






Por: Carlos Ferreira

📅 02/07/2018

💬 Comentar

Aprenda como Trabalhar com Eventos no Laravel



Compartilhe:  (<https://www.facebook.com/sharer/sharer.php?u=https://blog.especializati.com.br/aprenda-como-trabalhar-com-eventos-no-laravel/>)  (<https://twitter.com/intent/tweet?text=https://blog.especializati.com.br/aprenda-como-trabalhar-com-eventos-no-laravel/>)  (<https://plus.google.com/share?url=https://blog.especializati.com.br/aprenda-como-trabalhar-com-eventos-no-laravel/>)

Um tema bastante interessante e ao mesmo tempo muito útil do Laravel são os eventos. Como o próprio nome sugere evento é uma ação que pode ser disparada automaticamente, e a partir deste evento você pode executar diversas outras tarefas.

Um exemplo prático, imagine que temos uma página de um post (tutorial) e temos uma interação com o usuário onde ele pode comentar esse post, ao comentar o post nos podemos disparar um evento no Laravel informando ao autor que o post X foi comentado, e esse evento pode estar ligado a alguns Listeners (Ouvintes) que tomam ações automáticas, como por exemplo: enviar um e-mail para o dono do post, enviar um SMS para Admin e/ou registrar o log no sistema.

O exemplo do paragrafo acima poderia ser realizado manualmente, mas com Events do Laravel fica mais fácil, porque fica organizado, e também possibilita o reaproveitamento destas funcionalidades em outros locais;

Um evento pode conter diversos Listeners, e um mesmo Listener pode estar ligado a diversos Events. Os eventos no Laravel são registrados em `app/Events/` e os Listeners (Ouvintes) ficam em `app/Listeners/`

Criar Events no Laravel

Podemos e devemos abusar do artisan

(<https://laravel.com/docs/master/artisan>) para criar os Events e Listeners no Laravel.

Para criar um novo Event rode o comando do artisan `php artisan make:event NameEvent`, neste caso vamos criar um Event para quando um post é comentado:

```
1 | php artisan make:event CommentedPost
```

O arquivo será criado em `app/Events/CommentedPost.php`

Para criar os Listeners (Ouvintes) basta rodar o comando:

```
1 | php artisan make:listener SendMailCommentedPost --event=CommentedPost
```

O prefixo `--event` precisa receber o **CommentedPost**, porque já cria o Listener vinculado ao Event CommentedPost. Este arquivo ficará salvo em `app/Listeners/SendMailCommentedPost.php`

Outro formato de criar os **Events + Listeners** é rodar o comando:

```
1 | php artisan event:generate
```

Esse comando vai criar automaticamente todos os arquivos de Events e Listeners registrado em `app/Providers/EventServiceProvider.php` no atributo :

```
1 | protected $listen = [  
2 |     'App\Events\Event' => [  
3 |         'App\Listeners\EventListener',  
4 |     ],  
5 | ];
```

Neste exemplo vai gerar dois arquivos, o event que vai ficar salvo em `app/events/Event.php` e o Listener que vai ficar salvo em `app/listeners/EventListener.php`

Registrar Events no Laravel

Uma vez que os arquivos foram criado o próximo passo é vincular, ou seja, registrar o Event e definir qual/quais Listeners estarão vinculados com o mesmo. Em nosso exemplo vamos vincular o Listener **SendMailCommentedPost** ao event **CommentedPost**.

Para registrar basta adicionar no atributo \$listen no array, os Events e os Listeners do event. Exemplo:

```
1 | protected $listen = [  
2 |     \App\Events\CommentedPost::class => [  
3 |         \App\Listeners\SendMailCommentedPost::class,  
4 |     ],  
5 | ];
```

Neste exemplo vinculamos o Listener **SendMailCommentedPost** ao Event **CommentedPost**, neste caso sempre que disparar o Event **CommentedPost** automaticamente vai disparar o Listener **SendMailCommentedPost**.

