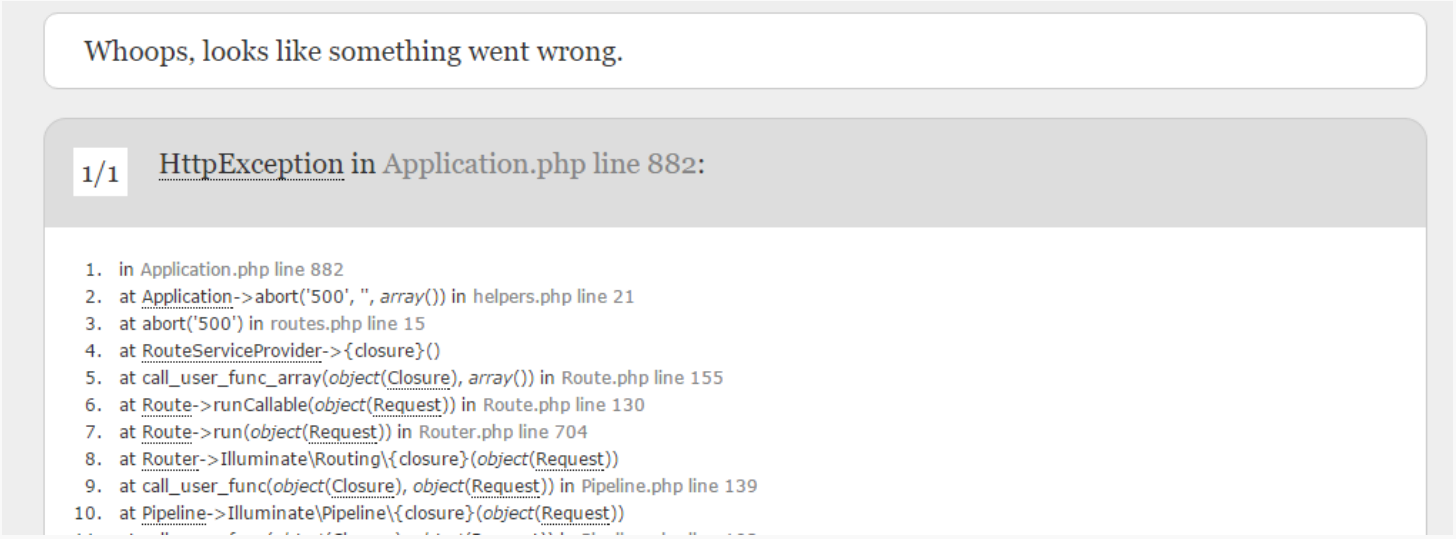
Laravel nos ofrece una forma muy conveniente de manejar todos los errores que pueda generar nuestra aplicación como 404 (Página no encontrada), 401 (Permiso denegado), 500 (Error interno del servidor) y cualquier otros posibles errores. No sólo podemos capturar y evaluar cada excepción sino también podemos devolver una vista personalizada para cada uno de ellas. En la lección de hoy aprenderemos a capturar dichas excepciones para acceder a su información y poder retornar la respuesta adecuada.

Gracias al helper de Laravel abort() podemos forzar a nuestra aplicación a devolver cualquier tipo de error cuando lo necesitemos, pasando como parámetro el código de error que queremos reproducir. Esto, es justo lo que haremos para simular las condiciones que nos lleven a obtener las excepciones y poder manejar los errores.

Editemos un poco el archivo routes.php dentro de app\Http para ver esto con mayor detalle:

```
1 Route::get('error', function(){
2     abort(500);
3 });
```

Ahora si ingresas desde la url de tu navegador a *url_de_tu_proyecto/error* veras algo similar a esto:



Lo ultimo que queremos es mostrar a nuestros usuarios una página con un error confuso y sin estilo, en lugar de ello lo más indicado sería devolver un mensaje más amigable o cualquier otro tipo de información más detallada al usuario.

Creando una nueva vista de error

Dentro de public/resources/views/errors podemos alojar todas las vistas que deseamos mostrar para cada tipo de error dentro de nuestra aplicación:

```
<html>
    <head>
        <title>Be right back.</title>
        <link href="https://fonts.googleapis.com/css?family=Lato:100"
rel="stylesheet" type="text/css">
        <style>
            html, body {
                height: 100%;
            }
            body {
                margin: 0;
                padding: 0;
                width: 100%;
                color: #B0BEC5;
                display: table;
                font-weight: 100;
                font-family: 'Lato';
            }
            .container {
                text-align: center;
                display: table-cell;
                vertical-align: middle;
            }
            .content {
                text-align: center;
                display: inline-block;
            }
            .title {
                font-size: 72px;
                margin-bottom: 40px;
            }
```

```
        </style>
    </head>
    <body>
        <div class="container">
            <div class="content">
                <div class="title">Be right back.</div>
            </div>
        </div>
    </body>
</html>
```

Capturando y manejando excepciones

Esto lo hacemos en el archivo App\Exceptions\Handler.php dentro de la función render()

```
    public function render($request, Exception $e)
    {
        //Code here

        return parent::render($request, $e);
    }
```

Como puedes ver $e es una instancia de la clase Exception que entre otras funcionalidades nos permite acceder al código de la excepción a través del método getStatusCode().

