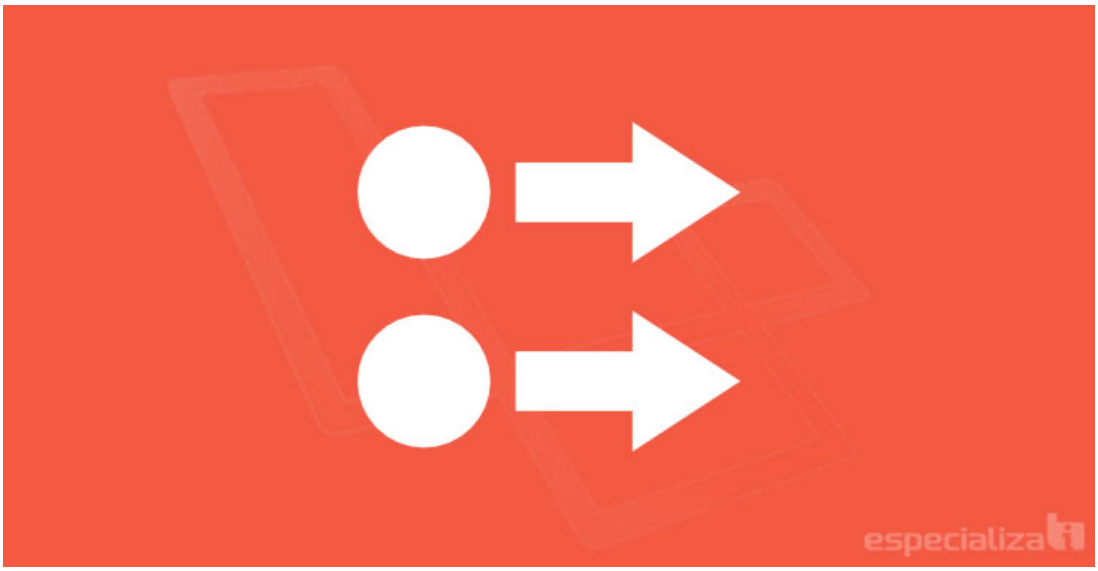
Por: Carlos Ferreira 09/07/2018 Comentar

Eventos no Laravel – Parte 2



Compartilhe: (https://www.facebook.com/sharer/sharer.php?u=https://blog.especializati.com.br/eventos-no-laravel-parte-2/) (https://twitter.com/intent/tweet?text=https://blog.especializati.com.br/eventos-no-laravel-parte-2/) (hhttps://plus.google.com/share?url=https://blog.especializati.com.br/eventos-no-laravel-parte-2/)

Esse tutorial é a continuação do tutorial sobre Eventos no Laravel, portanto se você ainda não leu o **tutorial anterior (https://blog.especializati.com.br/aprenda-como-trabalhar-com-eventos-no-laravel/)**, é muito importante que leia, para conseguir entender.

No exemplo do tutorial anterior disparamos um Evento que por sua vez está ligado a um Listener que por enquanto está apenas registrando log, mas agora nos vamos enviar de fato um e-mail para tornar o exemplo mais real.

Preparando o envio de e-mail

O Laravel dispõe de algumas formas de envio de e-mails, neste exemplo vamos fazer o envio através de uma classe, como o Markdown.

Crie uma classe para o envio do e-mail:

```
php artisan make:mail PostCommentedMail --markdown=emails.posts.new-comment.blade
```

Esse comando vai gerar uma classe para o envio de e-mails em `app/Mail/PostCommentedMail.php`

Podemos definir no construtor da nossa classe para o envio de e-mail que vai receber a mensagem, veja a implementação:

```
namespace App\Mail;

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

class PostCommentedMail extends Mailable
{
    use Queueable, SerializesModels;

    public $comment;

    /**
     * Create a new message instance.
     *
     * @return void
     */
    public function __construct(Comment $comment)
    {
        $this->comment = $comment;
    }

    /**
     * Build the message.
     *
     * @return $this
     */
    public function build()
    {
        return $this
                    ->subject('Novo Comentário')
                    ->markdown('mails.posts.new-comment');
    }
}
```

No método **build()** definimos o assunto do e-mail com o método **subject()** e no método **markdown()** passamos qual view terá o conteúdo da e-mail.

NOTA: Observe que o atributo **$comment** é public, isso para facilitar acessar esses valores na view de e-mail.

Agora o próximo passo é implementar a view, com o conteúdo do e-mail, crie um novo arquivo de view em `resources/views/mails/posts/new-comment.blade.php` e implemente a o conteúdo com Markdown:

```
@component('mail::message')

{{ $comment->body }}

@component('mail::button', ['url' => route('posts.show', $comment->post->id)])
Ver Comentário: {{ $comment->post->title }}
@endcomponent

Obrigado,<br>
{{ config('app.name') }}
@endcomponent
```

NOTA: Em diversos momentos usamos as rotas com os seus respectivos nomes (posts.show), precisa registrar essas rotas no arquivo `routes/web.php`

```
Route::resource('posts', 'Posts\PostController');

Route::post('comment', 'Posts\CommentController@store')->name('comments.store');
```

Nesse exemplo implementamos a rota resource para posts, e outra para o comentário.