NOTA: Observe que o atributo \$listen recebe um array, isso significa que podemos registrar N eventos. E o Event também receber um array, portanto para um mesmo evento podemos registrar N Listeners.

Configurar Events no Laravel

O próximo passo é configurar o Event, ou seja, definir o que vai receber no construtor e etc. Neste caso vamos receber no construtor do Event **CommentedPost** um objeto do comentário, veja:

```

1 | namespace App\Events;
2 |
3 | use App\Models\Comment;
4 | use Illuminate\Queue\SerializesModels;
5 |
6 | class CommentedPost
7 | {
8 |     use SerializesModels;
9 |
10 |    public $comment;
11 |
12 |    /**
13 |     * Create a new event instance.
14 |     *
15 |     * @param \App\Models\Comment $comment
16 |     * @return void
17 |     */
18 |    public function __construct(Comment $comment)
19 |    {
20 |        $this->comment = $comment;
21 |    }
22 | }

```

Neste exemplo ao usar o **Event CommentedPost** precisa passar obrigatoriamente no construtor um objeto do **Model Comment**, que é o próprio comentário em si.

NOTA: Observe que o atributo **\$comment** ficou como público, porque vamos acessar diretamente essa propriedade nos Listeners. Mas, se preferir deixar como privado e criar um método que retorna este valor, também pode, veja o exemplo:

```

1 | namespace App\Events;
2 |
3 | use App\Models\Comment;
4 | use Illuminate\Queue\SerializesModels;
5 |
6 | class CommentedPost
7 | {
8 |     use SerializesModels;
9 |
10 |    private $comment;
11 |
12 |    /**
13 |     * Create a new event instance.
14 |     *
15 |     * @param \App\Models\Comment $comment
16 |     * @return void
17 |     */
18 |    public function __construct(Comment $comment)
19 |    {
20 |        $this->comment = $comment;
21 |    }
22 |
23 |
24 |    /**
25 |     * Comment
26 |     *
27 |     * @return \App\Models\Comment $comment
28 |     */
29 |    public function comment(): Comment
30 |    {
31 |        return $this->comment;
32 |    }
33 | }

```

Configurar Listeners no Laravel

O próximo passo é configurar os Listeners, que são as ações si, aqui você pode fazer diversas ações, como registrar log, salvar alguma informação no banco de dados, incrementar uma quantidade, enviar um e-mail, enviar SMS e etc.

Neste exemplo vamos inicialmente registrar logs, apenas para demonstrar o Event funcionando, implemente o Listener:

```
1 namespace App\Listeners;
2
3 use Log;
4 use App\Events\CommentedPost;
5
6 class SendMailCommentedPost
7 {
8     /**
9      * Create the event listener.
10     *
11     * @return void
12     */
13     public function __construct()
14     {
15         //
16     }
17
18     /**
19      * Handle the event.
20     *
21     * @param \App\Events\CommentedPost $event
22     * @return void
23     */
24     public function handle(CommentedPost $event)
25     {
26         // Ex1: Access the comment using $event->comment...
27         // Ex2: Access the comment using $event->comment()...
28
29         // Registering log commented post
30         Log::info($event->comment());
31     }
32 }
```

No método construtor do Listener pode definir alguns comportamentos iniciais, como por exemplo injetar algumas classes.

E no método handle() que deve ficar a ação em si, como enviar e-mail, ou registrar log, ou salvar algo no banco de dados, enfim.

NOTA: Neste Listener está injetando automaticamente o objeto do Event CommentedPost, e a partir deste objeto podemos acessar os atributos públicos e métodos do Event CommentedPost;

Disparar Eventos no Laravel

Agora que já criou o Event em `app/Events/CommentedPost.php` e criou o Listener em `app/Listeners/SendMailCommentedPost.php` e já vinculou o Listener ao Event no Provider `app/Providers/EventServiceProvider.php` no atributo **\$listen**, o próximo passo é de fato disparar o evento.

