Email in Laravel — App Se Khat Bhejna

Letters from your app: mailable classes with envelope and content, Mail::to()->send(), Mailpit the fake inbox, and the log-mailer trick for safe testing.

Introduction

Apps write letters: *welcome, your report card is ready, reset your password* (LV.7's reset link travels by exactly this road). Laravel's mail system wraps the whole letter in a **Mailable class** — one PHP class per kind of letter, holding its subject (the envelope), its body (a Blade view, so emails wear your design) and its attachments.

This lesson builds the welcome letter for Students Manager, sends it at registration, and — crucially — reads it in **Mailpit**, a fake local inbox that catches every letter your app sends while you develop. No real inbox is spammed, no real user is emailed by accident, and every letter can be inspected like a specimen. Slow letters get their own lesson next (LV.16); here, the letter itself is the star.

(Roman Urdu: App bhi khat likhti hai — aik class aik khat. Aur jaanch ke liye nakli dak-khana: Mailpit, jahan har khat pakra jata hai, koi asli inbox pareshan nahi hota.)

Key points

- **One Mailable per kind of letter:** `php artisan make:mail WelcomeMail` creates the class. Its anatomy is three methods: `envelope()` (from, subject), `content()` (which Blade view is the body, with what data), `attachments()` (files riding along). Read the class and you know the letter completely.
- **Sending is one calm line:** `Mail::to($user->email)->send(new WelcomeMail($user))` — address, letter, send. The machinery (SMTP settings from `.env`) stays out of your controller.
- **Email bodies are Blade views:** `resources/views/emails/welcome.blade.php` — variables, `@if`, and your layout habits all work, so letters match the app's face instead of arriving as grey typewriter text.
- **Mailpit is the development inbox:** a small local tool that pretends to be a mail server. Your `.env` points at it (`MAIL_HOST=127.0.0.1`, `MAIL_PORT=1025`) and its web face at **http://localhost:8025** shows every caught letter, rendered, with headers — testing heaven, zero risk.

  (Roman Urdu: Asli dakiya baad mein; pehle har khat nakli dak-khanay mein khol kar dekho — ghalti wahan sasti parti hai.)
- **The log mailer is the emergency trick:** `MAIL_MAILER=log` writes the whole raw letter into `laravel.log` instead of sending it — on servers with no mail at all, the letter still exists somewhere readable.
- **Never hard-code addresses:** a `Mail::to('mytest@gmail.com')` left in the code will one day mail YOU a stranger's reset link — or worse, mail a real user your test junk. Addresses come from data: the registering user, the settings, never a typed constant.
- **Sending takes time:** a real SMTP round-trip can eat 2–3 seconds of a user's page load. That cost is real — and it is exactly what queues (LV.16) exist to move off the page.

Code examples

Example 1 — The letter itself: WelcomeMail's anatomy

```
php artisan make:mail WelcomeMail
```

```
// app/Mail/WelcomeMail.php
class WelcomeMail extends Mailable
{
    use Queueable, SerializesModels;

    public function __construct(public User $user) {}

    public function envelope(): Envelope
    {
        return new Envelope(
            subject: 'Welcome to Students Manager, ' . $this->user->name . '!',
        );
    }

    public function content(): Content
    {
        return new Content(
            view: 'emails.welcome',
        );
    }

    public function attachments(): array
    {
        return [];
    }
}
```

```
{{-- resources/views/emails/welcome.blade.php --}}
<h1>Assalam o Alaikum, {{ $user->name }}!</h1>
<p>Your teacher account is ready. Log in to manage your class,
add students and follow their marks.</p>
<p>— Team Students Manager</p>
```

What it does: Artisan's skeleton, filled honestly. The user arrives through the constructor (typed, public — so the view can use `$user` directly). The envelope carries the subject; the content names the Blade view; attachments waits empty, its turn comes in Example 3.

What you will see: nothing moves yet — a letter written and sealed, waiting for an address. Example 2 posts it.

Example 2 — Send at registration, catch in Mailpit

```
// right after the new user is created (registration / teacher store)
Mail::to($user->email)->send(new WelcomeMail($user));
```

```
# .env — the development post office
MAIL_MAILER=smtp
MAIL_HOST=127.0.0.1
MAIL_PORT=1025
MAIL_FROM_ADDRESS="hello@studentsmanager.test"
MAIL_FROM_NAME="Students Manager"
```

What it does: One line at the moment of registration. The `.env` aims the app's post at Mailpit's SMTP ear on port 1025, with a respectable from-address. Register a new teacher — say Fatima Noor, `fatima@school.test` — and refresh Mailpit's web inbox.

