Laravel - Event Handling

Topics : Laravel
Written on December 25, 2023

In Laravel, event handling is a powerful feature that allows you to decouple various components of your application. Events provide a way to broadcast information about specific occurrences, and listeners respond to these events.

1. Creating Events:

Events in Laravel are classes that represent something that has happened. You can create an event using the `make:event` Artisan command.

```
php artisan make:event OrderShipped
```

This will generate an `OrderShipped` event class in the `app/Events` directory.

2. Defining Event Data:

You can define data that should be passed along with the event. Edit the generated event class to include the necessary properties.

```
class OrderShipped
{
    public $order;

    public function __construct(Order $order)
    {
        $this->order = $order;
    }
}
```

3. Creating Listeners:

Listeners are classes that handle events when they are fired. You can generate a listener using the `make:listener` Artisan command.

```
 php artisan make:listener SendShipmentNotification
```

This will generate a `SendShipmentNotification` listener class in the `app/Listeners` directory.

4. Handling Events in Listeners:

Edit the generated listener class to define how it should handle the event.

```
class SendShipmentNotification
{
    public function handle(OrderShipped $event)
    {
        // Access the order from the event
        $order = $event->order;

        // Send a notification or perform other actions
    }
}
```

5. Registering Event Listeners:

You can register event listeners in the `EventServiceProvider` class.

```
protected $listen = [
    OrderShipped::class => [
        SendShipmentNotification::class,
    ],
];
```

6. Firing Events:

You can fire events using the `event` helper function.

```
event(new OrderShipped($order));
```

7. Queueing Events:

You can queue events to handle them asynchronously.

```
event(new OrderShipped($order));
```

8. Event Subscribers:

Event subscribers provide a way to listen to multiple events in a single class. You can generate an event subscriber using the `make:subscriber` Artisan command.

```
php artisan make:subscriber OrderSubscriber
```

9. Defining Subscribed Events:

Edit the generated subscriber class to define the events it should listen to.

```
class OrderSubscriber
{
    protected $events = [
        OrderShipped::class => [
            SendShipmentNotification::class,
        ],
    ];
}
```

10. Manually Subscribing Listeners:

You can manually subscribe listeners in the `EventServiceProvider` class.

```
public function boot()
{
    parent::boot();

    Event::listen(
        OrderShipped::class,
        [SendShipmentNotification::class, 'handle']
    );
}
```