Events

Events in Laravel

Severin Mengers

29.06.22

Universität Tübingen

Table of contents

Intro

Events

- Events implement an observer pattern
- allow to subscribe/listen to events
- Decoupling various aspects of an application

Example
For example decouple login-logic from login-logging.
Event fired, when login is done and logger listens to it.

Event vs. Listener vs. Subscriber

- Event basically data container with information about event
  Defined in *App\Events*
- Listener one class listening to one specific event
  Defined in *App\Listeners*
  Handle-Method is called
- Subscriber class subscribing multiple events
  Defined in *App\Listeners*

Registering

Registering events and listeners

- Registering events and listeners EventServiceProvider's *$listen* property

Listing 1: App\Providers\EventServiceProvider.php

```
protected $listen = [
    ButtonClickedAt::class => [
        RegisterButtonClickedAt::class,],];
```

With artisan

```
$ php artisan event:generate # events from Provider
$ php artisan make:event ButtonClickedAt
$ php artisan make:listener RegisterButtonClickedAt
      --event=ButtonClickedAt
```

Registering subscribers

Registering subscribers in EventServiceProvider’s $subscribe property

Listing 2: App\Providers\EventServiceProvider.php

```
protected $subscribe = [
    ButtonEventSubscriber::class,
];
```

No artisan command.

Manually registering events

- Defined in *EventServiceProvider's* boot method.
- Typically not used.

Listing 3: App\Providers\EventServiceProvider.php:boot()

```
Event::listen(
        ButtonClickedAt::class,
        [RegisterButtonClickedAt::class, 'handle']);
Event::listen(function (ButtonClickedAt $event) {});
```

Wildcard Events

- Use case of manually defined events
- Other use case: Wildcard events

Listing 4: App\Providers\EventServiceProvider.php:boot()

```
Event::listen('event.*',
        function ($eventName, array $data) {
    //
});
```

Events

Event

- Events basically container for app\Model.

Listing 5: app\event\ButtonClickedAt.php

```
class ButtonClickedAt
{
    public $time; //Model instance
    public function __construct($time){
        $this->time = $time$;
}}
```

Listeners

Listener

Listing 6: app\event\RegisterButtonClickedAt.php

```
class RegisterButtonClickedAt
{
    public function __construct() {}

    public function handle(ButtonClickedAt $event){}
}
```

- handle function to handle event
- return false to stop propagation of event

Subscribers

- Multiple Listeners in one class

Listing 7: app\listeners\MySubscriber.php

```
class UserEventSubscriber{
  public function handleUserLogin($event) {}
  public function handleUserLogout($event) {}
  public function subscribe($events){
    return [
        Login::class => 'handleUserLogin',
        Logout::class => 'handleUserLogout',
    ];}}
```

Dispatching Events

Dispatching Events

- static dispatch method
- Made available by *Illuminate\Foundation\Events\Dispatchable* trait

Listing 8: app\controller\MyController.php

```
ButtonClickedAt::dispatch($time);
ButtonClickedAt::dispatchIf($condition, $time);
ButtonClickedAt::dispatchUnless($condition, $time);
```