

Laravel - Sending Email

Laravel uses free feature-rich library **SwiftMailer** to send emails. Using the library function, we can easily send emails without too many hassles. The e-mail templates are loaded in the same way as views, which means you can use the Blade syntax and inject data into your templates.

The following table shows the syntax and attributes of **send** function –

Syntax	<code>void send(string array \$view, array \$data, Closure string \$callback)</code>
Parameters	<p><code>\$view(string array)</code> – name of the view that contains email message</p> <p><code>\$data(array)</code> – array of data to pass to view</p> <p><code>\$callback</code> – a Closure callback which receives a message instance, allowing you to customize the recipients, subject, and other aspects of the mail message</p>
Returns	nothing
Description	Sends email.

In the third argument, the `$callback` closure received message instance and with that instance we can also call the following functions and alter the message as shown below.

```
$message → subject('Welcome to the Tutorials Point');  
$message → from('email@example.com', 'Mr. Example');  
$message → to('email@example.com', 'Mr. Example');
```

Some of the less common methods include –

```
$message → sender('email@example.com', 'Mr. Example');  
$message → returnPath('email@example.com');  
$message → cc('email@example.com', 'Mr. Example');
```

```
$message → bcc('email@example.com', 'Mr. Example');  
$message → replyTo('email@example.com', 'Mr. Example');  
$message → priority(2);
```

To attach or embed files, you can use the following methods –

```
$message → attach('path/to/attachment.txt');  
$message → embed('path/to/attachment.jpg');
```

Mail can be sent as HTML or text. You can indicate the type of mail that you want to send in the first argument by passing an array as shown below. The default type is HTML. If you want to send plain text mail then use the following syntax.

Syntax

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

In this syntax, the first argument takes an array. Use **text** as the key name of the view as value of the key.

Example

Step 1 – We will now send an email from Gmail account and for that you need to configure your Gmail account in Laravel environment file - **.env** file. Enable 2-step verification in your Gmail account and create an application specific password followed by changing the .env parameters as shown below.

.env

```
MAIL_DRIVER = smtp  
MAIL_HOST = smtp.gmail.com  
MAIL_PORT = 587  
MAIL_USERNAME = your-gmail-username  
MAIL_PASSWORD = your-application-specific-password  
MAIL_ENCRYPTION = tls
```

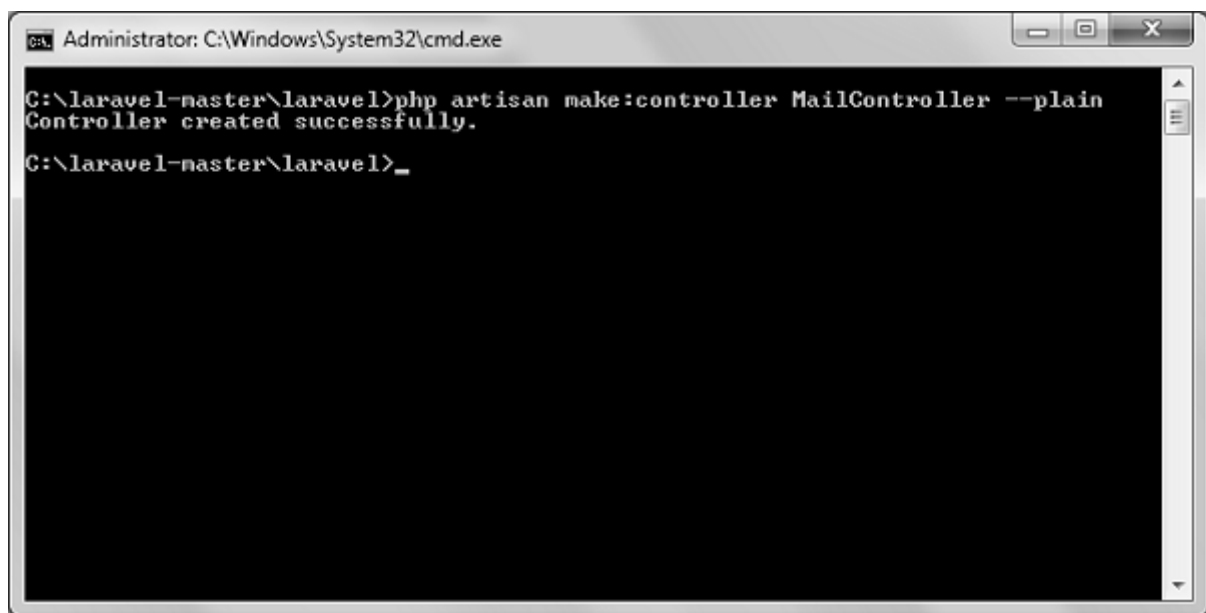
Step 2 – After changing the **.env** file execute the below two commands to clear the cache and restart the Laravel server.

```
php artisan config:cache
```

