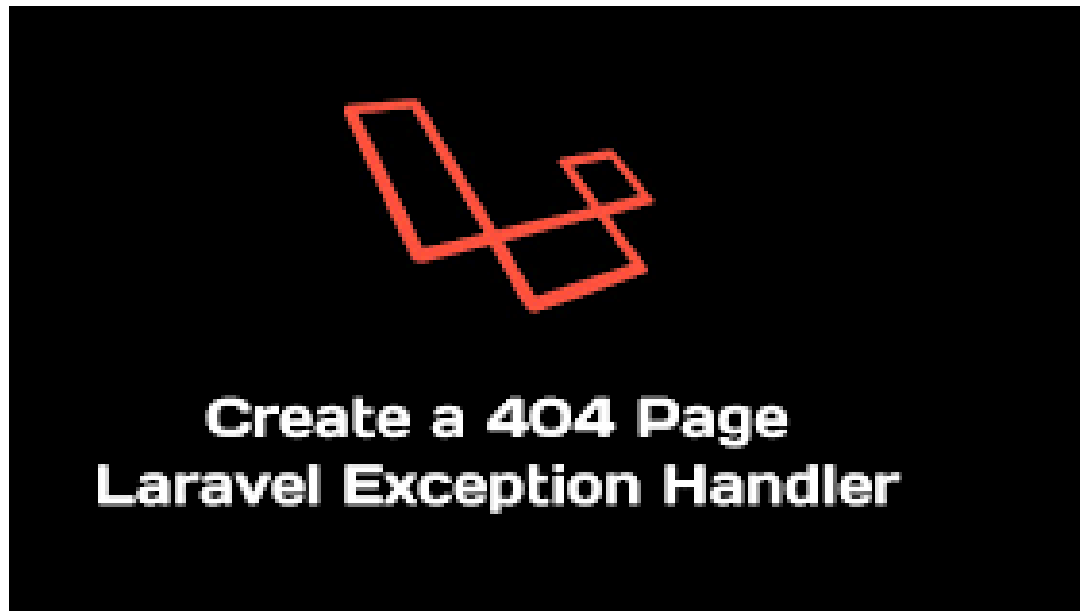
❑ cấu hình Monolog

- ❖ Để điều khiển toàn bộ quy trình Monolog cấu hình trên ứng dụng, có thể sử dụng phương thức `configureMonologUsing`.
- ❖ Nên gọi xử lý này trong file `bootstrap/app.php` ngay trước khi biến `$app` được trả về.

```
$app->configureMonologUsing(function($monolog) {  
    $monolog->pushHandler(...);  
});  
  
return $app;
```

❑ Quản lý Exception

- ❖ Tất cả các exception được xử lý bởi lớp `App\Exception\Handler`. Lớp này chứa hai phương thức: `report` và `render`.



❑ Phương thức Report

- ❖ Phương thức report được sử dụng để log các exception hoặc gửi chúng tới các dịch vụ bên ngoài như BugSnag hay Sentry.
- ❖ Mặc định, report đơn giản chỉ đẩy exception về class cơ sở nơi mà exception được log lại
- ❖ Report các kiểu exception bằng nhiều cách khác nhau, bằng cách sử dụng toán tử kiểm tra instanceof của PHP:

❑ Phương thức Report

```
/**
 * Report or log an exception.
 *
 * This is a great spot to send exceptions to Sentry, Bugsnag, etc.
 *
 * @param \Exception $e
 * @return void
 */
public function report(Exception $e)
{
    if ($e instanceof CustomException) {
        //
    }

    return parent::report($e);
}
```

❑ Phương thức Render

- ❖ Phương thức render chịu trách nhiệm chuyển đổi một exception thành một mẫu HTTP response để trả lại cho trình duyệt

```
/**
 * Render an exception into an HTTP response.
 *
 * @param \Illuminate\Http\Request $request
 * @param \Exception $e
 * @return \Illuminate\Http\Response
 */
public function render($request, Exception $e)
{
    if ($e instanceof CustomException) {
        return response()->view('errors.custom', [], 500);
    }

    return parent::render($request, $e);
}
```

□ HTTP Exceptions

- ❖ Một số exception mô tả mã lỗi HTTP từ server. Ví dụ, đó có thể là một lỗi "page not found" (404), một lỗi "unauthorized error" (401) hoặc lỗi 500
- ❖ Để sinh ra response cho mã lỗi trên ứng dụng, sử dụng:

```
abort(404);
```

- ❖ Phương thức abort sẽ đẩy ra một exception sẽ được render bởi exception handler. Ngoài ra có thể tùy chọn cung cấp thêm nội dung response:

```
abort(403, Môn Laravel thật thú vị!);
```






DEMO



-Demo bắt exception và thông báo
Trang lỗi



Phần I: Tổng quan Email

-  Giới thiệu email
-  Cấu hình gửi mail
-  Cơ chế mail

Phần II: The Exception Handler

-  Quản lý Error
-  Quản lý Log
-  Exception Handler





Cảm ơn