**url;** [**https://appdividend.com/2022/02/28/laravel-queues/**](https://appdividend.com/2022/02/28/laravel-queues/) **(best doc follow this)**

**url youtube:** [**https://www.youtube.com/watch?v=fFy-s7\_SbYM**](https://www.youtube.com/watch?v=fFy-s7_SbYM) **(bitfums videos watch for basic clear)**

**Laravel Queues**

Laravel queues provide a unified API across various queue backends, such as Beanstalk, Amazon SQS, Redis, or even a relational database. As a result, queues allow you to defer processing a time-consuming task, such as sending an email. Delaying these time-consuming tasks drastically speeds up web requests to your application.

The queue configuration file is stored in.config/queue.php In this file, you will find connection configurations for each queue driver included with the framework, consisting of a database, [Beanstalkd](https://kr.github.io/beanstalkd/), [Amazon SQS](https://aws.amazon.com/sqs/), [Redis](https://redis.io/), and synchronous driver that will execute jobs immediately (for local use). In addition, a **null** queue driver is also included, which discards queued jobs.

Advertisement

**Step 1: Configure the Laravel**

Install it by the following command.

[](https://go.ezodn.com/ads/charity/proxy?p_id=bdff5c79-cf27-40fe-6dc4-804caee65e38&d_id=69680&imp_id=4813403515153700&c_id=1093&l_id=10016&url=https://www.directrelief.org/emergency/ukraine-crisis/&ffid=1&co=BD)

composer create-project laravel/laravel --prefer-dist emailqueue

Now, in the **.env**file, configure the database and email server.

For email sending, I have used <https://mailtrap.io>. So if you want that use, go to the website and signup, and you will see your username and password, grab it and put it in the **.env**file, and you are good to go.

Also, we need to change the **queue driver to the database.** By default, laravel uses **sync.**

APP\_NAME=Laravel

APP\_ENV=local

APP\_KEY=base64:N/+b/PAQXgtG8OiYu3TKyudhicgYsGYfb551yrqkjmQ=

APP\_DEBUG=true

APP\_LOG\_LEVEL=debug

APP\_URL=http://localhost

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=queue

DB\_USERNAME=root

DB\_PASSWORD=

BROADCAST\_DRIVER=log

CACHE\_DRIVER=file

SESSION\_DRIVER=file

SESSION\_LIFETIME=120

QUEUE\_DRIVER=database

REDIS\_HOST=127.0.0.1

REDIS\_PASSWORD=null

REDIS\_PORT=6379

MAIL\_DRIVER=smtp

MAIL\_HOST=smtp.mailtrap.io

MAIL\_PORT=2525

MAIL\_USERNAME=

MAIL\_PASSWORD=

MAIL\_ENCRYPTION=tls

PUSHER\_APP\_ID=

PUSHER\_APP\_KEY=

PUSHER\_APP\_SECRET=

**Step 2: Create a route for sending mail.**

We will create one route to send an email and create one controller called **EmailController.**

php artisan make:controller EmailController

In **routes >> web.php** file, add the following code.

Route::get('email', 'EmailController@sendEmail');

Okay, now we need to create one mailable class, so type the following command in the terminal.

php artisan make:mail SendMailable

So, it will create this file inside **App\Mail\SendMailable.php.**

Now, write the **sendMail** function inside the **MailController.php**file.

[](https://go.ezodn.com/ads/charity/proxy?p_id=bdff5c79-cf27-40fe-6dc4-804caee65e38&d_id=69680&imp_id=9053860451136132&c_id=1093&l_id=10016&url=https://www.directrelief.org/emergency/ukraine-crisis/&ffid=1&co=BD)

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Mail;

use App\Mail\SendMailable;

class EmailController extends Controller

{

public function sendEmail()

{

Mail::to('mail@appdividend.com')->send(new SendMailable());

echo 'email sent';

}

}

Also, our **SendMailable.php** file looks like this.

<?php

namespace App\Mail;

use Illuminate\Bus\Queueable;

use Illuminate\Mail\Mailable;

use Illuminate\Queue\SerializesModels;

use Illuminate\Contracts\Queue\ShouldQueue;

class SendMailable extends Mailable

{

use Queueable, SerializesModels;

/\*\*

\* Create a new message instance.

\*

\* @return void

\*/

public function \_\_construct()

{

//

}

/\*\*

\* Build the message.

\*

\* @return $this

\*/

public function build()

{

return $this->view('welcome');

}

}

Now, start the server with the following command.

php artisan serve

Switch to the browser and hit this URL: **http://localhost:8000/email**

You will see a delay, and then we can get the mail. So here is the solution, and that is a **queue.**

**Step 3: Configure Queue.**

We can use queues with the **Database, Redis,** and other drivers as mentioned in [Laravel Queues](https://laravel.com/docs/5.5/queues).

We will use a Database, so we need to create a migration for the Jobs table.

php artisan queue:table

It will create a migration, and then we need to migrate the database.

php artisan migrate

It will create the **jobs**table in the database.

Finally, don't forget to instruct your application to use the database driver by updating the QUEUE\_CONNECTION variable in your application's .env file:

QUEUE\_CONNECTION=database ( VVVVVVI)

**Step 4: Create a job**

We will need to create the job to send the actual email to the user.

php artisan make:job SendEmailJob

Now, the send email function will reside in the job file. So that job file looks like this.

<?php

namespace App\Jobs;

use Illuminate\Bus\Queueable;

use Illuminate\Queue\SerializesModels;

use Illuminate\Queue\InteractsWithQueue;

use Illuminate\Contracts\Queue\ShouldQueue;

use Illuminate\Foundation\Bus\Dispatchable;

use Illuminate\Support\Facades\Mail;

use App\Mail\SendMailable;

class SendEmailJob implements ShouldQueue

{

use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;

/\*\*

\* Create a new job instance.

\*

\* @return void

\*/

public function \_\_construct()

{

//

}

/\*\*

\* Execute the job.

\*

\* @return void

\*/

public function handle()

{

Mail::to('mail@appdividend.com')->send(new SendMailable());

}

}

In the **EmailController.php** file, we will need to trigger this job.

So we need to **dispatch** the newly created **job** from the **EmailController.** So that code will look like this.

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Jobs\SendEmailJob;

class EmailController extends Controller

{

public function sendEmail()

{

dispatch(new SendEmailJob());

echo 'email sent';

}

}

Now, again hit that email URL.

Still, the delay is there; we have just moved the logic to the job file. We need to send the mail in the background. So we need to delay our **queue**.

public function sendEmail()

{

$emailJob = (new SendEmailJob())->delay(Carbon::now()->addSeconds(3));

dispatch($emailJob);

echo 'email sent';

}

Now, hit the URL again.

This time there is only 3 seconds delay, or we can say no delay, but wait, if you switch to **mailtrap,** there is no mail either. So how can we send real mail also?

Go to the database and see the **jobs** table. There is one entry for the job. So that means the process of that job is not started. So Laravel, there is a concept called **Queue Worker. So we** need to run that **Queue Worker**.

Switch to the terminal and type the following command to start the queue worker.

php artisan queue:work

In the terminal, it will say like this.

[2017-12-21 13:58:14] Processing: App\Jobs\SendEmailJob

[2017-12-21 13:58:19] Processed: App\Jobs\SendEmailJob