Exemplo prático:

```

1 namespace App\Http\Controllers\Posts;
2
3 use App\Models\Post;
4 use App\Events\CommentedPost;
5 use App\Http\Controllers\Controller;
6
7 class MessageController extends Controller
8 {
9     /**
10      * Ship the given order.
11      *
12      * @param int $orderId
13      * @return Response
14      */
15     public function store(Request $request, Post $post)
16     {
17         $post = $post->find($request->id);
18
19         $comment = $post->comments()->create($request->all());
20
21         // Dispatching Event
22         event(new CommentedPost($comment));
23
24         return redirect()
25             ->route('posts.show', $post->id)
26             ->withSuccess('Comentário realizado com sucesso!');
27     }
28 }

```

Agora ao fazer um novo comentário por usar o helper **event()** vai automaticamente disparar o nosso evento, que por sua vez vai disparar os Listeners (Ouvintes) e fazer as ações necessárias, neste simples exemplo vai registrar o log com o comentário.

Esse exemplo continua neste próximo tutorial

(<https://blog.especializati.com.br/eventos-no-laravel-parte-2/>)

Compartilhe:  (<https://www.facebook.com/sharer/sharer.php?u=https://blog.especializati.com.br/aprenda-como-trabalhar-com-eventos-no-laravel/>)  (<https://twitter.com/intent/tweet?text=https://blog.especializati.com.br/aprenda-como-trabalhar-com-eventos-no-laravel/>)  (<https://plus.google.com/share?url=https://blog.especializati.com.br/aprenda-como-trabalhar-com-eventos-no-laravel/>)



(<https://www.especializati.com.br/curso-laravel-55>)

Gostou deste conteúdo?

Junte-se a milhares de profissionais de ti inteligentes e seja o primeiro a receber as nossas novidades e dicas!



Receber



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

Queues no Laravel (Filas) há 3 anos • 3 COMENTÁRIOS Imagine uma aplicação recebendo uma demanda muito grande de ...	Laravel: Redirecionar Após a Autenticação há 2 anos • 2 COMENTÁRIOS Quando usamos o mecanismo padrão de autenticação do Laravel, ...	Eventos no Laravel – Parte 2 há 3 anos • 2 COMENTÁRIOS Esse tutorial é a continuação do tutorial sobre Eventos no ...	Laravel Global há 3 anos No último sobre Laravel,
--	---	--	--

1 COMENTÁRIO

especializati

Disqus' Privacy Policy

1 Iniciar sessão ▾

♥ Recomendar 1

🐦 Tweet

f Partilhar

Mostrar primeiro os mais votados ▾



Escreva o seu comentário...

INICIE SESSÃO COM O

OU REGISTE-SE NO DISQUS ?

Nome



samhk222 • há 2 anos

Muito foda, valeu!

3 ^ | ▾ • Responder • Partilhar ›

✉ Subscriver

D Acerca do DisqusAdicionar o DisqusAdicionar

⚠ Do Not Sell My Data

CONTEÚDOS RELACIONADO:



LARAVEL: REDIRECIONAR APÓS A AUTENTICAÇÃO

POR: CARLOS FERREIRA

especializaTi

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



([HTTPS://BLOG.ESPECIALIZATI.COM.BR/LARAVEL-BLADE/](https://blog.especializati.com.br/laravel-blade/))



([HTTPS://BLOG.ESPECIALIZATI.COM.BR/ENVIO-DE-E-MAILS-NO-LARAVEL/](https://blog.especializati.com.br/envio-de-e-mails-no-laravel/))

especializaTi | BLOG

YouTube f t in G
(<https://www.youtube.com/watch?v=117t0k1E-410>)
sub_confirmation=1)

Todos os direitos reservados © 2021 - EspecializaTi. É proibida a reprodução total ou parcial deste conteúdo.