Automating Email Sending in Laravel: A Step-by-Step Guide with Scheduler

Tutorial view(s): 4364 Author: Jing Rayo Published on: October 4, 2023 Updated at: May 30, 2025

A smart method to automate email sending jobs in your Laravel application is by setting up a mailing system using the scheduler. In this tutorial, I'll guide you through the steps to set up a mailing system using Laravel's schedule.

Prerequisites:

- Laravel installed on your local development environment. You can check the tutorial here on Creating your first laravel project locally
- A basic understanding of Laravel and its components.

Step 1: Create a Laravel Project

In case you haven't already created one, you can simply run this composer command on your terminal.

```
composer create-project --prefer-dist laravel/laravel laravel-scheduler-mailing
```

Step 2: Configure Mail Settings

Open your **.env** file located on the root folder of your project. In your **.env** file you need to configure the mail settings such as mailer, port, host, username, password and etc.

```
MAIL_MAILER=smtp
MAIL_HOST=your-smtp-server.com
MAIL_PORT=587
MAIL_USERNAME=your-smtp-username
MAIL_PASSWORD=your-smtp-password
MAIL_ENCRYPTION=tls
```

Note: Mail mailer can be mailgun, sendgrid, SMTP.

Step 3: Create Mailable Class

We need to create a mail class to define the mail that we want to send. You can generate this class or file by runing the command below and it will be created on your **app/Mail** directory.

```
php artisan make:mail MyCustomMail
```

```
<?php

namespace App\Mail;

use Illuminate\Bus\Queueable;
use Illuminate\Mail\Mailable;
use Illuminate\Queue\SerializesModels;
use Illuminate\Contracts\Queue\ShouldQueue;

class MyCustomMail extends Mailable
{
```

```
    use Queueable, SerializesModels;
    public $data;
    /**
     * Create a new message instance.
     *
     * @return void
     */
    public function __construct($data)
    {
        $this->data = $data;
    }

    /**
     * Build the message.
     *
     * @return $this
     */
    public function build()
    {
        return $this->replyTo('support@example.com.ph')->subject('Prime')->markdown('email.template');
    }
}
```

Step 4: Create a Scheduled Command

Create a scheduled command to send emails at specific intervals. Run the following command to generate a new scheduled command and it will be created on your **app/Console/Commands** directory.

```
php artisan make:command SendEmails
```

Step 5: Configure the Scheduled Command

The logic for sending emails can be configured in the handle method by opening the SendEmails.php file.

```
use Illuminate\Console\Command;
use Illuminate\Support\Facades\Mail;
use App\Mail\MyCustomMail;

class SendEmails extends Command
{
    protected $signature = 'emails:send';
    protected $description = 'Send custom emails';

    public function __construct()
    {
        parent::__construct();
    }

    public function handle()
    {
        $users = User::all();

        foreach ($users as $user) {
            Mail::to($user->email)->send(new MyCustomMail());
        }

        $this->info('Emails sent successfully!');
    }
}
```

Step 6: Schedule the Command

On your **app/Console/Kernel.php** , you can add when the email or reminder will send. You can set it up on schedule method or function. You can schedule it to send every seconds, minutes, hours and based on your prefer time. You can check the laravel documentation for more reference https://laravel.com/docs/10.x/scheduling.

```
protected function schedule(Schedule $schedule)
{
    $schedule->command('emails:send')->dailyAt('08:00')->withoutOverlapping(10); //This email will send e
}
```

In the example above, we use *withoutOverLapping(10)* it is a method to avoid to run 2 instance at a time.

Step 7: Run the Scheduler

After you set up the commands, schedule and other related function for your automatic email. You can now run the scheduler locally using the artisan command below:

```
php artisan schedule:run

//or

php artisan schedule:work
```

or set up it on your server cronjob

```
* * * * * cd <project directory> && sudo php artisan schedule:run >> /dev/null 2>&1
```

About the Author

Jing Rayo

Web Developer

Socials Email

jing@molavenet.com

1 Comment(s) and 0 Reply(s)

Login

Please log in to comment.

Bar Molavenet

November 13, 2024, 1:44 PM

This tutorial is helpful to me. Thanks!

Seat Leasing

OFFSHORE STAFF LEASING

- Dedicated Staff
- Monthly Flat Rate
- Full-time/Part-time
- Flexible Staffing
- Offshore Staff Leasing

PROJECT OUTSOURCING

Back Office
Web Marketing
Contact Center
Design
Development

PRIME SERVICE PACKS

Standard Packages
Custom Selection
Prices Start at $100
Pay as You Go
Professional Staff