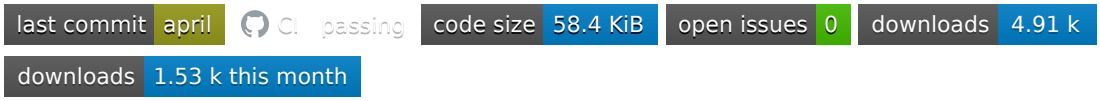
Introduction

Elegant orchestration of email communications for PHP

Anonymous · 09/09/2026

Elegant orchestration of email communications for PHP



A flexible PHP email library, built on Derafu Backbone architecture, that leverages other libraries and orchestrates the entire sending and receiving process.

Overview

Derafu Mail provides a robust, extensible framework for sending and receiving emails in PHP applications. Built on the Derafu Backbone architecture, it offers a clean, maintainable structure with clear separation of concerns.

Features

- **Clean Architecture**: Follows the Derafu Backbone hierarchical structure.
- **Sending Emails**: SMTP support with easy extensibility for other transport methods.
- **Receiving Emails**: IMAP support with customizable search criteria and filtering.
- **Flexible Configuration**: Comprehensive options for both sending and receiving.
- **Robust Error Handling**: Proper exception handling throughout the library.
- **Attachment Management**: Support for both sending and receiving attachments.
- **Strategy Pattern**: Easily swap out sending/receiving implementations.

Installation

Install the package via Composer:

```
composer require derafu/mail
```

Quick Start

Sending Emails

```
use Derafu\Backbone\Contract\PackageRegistryInterface;
use Derafu\Mail\Model\Message;
use Derafu\Mail\Model\Envelope;
use Derafu\Mail\Model\Postman;
use Symfony\Component\Mime\Address;

// Find the package registry using dependency injection and get a $senderWorker
// Also you can inject directly the SenderWorkerInterface wherever you want.
$packageRegistry = $container->get(PackageRegistryInterface::class);
$mailPackage = $packageRegistry->getPackage('mail');
$exchangeComponent = $mailPackage->getExchangeComponent();
$senderWorker = $exchangeComponent->getSenderWorker();

// Create a message.
$message = new Message();
$message->subject('Hello World')
    ->text('This is a plain text message')
    ->html('<h1>Hello World</h1><p>This is an HTML message</p>')
    ->from(new Address('sender@example.com', 'Sender Name'))
    ->to(new Address('recipient@example.com', 'Recipient Name'));

// Create an envelope and add the message.
$envelope = new Envelope(
    new Address('sender@example.com', 'Sender Name'),
    [new Address('recipient@example.com', 'Recipient Name')]
);
$envelope->addMessage($message);

// Create a postman with SMTP configuration.
$postman = new Postman([
    'strategy' => 'smtp',
    'transport' => [
        'host' => 'smtp.example.com',
        'port' => 587,
        'encryption' => 'tls',
        'username' => 'your_username',
        'password' => 'your_password',
    ],
]);
$postman->addEnvelope($envelope);

// Send the email.
$envelopes = $senderWorker->send($postman);
```

Receiving Emails

```
use Derafu\Backbone\Contract\PackageRegistryInterface;
use Derafu\Mail\Model\Postman;

// Find the package registry using dependency injection and get a $receiverWorker
// Also you can inject directly the ReceiverWorkerInterface wherever you want.
$packageRegistry = $container->get(PackageRegistryInterface::class);
$mailPackage = $packageRegistry->getPackage('mail');
$exchangeComponent = $mailPackage->getExchangeComponent();
$receiverWorker = $exchangeComponent->getReceiverWorker();

// Create a postman with IMAP configuration.
$postman = new Postman([
    'strategy' => 'imap',
    'transport' => [
        'host' => 'imap.example.com',
        'port' => 993,
        'encryption' => 'ssl',
        'username' => 'your_username',
        'password' => 'your_password',
        'mailbox' => 'INBOX',
        'search' => [
            'criteria' => 'UNSEEN',
            'markAsSeen' => true,
            'attachmentFilters' => [
                'extension' => ['pdf', 'doc', 'docx'],
            ],
        ],
    ],
]);

// Receive emails.
$envelopes = $receiverWorker->receive($postman);

// Process received emails.
foreach ($envelopes as $envelope) {
    foreach ($envelope->getMessages() as $message) {
        echo "Subject: " . $message->getSubject() . PHP_EOL;
        echo "From: " . $message->getFrom()[0]->getAddress() . PHP_EOL;
        echo "Body: " . $message->getTextBody() . PHP_EOL;

        // Process attachments.
        foreach ($message->getAttachments() as $attachment) {
            file_put_contents('/path/to/save/' . $attachment->getFilename(),
$attachment->getBody());
        }
    }
}
```

Extending with New Strategies

The library is designed to be easily extended with new sending or receiving strategies:

1. Create a new strategy class implementing `SenderStrategyInterface` or `ReceiverStrategyInterface`.
2. Tag it with the appropriate attribute.

Example:

```
use Derafu\Backbone\Attribute\Strategy;
use
Derafu\Mail\Component\Exchange\Worker\Sender\Strategy\Abstract\AbstractMailerStrategy;
use
Derafu\Mail\Component\Exchange\Worker\Sender\Strategy\Contract\SenderStrategyInterface
;

#[Strategy(name: 'mailgun', worker: 'sender', component: 'exchange', package: 'mail')]
class MailgunStrategy extends AbstractMailerStrategy implements
SenderStrategyInterface
{
    // Implementation of SenderStrategyInterface that leverages
AbstractMailerStrategy.
}
```

Architecture

Derafu Mail follows the Derafu Backbone architecture:

- **Package**: MailPackage - The main entry point.
- **Component**: ExchangeComponent - Handles email exchange.
- **Workers**: SenderWorker and ReceiverWorker - Handle sending and receiving.
- **Handlers**: SendHandler and ReceiveHandler - Orchestrate the process.
- **Strategies**: Implement different methods of sending or receiving.

Last updated on 09/09/2026