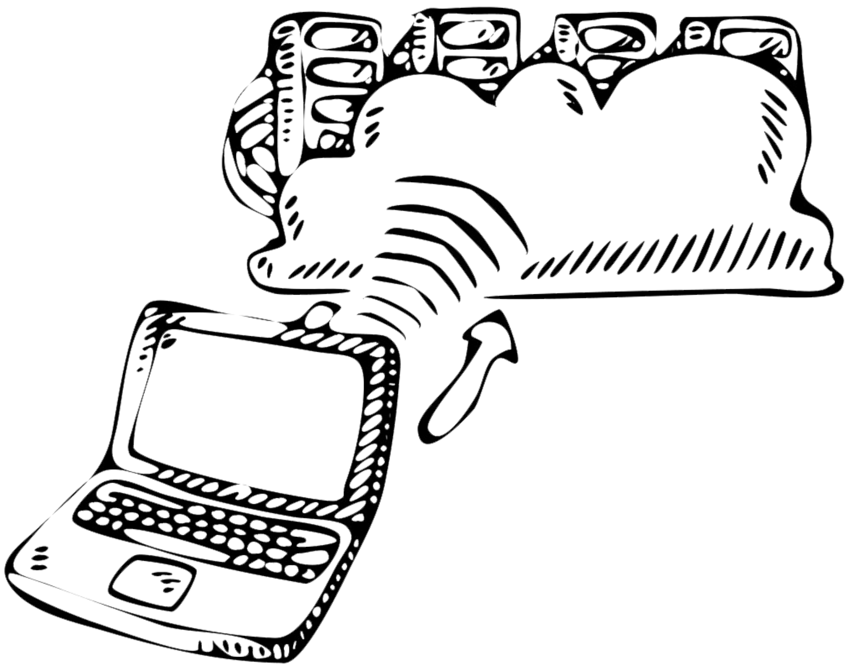
DEV DOJO

Laravel

Table of Contents

Laravel Requests



A zombie developer doesn't handle requests efficiently, whereas a Laravel developer uses the built-in Request class to handle input sanitization and security.

Requests allow our app to receive messages from the client. If these are not handled well or efficiently it can lead to a deteriorating user experience and could have significant vulnerabilities.

Requests

Laravel is built on top of PHP, which is a server-side scripting language.

This server side language can accept requests from the client (web browser) such as submitting a form or even requesting a route URL.

When a user types in a URL in their browser, the browser will then contact or send a request to the server. So, when data is sent from the browser to the server it is called a **Request**.

We've used the **Request** object in a previous section. Back in section 4, we used it to capture the id of a zombie:

```
<?php

use Illuminate\Http\Request;

Route::delete('/zombie', function(Request $request){
    $id = $request->id;
    Zombie::destroy($id);
});
```

So, whenever we want to use the Request Class we need to make sure to specify that we want to use the `` `Illuminate\Http\Request` `` namespace, then we can simply pass the (Request $request) object as a parameter.

The request class offers us a ton of awesome information including:

- The Request URI
- The Request Method
- The Request Input
- The Request Cookies
- The Request Files

The Request URI

To get the current request URI we could do the following:

```
$uri = $request->path();
```

So, if we tried to access the following route: `site.com/zombie/1`, the `$uri` in the above example would be `zombie/1`. Another cool thing that we can do is check if we are currently on a particular route, like this:

```
if ($request->is('zombie/*')) {
    // we have hit the zombie/{id} route
}
```

Inside of the `is` method we can use an asterisk `*` as a wildcard.

If we wanted to fetch the full URL instead we could simply get it like this:

```
$url = $request->url();
```

Request Method

Now, let's say we wanted to figure out what kind of request our application is recieving.

Is it a GET, POST, PUT, or DELETE method? Easy peasy, we can do that like this:

```
$method = $request->method();
```

Or we could use the isMethod function to check if it is a certain request:

```
if ($request->isMethod('post')) {
    // we have a post method
}
```

The Request Input

This is the request method that we have used in the previous sections. This is where we get input data submitted by a form:

```
$zombie_name = $request->input('name');
```

We could simplify this even more and specify `` `$request->name` `` instead of `` `$request->input('name')` ``, like so:

```
$zombie_name = $request->name;
```

We could also get all the input data as an array by doing the following:

```
$input = $request->all();
```

Finally, we could do a quick check to see if the request has a certain input value:

```
if ($request->has('name')) {
    //
}
```

The Request Cookies

To retrieve a cookie value from our request we could simply do the following:

```
$cookie = $request->cookie('name');
```

The Request Files

We could also use the Request class to retrieve uploaded files like so:

```
$file = $request->file('name');
```

And we could check if the request has a file:

```
if ($request->hasFile('name')) {
    //
}
```

There are a few more request types that are provided by the Laravel Request class. Be sure to checkout the full documentation (https://laravel.com/docs/requests) to learn more.

We just learned a few of the awesome things that the Request class offers and we know how our app can accept requests, but what can it do after it gets a request? Well, it can also send a response. Let's learn about **Responses** in the next section.

Previous Section **Next Section**