**Laravel 中的事件监听**

有时候当我们单纯的看 Laravel 手册的时候会有一些疑惑，比如说系统服务下的授权和事件，这些功能服务的应用场景是 > 什么，其实如果没有经历过一定的开发经验有这些疑惑是很正常的事情，但是当我们在工作中多加思考会发现有时候这 >> 些服务其实我们一直都见过。下面就事件、事件监听举一个很简单的例子你就会发现

这个例子是关于文章的浏览数的实现，当用户查看文章的时候文章的浏览数会增加 1，用户查看文章就是一个事件，有了事件，就需要一个事件监听器，对监听的事件发生后执行相应的操作（文章浏览数加 1），其实这种监听机制在 Laravel 中是通过观察者模式实现的

**1. 注册事件以及监听器**

首先我们需要在 app/Providers/ 目录下的 EventServiceProvider.php 中注册事件监听器映射关系，如下：

protected $listen = [

'App\Events\BlogView' => [

'App\Listeners\BlogViewListener',

],

];

然后项目根目录下执行如下命令

php artisan event:generate

该命令完成后，会分别自动在 app/Events 和 app/Listensers 目录下生成 BlogView.php 和 BlogViewListener.php 文件

**2. 定义事件**

<?php

namespace App\Events;

use App\Events\Event;

use App\Post;

use Illuminate\Queue\SerializesModels;

use Illuminate\Contracts\Broadcasting\ShouldBroadcast;

class BlogView extends Event

{

use SerializesModels;

/\*\*

\* Create a new event instance.

\*

\* @return void

\*/

public function \_\_construct(Post $post)

{

$this->post = $post;

}

/\*\*

\* Get the channels the event should be broadcast on.

\*

\* @return array

\*/

public function broadcastOn()

{

return [];

}

}

**3. 定义监听器**

事件监听器在 handle 方法中接收事件实例，event:generate 命令将会自动在 handle 方法中导入合适的事件类和类型提示事件。在 handle 方法内，你可以执行任何需要的逻辑以响应事件，我们的代码实现如下：

<?php

namespace App\Listeners;

use App\Events\BlogView;

use Illuminate\Queue\InteractsWithQueue;

use Illuminate\Contracts\Queue\ShouldQueue;

use Illuminate\Session\Store;

class BlogViewListener

{

protected $session;

/\*\*

\* Create the event listener.

\*

\* @return void

\*/

public function \_\_construct(Store $session)

{

$this->session = $session;

}

/\*\*

\* Handle the event.

\*

\* @param BlogView $event

\* @return void

\*/

public function handle(BlogView $event)

{

$post = $event->post;

//先进行判断是否已经查看过

if (!$this->hasViewedBlog($post)) {

//保存到数据库

$post->view\_cache = $post->view\_cache + 1;

$post->save();

//看过之后将保存到 Session

$this->storeViewedBlog($post);

}

}

protected function hasViewedBlog($post)

{

return array\_key\_exists($post->id, $this->getViewedBlogs());

}

protected function getViewedBlogs()

{

return $this->session->get('viewed\_Blogs', []);

}

protected function storeViewedBlog($post)

{

$key = 'viewed\_Blogs.'.$post->id;

$this->session->put($key, time());

}

}

**4. 触发事件**

事件和事件监听完成后，我们要做的就是实现整个监听，即触发用户打开文章事件在此我们使用和 Event 提供的 fire 方法，如下：

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Post;

use Illuminate\Support\Facades\Event;

use App\Http\Requests;

use App\Events\BlogView;

use App\Http\Controllers\Controller;

class BlogController extends Controller

{

public function showPost($slug)

{

$post = Post::whereSlug($slug)->firstOrFail();

event (new BlogView($post));

return view('home.blog.content')->withPost($post);

}

}