Step 3 – Create a controller called **MailController** by executing the following command.

```
php artisan make:controller MailController --plain
```

Step 4 – After successful execution, you will receive the following output –



```
Administrator: C:\Windows\System32\cmd.exe
C:\laravel-master\laravel>php artisan make:controller MailController --plain
Controller created successfully.
C:\laravel-master\laravel>_
```

Step 5 – Copy the following code in

app/Http/Controllers/MailController.php file.

app/Http/Controllers/MailController.php

```
<?php

namespace App\Http\Controllers;
use Illuminate\Http\Request;
use Mail;

use App\Http\Requests;
use App\Http\Controllers\Controller;

class MailController extends Controller {
    public function basic_email() {
        $data = array('name'=>"Virat Gandhi");
```

```
Mail::send(['text'=>'mail'], $data, function($message) {
    $message->to('abc@gmail.com', 'Tutorials Point')->subject
        ('Laravel Basic Testing Mail');
    $message->from('xyz@gmail.com','Virat Gandhi');
});
echo "Basic Email Sent. Check your inbox.";
}
public function html_email() {
    $data = array('name'=>"Virat Gandhi");
    Mail::send('mail', $data, function($message) {
        $message->to('abc@gmail.com', 'Tutorials Point')->subject
            ('Laravel HTML Testing Mail');
        $message->from('xyz@gmail.com','Virat Gandhi');
    });
    echo "HTML Email Sent. Check your inbox.";
}
public function attachment_email() {
    $data = array('name'=>"Virat Gandhi");
    Mail::send('mail', $data, function($message) {
        $message->to('abc@gmail.com', 'Tutorials Point')->subject
            ('Laravel Testing Mail with Attachment');
        $message->attach('C:\laravel-master\laravel\public\uploads\
        $message->attach('C:\laravel-master\laravel\public\uploads\
        $message->from('xyz@gmail.com','Virat Gandhi');
    });
    echo "Email Sent with attachment. Check your inbox.";
}
}
```

Step 6 – Copy the following code in **resources/views/mail.blade.php** file.

resources/views/mail.blade.php

```
<h1>Hi, {{ $name }}</h1>
<p>Sending Mail from Laravel.</p>
```

Step 7 – Add the following lines in **app/Http/routes.php**.

app/Http/routes.php

```
Route::get('sendbasicemail','MailController@basic_email');  
Route::get('sendhtmlmail','MailController@html_email');  
Route::get('sendattachmentemail','MailController@attachment_email');
```

Step 8 – Visit the following URL to test basic email.

<http://localhost:8000/sendbasicemail>

Step 9 – The output screen will look something like this. Check your inbox to see the basic email output.

Basic Email Sent. Check your inbox.

Step 10 – Visit the following URL to test the HTML email.

<http://localhost:8000/sendhtmlmail>

Step 11 – The output screen will look something like this. Check your inbox to see the html email output.

HTML Email Sent. Check your inbox.

Step 12 – Visit the following URL to test the HTML email with attachment.

<http://localhost:8000/sendattachmentemail>

Step 13 – You can see the following output

Email Sent with attachment. Check your inbox.

Note – In the **MailController.php** file the email address in the from method should be the email address from which you can send email address. Generally, it should be the email address configured on your server.