Vistas

Las vistas son la forma de presentar el resultado (una pantalla de nuestro sitio web) de forma visual al usuario, el cual podrá interactuar con él y volver a realizar una petición. Las vistas además nos permiten separar toda la parte de presentación de resultados de la lógica (controladores) y de la base de datos (modelos). Por lo tanto no tendrán que realizar ningún tipo de consulta ni procesamiento de datos, simplemente recibirán datos y los prepararán para mostrarlos como HTML.

Definir vistas

Las vistas se almacenan en la carpeta `resources/views` como ficheros PHP, y por lo tanto tendrán la extensión `.php`. Contendrán el código HTML de nuestro sitio web, mezclado con los assets (CSS, imágenes, Javascripts, etc. que estarán almacenados en la carpeta `public`) y algo de código PHP (o código *Blade* de plantillas, como veremos más adelante) para presentar los datos de entrada como un resultado HTML.

A continuación se incluye un ejemplo de una vista simple, almacenada en el fichero `resources/views/home.php`, que simplemente mostrará por pantalla `¡Hola <nombre>!`, donde `<nombre>` es una variable de PHP que la vista tiene que recibir como entrada para poder mostrarla.

```
<html>
    <head>
        <title>Mi Web</title>
    </head>
    <body>
        <h1>¡Hola <?php echo $nombre; ?>!</h1>
    </body>
</html>
```

Referenciar y devolver vistas

Una vez tenemos una vista tenemos que asociarla a una ruta para poder mostrarla. Para esto tenemos que ir al fichero `routes.php` como hemos visto antes y escribir el siguiente código:

```
Route::get('/', function()
{
    return view('home', array('nombre' => 'Javi'));
});
```

En este caso estamos definiendo que la vista se devuelva cuando se haga una petición tipo GET a la raíz de nuestro sitio. El único cambio que hemos hecho con respecto a lo que vimos en la sección anterior de rutas ha sido en el valor devuelto por la función, el cual genera la vista usando el método `view` y la devuelve. Esta función recibe como parámetros:

- El nombre de la vista (en este caso `home`), el cual será un fichero almacenado en la carpeta `views`, acordaros que la vista anterior de ejemplo la habíamos guardado en `resources/views/home.php`. Para indicar el nombre de la vista se utiliza el mismo nombre del fichero pero sin la extensión `.php`.
- Como segundo parámetro recibe un array de datos que se le pasarán a la vista. En este caso la vista recibirá una variable llamada `$nombre` con valor "Javi".

Como hemos visto para referenciar una vista únicamente tenemos que escribir el nombre del fichero que la contiene pero sin la extensión `.php`. En el ejemplo, para cargar la vista almacenada en el fichero `home.php` la referenciamos mediante el nombre `home`, sin la extensión `.php` ni la ruta `resources/views`.

Las vistas se pueden organizar en sub-carpetas dentro de la carpeta `resources/views`, por ejemplo podríamos tener una carpeta `resources/views/user` y dentro de esta todas las vistas relacionadas, como por ejemplo `login.php`, `register.php` o `profile.php`. En este caso para referenciar las vistas que están dentro de sub-carpetas tenemos que utilizar la notación tipo "*dot*", en la que las barras que separan las carpetas se sustituyen por puntos. Por ejemplo, para referenciar la vista `resources/views/user/login.php` usaríamos el nombre `user.login`, o la vista `resources/views/user/register.php` la cargaríamos de la forma:

```
Route::get('register', function()
{
    return view('user.register');
});
```

Pasar datos a una vista

Como hemos visto, para pasar datos a una vista tenemos que utilizar el segundo parámetro del método `view`, el cual acepta un array asociativo. En este array podemos añadir todas la variables que queramos utilizar dentro de la vista, ya sean de tipo variable normal

(cadena, entero, etc.) u otro array o objeto con más datos. Por ejemplo, para enviar a la vista `profile` todos los datos del usuario cuyo `id` recibimos a través de la ruta tendríamos que hacer:

```
Route::get('user/profile/{id}', function($id)
{
    $user = // Cargar los datos del usuario a partir de $id
    return view('user.profile', array('user' => $user));
});
```

Laravel además ofrece una alternativa que crea una notación un poco más clara. En lugar de pasar un array como segundo parámetro podemos utilizar el método `with` para indicar una a una las variables o contenidos que queremos enviar a la vista:

```
$view = view('home')->with('nombre', 'Javi');

$view = view('user.profile')
            ->with('user', $user)
            ->with('editable', false);
```