Listener Enviar o E-mail

Agora que já preparamos a nossa classe de envio de e-mail, precisa fazer o Listener **SendMailCommentedPost** enviar o e-mail, no método **handle()** troque o ~~Log~~ por isso:

```
/**
 * Handle the event.
 *
 * @param  CommentedPost  $event
 * @return void
 */
public function handle(CommentedPost $event)
{
    // Registring log commented post
    // Log::info($event->comment());
    $comment = $event->comment();

    Mail::to($comment->post->user->email)
                ->send(new PostCommentedMail($comment));
}
```

Nesse exemplo usamos o método **to()** para definir que vai receber o e-mail, no caso o autor do post, e no método **send()** passamos um objeto com a nossa classe de e-mail, que criamos anteriormente.

Definindo Relacionamentos nos Models

Um item muito importante é definir os relacionamentos de tabelas nos Models, atualmente nesse exemplo temos 3 Models: Comment, Post, User.

No model `app/Models/User.php` vamos definir o relacionamento entre usuário (author) e posts:

```
public function posts()
{
    return $this->hasMany(Post::class);
}
```

No model `app/Models/Post.php` vamos definir o relacionamento entre usuário (author) e post, e também entre post e cometários:

```
public function user()
{
    return $this->belongsTo(User::class);
}

public function comments()
{
    return $this->hasMany(Comment::class);
}
```

E por último, no model `app/Models/Comment.php` vamos definir o relacionamento entre comentários e post, e também entre comentário e usuário:

```
public function post()
{
    return $this->belongsTo(Post::class);
}

public function user()
{
    return $this->belongsTo(User::class);
}
```

NOTA: Estes relacionamentos são de extrema importância, para que consigamos retornar as informações relacionadas de forma mais fácil, como por exemplo: **$comment->user->email**

Agora ao fazer o comentário, automaticamente o autor do post vai receber um e-mail informando que o seu post foi comentário. Faça o teste!

Listeners e Queues

Embora já esteja funcionando com sucesso o nosso evento com o Listener, e o envio de e-mail, ainda sim é interessante trabalhar com filas. Porque o envio de e-mail é um processo pesado, e isso pode travar a aplicação do usuário até que o envio seja finalizado.

Uma das formas de não travar a aplicação enquanto o Listener é processado, é usar filas (queues). Podemos aplicar o conceito de filas diretamente na classe de e-mail, mas também pode ser aplicado no Listener. Nesse exemplo vamos aplicar no Listener para fechar o exemplo.

Não vou entrar em detalhes sobre Queues (https://laravel.com/docs/5.6/queues) neste tutorial, mas futuramente escrevo algo sobre.

Para executar o Listener **SendMailCommentedPost** com queue (filas) sem travar o momento que a pessoa comentar, basta fazer a classe **SendMailCommentedPost** implementar a interface **ShouldQueue**, apenas isso!

```php
namespace App\Listeners;

use App\Mail\PostCommentedMail;
use Mail;
use Log;
use App\Events\CommentedPost;
use Illuminate\Queue\InteractsWithQueue;
use Illuminate\Contracts\Queue\ShouldQueue;

class SendMailCommentedPost implements ShouldQueue
{
    /**
     * Create the event listener.
     *
     * @return void
     */
    public function __construct()
    {
        //
    }

    /**
     * Handle the event.
     *
     * @param  CommentedPost  $event
     * @return void
     */
    public function handle(CommentedPost $event)
    {
        // Registring log commented post
        // Log::info($event->comment());
        $comment = $event->comment();

        Mail::to($comment->post->user->email)
                    ->send(new PostCommentedMail($comment));
    }
}
```

Nesses dois tutoriais usamos Eventos, Listeners, E-mails e Filas.

Espero que tenha gostado, e que tenha sido útil para você. Se você quiser baixar o exemplo prático desse tutorial, segue o link do GitHub (aqui (https://github.com/carlosfgti/laravel-events-listeners)). **PS.** Não se esqueça de deixar o star no repositório desse exemplo.

Qualquer dúvida só deixa aquele comentário! ☺

Abraços []'s

Compartilhe: (https://www.facebook.com/sharer/sharer.php?u=https://blog.especializati.com.br/eventos-no-laravel-parte-2/) (https://twitter.com/intent/tweet?text=https://blog.especializati.com.br/eventos-no-laravel-parte-2/) (hhttps://plus.google.com/share?url=https://blog.especializati.com.br/eventos-no-laravel-parte-2/)

Sobre o Autor:

Carlos Ferreira

Carlos Ferreira é Analista de Sistemas Experiente, Empreendedor, Fundador da empresa EspecializaTi.

Certificações: Comptia Linux +, LPI, Novell Certification.

TAMBÉM NO **ESPECIALIZATI**

Aprenda como Trabalhar com …

há 3 anos • 1 COMENTÁRIO

Um tema bastante interessante e ao mesmo tempo muito útil do …

Laravel Eloquent Global Scope

há 3 anos • 2 COMENTÁRIOS

No último tutorial detalhei sobre Local Scopes no Laravel, agora dando …

Qual versão do Laravel 5.x escolher?

há 3 anos • 5 COMENTÁRIOS

Se você desenvolve com o Laravel com frequência, certamente no momento …

Dois se TODO

há 3 anos

Antes de pretendo essa é m

CONTEÚDOS RELACIONADO:

(HTTPS://BLOG.ESPECIALIZATI.COM.BR/LARAVEL-REDIRECIONAR-APOS-AUTENTICACAO/)

(HTTPS://BLOG.ESPECIALIZATI.COM.BR/LARAVEL-BLADE/)

(HTTPS://BLOG.ESPECIALIZATI.COM.BR/ENVIO-DE-E-MAILS-NO-LARAVEL/)

especializa BLOG

(https://www.youtube.com/channel/UCxxxxxxxxxxxxxxxxxxxxxx?sub_confirmation=1)