Definición y uso de formularios

El envío de formularios en Laravel implica, por un lado, definir un formulario, empleando HTML sencillo junto con algunas opciones ofrecidas por Blade. Por otra parte, debemos recoger los datos enviados por el formulario en algún método de algún controlador, y procesarlos adecuadamente.

1. Creación y envío de formularios

Si definimos un formulario en una vista, se define con los conceptos que ya sabemos de HTML. Como único añadido, en el campo `action` del formulario podemos utilizar Blade y la función `route` para indicar el nombre de ruta a la que queremos enviar el formulario.

Veamos, por ejemplo, cómo definir un formulario para dar de alta nuevos libros en nuestro proyecto de *biblioteca*. Creamos una vista llamada `create.blade.php` en la subcarpeta `resources/views/libros`, con un contenido como éste:

```
@extends('plantilla')

@section('titulo', 'Nuevo libro')

@section('contenido')
    <h1>Nuevo libro</h1>

    <form action="{{ route('libros.store') }}" method="POST">

        <div class="form-group">
            <label for="titulo">Título:</label>
            <input type="text" class="form-control" name="titulo"
                id="titulo">
        </div>

        <div class="form-group">
            <label for="editorial">Editorial:</label>
            <input type="text" class="form-control" name="editorial"
                id="editorial">
        </div>

        <div class="form-group">
            <label for="precio">Precio:</label>
            <input type="text" class="form-control" name="precio"
                id="precio">
        </div>

        <div class="form-group">
            <label for="autor">Autor:</label>
            <select class="form-control" name="autor" id="autor">
                @foreach ($autores as $autor)
                    <option value="{{ $autor->id }}">
                        {{ $autor->nombre }}
                    </option>
                @endforeach
            </select>
        </div>

        <input type="submit" name="enviar" value="Enviar"
            class="btn btn-dark btn-block">

    </form>
@endsection
```

Un segundo añadido más que tenemos que tener en cuenta es que Laravel por defecto protege de ataques XSS (*Cross Site Scripting*) de suplantación de identidad, por lo que obtendremos un error de tipo 419 si enviamos un formulario no validado. Para solucionar este problema, basta con utilizar la directiva `@csrf` en el formulario, que añade un campo oculto con un token de validación:

```
<form action="{{ route('libros.store') }}" method="POST">
    @csrf
    ...
</form>
```

En cualquier caso, este formulario se enviará a la ruta indicada. Dado que en nuestro proyecto hemos definido un conjunto de recursos como éste en `routes/web.php`, la ruta ya está automáticamente definida como `libros.store`:

```
Route::resource('libros', LibroController::class);
```

De lo contrario, tendríamos que añadir a mano la ruta correspondiente para recoger el formulario.

Además, debemos redefinir los métodos involucrados en el controlador: por un lado, el método `create` deberá renderizar el formulario anterior. Como necesitamos mostrar el listado de autores para asociar uno al libro, le pasaremos a la vista anterior el listado de autores como parámetro (recuerda incluir el modelo `Autor` con `use`, si no lo has hecho antes):

```
public function create()
{
    $autores = Autor::get();
    return view('libros.create', compact('autores'));
}
```

Por otra parte, el método `store` se encargará de recoger los datos de la petición a través del parámetro `Request` de dicho método. Disponemos de un método `get` para acceder a cada campo del formulario a partir de su nombre:

```
public function store(Request $request)
{
    $libro = new Libro();
    $libro->titulo = $request->get('titulo');
    $libro->editorial = $request->get('editorial');
    $libro->precio = $request->get('precio');
    $libro->autor()->associate(Autor::findOrFail($request->get('autor')));
    $libro->save();

    return redirect()->route('libros.index');
}
```

Podemos emplear también algún método auxiliar de la petición, como `has`, que comprueba si existe un campo con un nombre determinado:

```
public function store(Request $request)
{
    if($request->has('titulo'))
    {
        ...
    }
}
```

Para poder lanzar esta operación de inserción, necesitamos algún enlace que muestre el formulario. Podemos añadir una nueva opción en el menú superior de navegación (archivo `resources/views/partials/nav.blade.php`):

```
<nav class="navbar navbar-expand-lg navbar-dark bg-secondary">
    ...
    <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav">
            ...
            <li class="{{ setActivo('libros.create') }} nav-item">
                <a class="nav-link" href="{{ route('libros.create') }}">
                Nuevo libro</a>
            </li>
        </ul>
    </div>
</nav>
```

2. Actualizaciones y borrados

Por defecto, el atributo `method` de un formulario sólo admite los valores GET o POST. Si queremos enviar un formulario de actualización o borrado, éste debe ir asociado a los métodos PUT o DELETE, respectivamente. Para esto, podemos emplear dentro del mismo formulario la directiva `@method`, indicando el nombre del método que queremos usar:

```
<form ...>
    @csrf
    @method('PUT')
    ...
</form>
```

Por ejemplo, para borrar libros en nuestra aplicación de biblioteca, podríamos añadir un formulario como este en la ficha del libro (vista `resources/views/libros/show.blade.php`):

```
<form action="{{ route('libros.destroy', $libro->id) }}" method="POST">
    @csrf
    @method('DELETE')
    <input type="submit" class="btn btn-danger" value="Borrar libro" />
</form>
```

Observa cómo le pasamos a la ruta el *id* del libro a borrar, para que le llegue como parámetro al método `destroy` del controlador. Dentro de este método, buscamos el libro afectado, lo borramos, y mostramos el listado de libros nuevamente:

```
public function destroy($id)
{
    $libro = Libro::findOrFail($id);
    $libro->delete();
    return redirect()->route('libros.index');
}
```