Introducción a las vistas en Laravel 5

Cómo se trabaja con vistas en Laravel 5, creamos las primeras vistas y las llamamos desde el sistema de enrutado.

En este artículo del vamos a comenzar a tratar el tema de las vistas. Las vistas no son más que los archivos PHP desde donde tenemos que realizar la salida de la aplicación. Es un concepto que esperamos que ya se tenga en la cabeza cuando estamos introduciéndonos en Laravel, puesto que no pertenece en si al framework PHP sino al MVC en general.

En este artículo veremos cómo crear nuestras primeras vistas en Laravel 5 y como invocarlas para crear una salida de la aplicación web totalmente personalizada.

Qué son las vistas

Como decíamos, la mayoría seguro entenderá este concepto, no obstante, vamos a exponerlo rápidamente para aquel que no sepa de qué estamos hablando. Las vistas son una de las capas que tiene el sistema MVC, que trata de la separación del código según sus responsabilidades. En este caso, las vistas mantienen el código de lo que sería la capa de presentación.

Como capa de presentación, las vistas se encargan de realizar la salida de la aplicación que generalmente en el caso de PHP será código HTML. Por tanto, una vista será un archivo PHP que contendrá mayoritariamente código HTML, que se enviará al navegador para que éste renderice la salida para el usuario.

Nota: En la práctica una vista podrá tener cualquier tipo de salida, no solo HTML. Hay ocasiones que será código PHP para generar una imagen, un archivo de texto o cualquier otra necesidad. Por ejemplo, si ante una solicitud el servidor debe enviar como respuesta datos en notación JSON, esos datos se escribirán mediante una vista.

Dónde almacenar las vistas en Laravel 5

Existe una carpeta en el proyecto que es donde debemos colocar las vistas en Laravel. Está en "resources/views". Si navegas a esa carpeta observarás que dentro ya hay diversos archivos, incluso directorios. Esto es porque las vistas se pueden organizar por carpetas, para mantener agrupadas las de cada una de las secciones de la aplicación. Nosotros podemos hacer lo mismo, o dejarlas sueltas en el directorio view.

Además verás que las vistas tiene una extensión ".blade.php". Esto hace referencia a que es un archivo de salida que usa el motor de plantillas "Blade", el oficial de Laravel.

Ten en cuenta que, a pesar de la extensión ".blade.php", las vistas no dejan de ser archivos PHP donde podríamos colocar HTML plano mezclado con códigos PHP. De hecho, podríamos perfectamente nombrar a nuestras vistas como ".php" y el tratamiento que

haremos para invocarlas será exactamente el mismo que si tienen la extensión ".blade.php". Es más, usar Blade es opcional, por lo que para mantener la simplicidad de nuestras primeras vistas no vamos a usar el motor de plantillas.

Así pues, para construir nuestra primera vista, simplemente creamos un archivo llamado "algo.php" en la carpeta "resources/views/". Dentro le colocamos cualquier documento HTML, con cualquier contenido. Con eso ya tenemos la vista lista para ser utilizada.

Como decíamos, el código de ese archivo "algo.php" de momento es indiferente, pero por si alguien necesita la aclaración sería algo como esto:

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title>Esto es una vista de prueba</title>
</head>
<body>
        <h1>Vista "algo"</h1>
        <p>Esta es mi primera vista en Laravel</p>
</body>
</html>
```

Como has observado, la vista no contiene ningún código PHP, porque de momento no lo necesitamos para probar. Es un HTML plano, pero no obstante, sí debemos respetar la extensión ".php" o ".blade.php".

Cómo invocar una vista en Laravel 5

Ahora toca usar la vista creada en el paso anterior. Lo vamos a hacer desde el propio sistema de rutas, por simplificarnos la vida en este primer ejemplo. De hecho es como se invoca a la vista "welcome" en la ruta predeterminada que podremos ver al instalar el framework.

Nota: Aunque en una arquitectura MVC muchos desarrolladores prefieren que sean los controladores quienes invoquen a las vistas, nosotros nos vamos a saltar de momento ese paso porque realmente Laravel no lo necesita, pero sobre todo porque todavía no hemos llegado a explicar los controladores. En lugar de cargar la vista desde el controlador la vamos a invocar directamente desde el sistema de routing.

La corrección de esta operación sería discutible. No es lo habitual pero en ocasiones puede ser adecuada, por ejemplo que sea una vista que no necesita datos externos para mostrarse a sí misma. Pero esto es otra discusión diferente al objetivo de este artículo.

En el siguiente código realizamos la invocación a una vista. Debes apreciar la función view() dentro del closure, a la que enviamos el nombre de la vista a mostrar.