What you will see: the letter, caught and rendered at http://localhost:8025 — subject `Welcome to Students Manager, Fatima Noor!`, the greeting wearing her name, from `hello@studentsmanager.test`. Zero real inboxes touched, and the letter can be re-read, re-sent and inspected header by header.

Example 3 — The log trick, and an attachment rides along

```
# .env — no mail server available? Letters become log entries
MAIL_MAILER=log
```

```
// WelcomeMail grows an attachment: the class report card PDF
public function attachments(): array
{
    return [
        Attachment::fromPath(storage_path('app/reports/class-report.pdf'))
            ->as('Class-Report.pdf')
            ->withMime('application/pdf'),
    ];
}
```

What it does: Two survival skills. First: where no mail server exists, `MAIL_MAILER=log` keeps the letters flowing into `storage/logs/laravel.log` — the full raw letter, headers and all, readable in the diary LV.8 taught you to trust. Second: attachments are declared in the Mailable itself — a real file from storage, renamed politely for the reader (`Class-Report.pdf`), with its type stated.

What you will see: with the log mailer on, a registration adds a long, complete letter to `laravel.log` — subject, HTML body, everything — while the attachments version, sent through Mailpit, shows a paperclip in

the inbox and the PDF opens from the letter, named exactly as promised.

Quick recap

- A Mailable class is one kind of letter: `envelope()` subject, `content()` Blade view, `attachments()` files.
- Send with `Mail::to($address)->send(new WelcomeMail($user))` — addresses from data, never hard-coded.
- Develop against Mailpit (localhost:8025); fall back to `MAIL_MAILER=log` where no server exists.
- Slow sending is real — that cost moves to queues in the very next lesson.

Common mistakes

1. **Hard-coded test addresses in the code** — the leftover `Mail::to('myself@test.com')` that one day receives a real user's password reset. Fix: the recipient always comes from the data being processed (the user, the order, the booking).
2. **Testing against real users' inboxes** — a broken template emailed to 400 parents is a permanent story. Fix: Mailpit in development, a staging inbox after that; production addresses are earned, never experimented on.
3. **Plain-text-only letters when a view was promised** — forgetting the `emails.welcome` view exists and mailing raw concatenated strings. Fix: one Mailable + one Blade view per letter kind; strings are for log lines, not letters.
4. **Forgetting the queue cost** — attaching mail-sending directly to five places on a slow SMTP server until every page crawls. Fix: send simply here; at LV.16 the same Mailable gets queued, unchanged — design it sendable from day one.

Practice

1. Build a second letter, `MarksUpdatedMail`, sent when a student's marks change (LV.6's update), with the old and new marks in its Blade view. Catch two sends in Mailpit and verify both names and numbers.
2. Switch `MAIL_MAILER=log`, register a teacher, and find the complete letter inside `laravel.log`. Write down: its subject line, its from address, and the first line of its body — all from the log alone.

Practice quiz

1. What are the three methods of a Mailable's anatomy?

- to(), from(), send()
- envelope() for from and subject, content() for which Blade view is the body with what data, attachments() for files riding along
- subject(), body(), footer()
- create(), send(), delete()

Answer: envelope() for from and subject, content() for which Blade view is the body with what data, attachments() for files riding along

2. What is Mailpit, and where do you watch its caught letters?

- A real mail server for production
- A small local tool that pretends to be a mail server; your .env points at it and its web face at http://localhost:8025 shows every caught letter, rendered, with headers
- A Laravel package for spam filtering
- A Blade view for emails

Answer: A small local tool that pretends to be a mail server; your .env points at it and its web face at http://localhost:8025 shows every caught letter, rendered, with headers

3. What does MAIL_MAILER=log do?

- It sends mail faster
- It writes the whole raw letter into laravel.log instead of sending it, so on servers with no mail at all the letter still exists somewhere readable
- It deletes all mail
- It logs the user out

Answer: It writes the whole raw letter into laravel.log instead of sending it, so on servers with no mail at all the letter still exists somewhere readable

---

Watch the Code Scope channel: https://www.youtube.com/@CodeScopePK — this lesson's own video joins the channel when it is recorded.

Read online: https://scopetreasure.com/notes/laravel-email · More free tools and notes at https://scopetreasure.com