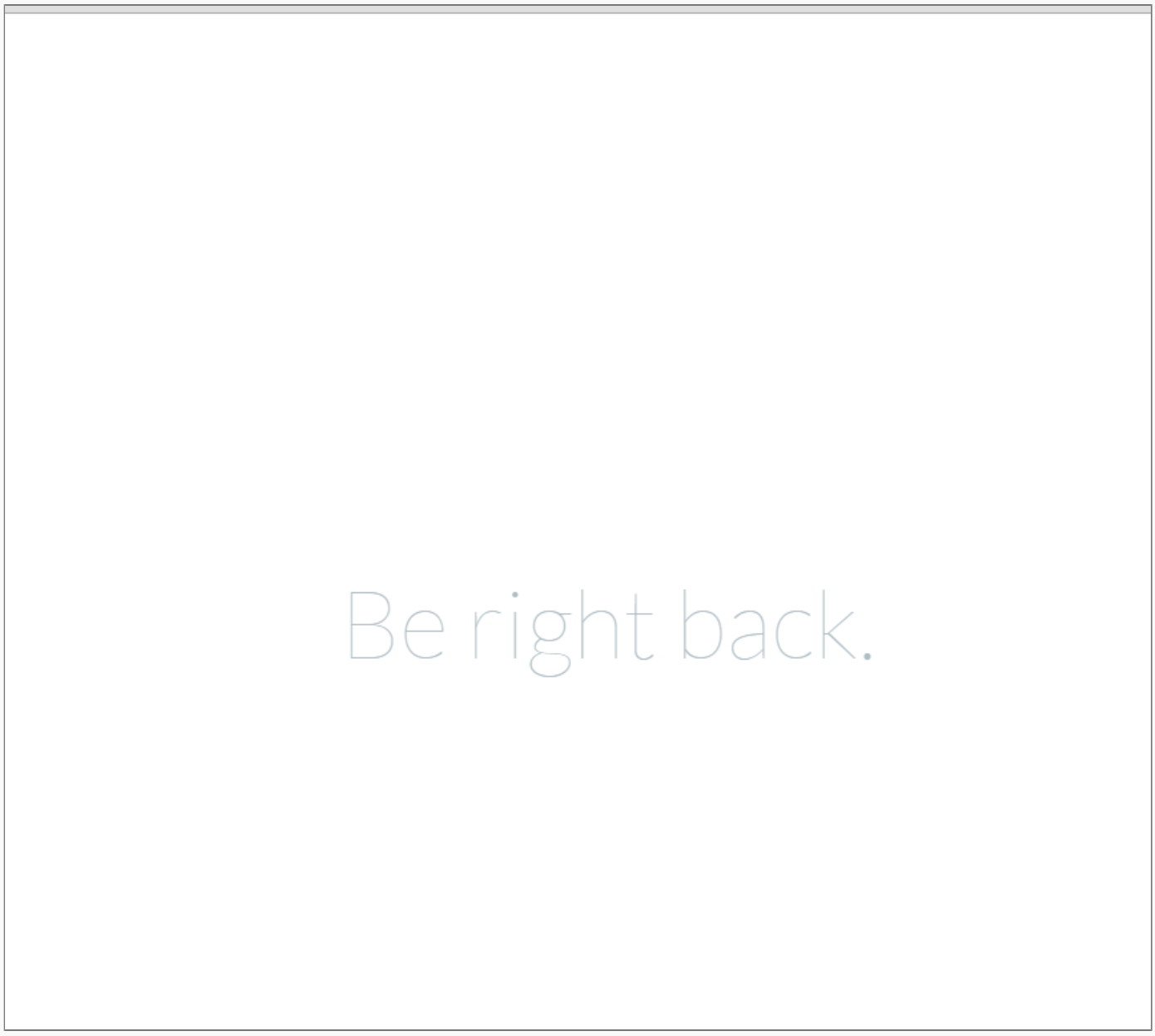
```
    public function render($request, Exception $e)
    {

        if ($e->getStatusCode() == 500) {
            return response()->view('errors.500', [], 500);
        }

        return parent::render($request, $e);
    }
```

Cuando la aplicación detecte que se trata de un error del código específico (500 en nuestro caso) devolverá como respuesta la vista del error; en este caso pasamos como primer parámetro la ubicación de la vista, como segundo los parámetros que deseamos enviar a dicha vista y como tercer el código de la respuesta que estamos enviando.

El resultado final será algo como esto: *url_de_tu_proyecto/error*



Puedes hacer esto para cualquier código de error que necesites, en algunos casos específicos necesitamos evaluar no sólo el código de error recibido sino la instancia devuelta por dicho error, por ejemplo

ModelNotFoundException.

Para esos casos evaluamos si la instancia recibida coincide con una instancia de la clase que deseamos evaluar con *isinstanceof*

```
1    public function render($request, Exception $e)
2    {
3        if ($e instanceof Illuminate\Database\Eloquent\ModelNotFoundException) {
4            // Response here
5        }
6        return parent::render($request, $e);
7    }
```

> **Puedes ver un ejemplo más claro de este último en API REST en Laravel 5.1 – Validaciones y Excepciones**

Espero que te haya gustado esta lección, no olvides compartir el contenido en redes sociales. Recuerda que puedes escribir todas tus dudas en la sección de comentarios.