```
Route::get('algo',function(){
returnview('algo');
});
```

Como has podido ver, view() es una función global, un helper global en Laravel, que se encarga de cargar una vista y devolver la salida producida por ella.

El nombre de la vista que estamos cargando es "algo". Para que no nos dé un error la vista deberá estar creada dentro de la carpeta "resources/views", con esta ruta completa:

```
resources/views/algo.php
```

O bien:

```
resources/views/algo.blade.php
```

Nota: Si en algún momento queremos preguntar si una vista existe antes de cargarla, podemos usar la función view() de esta manera:

```
view()->exists('calendario.mes')

Si no se le envían parámetros a view() se recibe un objeto que tiene una
serie de métodos útiles para vistas. El método exists() devuelve un true
o false, dependiendo de si existe o no una vista indicada.
```

Organizar las vistas por carpetas

Es común que queramos poner todas las vistas que tengan que ver con la misma sección del sitio en una misma carpeta, o todas las vistas de los correos electrónicos enviados por la aplicación, por ejemplo.

Las carpetas las situaremos dentro de "resources/views" y en ellas colocaremos los archivos php de las vistas, como se ha descrito antes.

Por ejemplo crea una vista llamada index.php y colócala en otro directorio dentro de views. Su ruta sería algo como esto:

```
resources/views/otro/index.php
```

Ahora invocaremos esa vista indicando la ruta donde se encuentra, desde el directorio views. Omitimos de nuevo la extensión, ya sea ".php" o ".blade.php".

```
Route::get('/otro',function(){
returnview('otro/index');
});
```

Como alternativa podemos especificar la ruta de la vista con un punto en lugar de una barra.

```
Route::get('/otro',function(){
returnview('otro.index');
});
```

Pasar datos a las vistas en Laravel

Seguro que lo siguiente que te preguntabas era cómo pasar los datos a las vistas. Si tienes que pasar datos para que las vistas los representen, los enviarás a través de la función global view() como un array asociativo.

```
view('calendario.eventos',[
        'mes'=>$mes,
        'ano'=>$ano,
        'eventos'=>$eventos
]);
```

Una vez dentro de la vista los recoges en el ámbito global, a partir de las llaves de los elementos del array de datos. En el ejemplo anterior $mes, $ano o $eventos.

```
<p>
    Estás viendo el mes <?= $mes; ?> y el año <?= $ano; ?>.
</p>
```

Nota: Hemos hablado de las plantillas Blade, pero advirtiendo que las vamos a ver en detalle más adelante. Sin embargo es útil que mencionemos una estructura de sintaxis de este tipo de plantillas que verás sin duda en varios ejemplos por ahí y que nosotros también pensamos usar en el futuro, que es el volcado de datos que tienes en variables en el contenido de la vista.

El mismo código que acabamos de ver en el párrafo anterior a esta nota, en el que mostramos el valor del mes y el año, podríamos haberlo escrito así con la sintaxis de Blade:

```
<p>
    Estás viendo el mes {{$mes}} y el año {{$ano}}.
</p>
```

Como se puede apreciar, simplemente sustituyes los tag de cierre y de apertura de código PHP y el "echo" por unas dobles llaves que engloban aquello que quieres volcar a la vista. Es una sintaxis que mejora la legibilidad del código.

Para usar esa sintaxis recuerda que debes tener una plantilla Blade, por lo que el archivo lo tendrás que nombrar necesariamente con extensión ".blade.php".