Plantillas mediante *Blade*

Laravel utiliza *Blade* para la definición de plantillas en las vistas. Esta librería permite realizar todo tipo de operaciones con los datos, además de la sustitución de secciones de las plantillas por otro contenido, herencia entre plantillas, definición de *layouts* o plantillas base, etc.

Los ficheros de vistas que utilizan el sistema de plantillas *Blade* tienen que tener la extensión `.blade.php`. Esta extensión tampoco se tendrá que incluir a la hora de referenciar una vista desde el fichero de rutas o desde un controlador. Es decir, utilizaremos `view('home')` tanto si el fichero se llama `home.php` como `home.blade.php`.

En general el código que incluye *Blade* en una vista empezará por los símbolos `@` o `{{`, el cual posteriormente será procesado y preparado para mostrarse por pantalla. Blade no añade sobrecarga de procesamiento, ya que todas las vistas son preprocesadas y cacheadas, por el contrario nos brinda utilidades que nos ayudarán en el diseño y modularización de las vistas.

Mostrar datos

El método más básico que tenemos en *Blade* es el de mostrar datos, para esto utilizaremos las llaves dobles ({{ }}) y dentro de ellas escribiremos la variable o función con el contenido a mostrar:

```
Hola {{ $name }}.
La hora actual es {{ time() }}.
```

Como hemos visto podemos mostrar el contenido de una variable o incluso llamar a una función para mostrar su resultado. *Blade* se encarga de escapar el resultado llamando a `htmlentities` para prevenir errores y ataques de tipo XSS. Si en algún caso no queremos escapar los datos tendremos que llamar a:

```
Hola {!! $name !!}.
```

> Nota: En general siempre tendremos que usar las llaves dobles, en especial si vamos a mostrar datos que son proporcionados por los usuarios de la aplicación. Esto evitará que inyecten símbolos que produzcan errores o inyecten codigo javascript que se ejecute sin que nosotros queramos. Por lo tanto, este último método solo tenemos que utilizarlo si estamos seguros de que no queremos que se escape el contenido.

Mostrar un dato solo si existe

Para comprobar que una variable existe o tiene un determinado valor podemos utilizar el operador ternario de la forma:

```
{{ isset($name) ? $name : 'Valor por defecto' }}
```

O simplemente usar la notación que incluye *Blade* para este fin:

```
{{ $name or 'Valor por defecto' }}
```

Comentarios

Para escribir comentarios en *Blade* se utilizan los símbolos `{{--` y `--}}`, por ejemplo:

```
{{-- Este comentario no se mostrará en HTML --}}
```

Estructuras de control

Blade nos permite utilizar la estructura `if` de las siguientes formas:

```
@if( count($users) === 1 )
    Solo hay un usuario!
@elseif (count($users) > 1)
    Hay muchos usuarios!
@else
    No hay ningún usuario :(
@endif
```

En los siguientes ejemplos se puede ver como realizar bucles tipo *for*, *while* o *foreach*:

```
@for ($i = 0; $i < 10; $i++)
    El valor actual es {{ $i }}
@endfor

@while (true)
    <p>Soy un bucle while infinito!</p>
@endwhile

@foreach ($users as $user)
    <p>Usuario {{ $user->name }} con identificador: {{ $user->id }}</p>
@endforeach
```

Esta son las estructuras de control más utilizadas. Ademas de estas *Blade* define algunas más que podemos ver directamente en su documentación: http://laravel.com/docs/5.1/blade

Incluir una plantilla dentro de otra plantilla

En *Blade* podemos indicar que se incluya una plantilla dentro de otra plantilla, para esto disponemos de la instrucción `@include` :

```
@include('view_name')
```

Ademas podemos pasarle un array de datos a la vista a cargar usando el segundo parámetro del método `include` :

```
@include('view_name', array('some'=>'data'))
```

Esta opción es muy útil para crear vistas que sean reutilizables o para separar el contenido de una vista en varios ficheros.

Layouts

Blade también nos permite la definición de *layouts* para crear una estructura HTML base con secciones que serán rellenadas por otras plantillas o vistas hijas. Por ejemplo, podemos crear un *layout* con el contenido principal o común de nuestra web (*head*, *body*, etc.) y definir una serie de secciones que serán rellenados por otras plantillas para completar el código. Este *layout* puede ser utilizado para todas las pantallas de nuestro sitio web, lo que nos permite que en el resto de plantillas no tengamos que repetir todo este código.

A continuación de incluye un ejemplo de una plantilla tipo *layout* almacenada en el fichero `resources/views/layouts/master.blade.php` :

```
<html>
    <head>
        <title>Mi Web</title>
    </head>
    <body>
        @section('menu')
            Contenido del menu
        @show

        <div class="container">
            @yield('content')
        </div>
    </body>
</html>
```

Posteriormente, en otra plantilla o vista, podemos indicar que extienda el *layout* que hemos creado (con `@extends('layouts.master')`) y que complete las dos secciones de contenido que habíamos definido en el mismo:

```
@extends('layouts.master')

@section('menu')
    @parent
    <p>Este condenido es añadido al menú principal.</p>
@endsection

@section('content')
    <p>Este apartado aparecerá en la sección "content".</p>
@endsection
```

Como se puede ver, las vistas que extienden un *layout* simplemente tienen que sobrescribir las secciones del *layout*. La directiva `@section` permite ir añadiendo contenido en las plantillas hijas, mientras que `@yield` será sustituido por el contenido que se indique. El método `@parent` carga en la posición indicada el contenido definido por el padre para dicha sección.

El método `@yield` también permite establecer un contenido por defecto mediante su segundo parámetro:

```
@yield('section', 'Contenido por defecto')
```