HTTP Request en Laravel 5

Laravel nos facilita todos los datos de la solicitud actual a través HTTP Request, un objeto sobre el que podremos consultar información sobre el cliente que realiza la solicitud y datos que pueda estar enviando.

Toda aplicación web recibe solicitudes para completar todo tipo de acciones. Cada solicitud que recibe el servidor viene acompañada de una serie de datos, que se envían en el protocolo HTTP. Entre la información que recibe PHP podemos encontrar desde el user-agent del visitante o su IP, hasta datos que viajan en las cabeceras ante una operación post.

Esos datos en Laravel se acceden a través del objeto Request, que podemos recibir en el controlador mediante la inyección de dependencias, o mediante la correspondiente facade. En este artículo te vamos a explicar las bases del trabajo con el objeto Request.

Recibir la solicitud (request)

Comencemos observando cómo se consigue el objeto Request en un controlador, para lo que vamos a hacer mención a un patrón de diseño de programación orientada a objetos, usado en Laravel así como en otra serie de frameworks populares: Inyección de dependencias. En este patrón se busca separar la responsabilidad de creación de los objetos de su uso, simplificando y abstrayéndonos de toda la complejidad que puede suponer crear todas y cada una de las dependencias que tenga un código. Cuando una clase necesite de un objeto para completar sus operaciones no lo construye él, sino que lo recibe en el constructor o en los métodos que lo necesiten.

El hecho de recibir los objetos de los que depende una clase por parámetro es lo que se conoce como inyección. La dependencia es aquello que necesita, que no es más que un objeto de una clase. Además el patrón incluye lo que se llama el contenedor de dependencias o contenedor de servicios, que es la pieza de software encargado de instanciar los objetos que se deben inyectar e inyectarlos en los métodos necesarios.

La clase Request está en el namespace "Illuminate\Http\Request", por lo que un primer paso sería declarar que vamos a usarlo.

```
useIlluminate\Http\Request;
```

A continuación podemos definir, en la cabecera del método donde queramos recibir el objeto request, la correspondiente dependencia, de la clase Illuminate\Http\Request.

```
publicfunctionrecibirPost(Request $request){
// en este punto del código $request es mi objeto HTTP Request.
(inyectado)
}
```

Como puedes ver, para que el inyector de dependencias sepa qué es lo que debe inyectar, estamos definiendo la clase del parámetro $request que no es otra que la clase Request.

Nota: La posibilidad de escribir el tipo de datos de los parámetros de métodos o funciones es algo relativamente nuevo en PHP y toma el nombre de TypeHinting, "Implicación de tipos" en español.

Al informar el tipo observarás que no hemos colocado toda la ruta en la jerarquía del namespace "Illuminate\Http\Request", el motivo es simplemente porque ya habíamos definido un alias de esa clase mediante la sentencia "use Illuminate\Http\Request".

Nota: Si no hiciéramos el alias del namespace con la sentencia use Illuminate\Http\Request, entonces estaríamos obligados a definir la jerarquía de espacios de nombres donde está situada la clase Request, con un código como el que puedes ver a continuación.

```
publicfunctionrecibirPost(\Illuminate\Http\Request$request){
// Esto es equivalente y funcionará aunque no hagas el "use"
}
```

Suponemos que estas nociones de espacios de nombres las tendrás claras, si no es así te recomendamos la lectura de los artículos sobre Namespaces en PHP.

En otros artículos también veremos cómo trabajar con el HTTP Request sin necesidad de la inyección de dependencias, directamente a través de la "facade" (fachada) Request.

Usar el HTTP Request

Una vez ya disponemos del objeto request podemos consultar toda la información disponible acerca de la solicitud. Es tan sencillo como enviar mensajes, invocando sus métodos. En este artículo haremos un ejercicio básico de envío por post, pero antes vamos a probar un par de métodos sencillos.

1. Para recuperar la URI actual de una solicitud (lo que va después del dominio), sobre nuestro objeto request invocaremos el método path().

2. Para recuperar la URL completa de una solicitud (toda la ruta completa, incluyendo el dominio), invocamos el método url().

Así quedaría un método de un controlador que mostrase ambas informaciones de la solicitud.

```
publicfunctionmostrarUriUrl(Request $request){
echo$request->path();
echo "<br>";
echo$request->url();
}
```

Ahora vamos a ver el procedimiento completo, desde la creación de la ruta hasta la codificación del controlador, para recuperar un dato que enviarían por post en una solicitud, recogiendo los valores enviados usando HTTP Request.

En primer paso, debo de registrar una ruta en mi sistema, archivo routes.php.

```
Route::post('recibir','PrimerController@recibirPost');
```

Segundo paso defino el método recibirPost, que debemos de crear dentro de PrimerController (es un controlador que creamos en un ejemplo anterior de este manual de Laravel). En ese método inyecto la dependencia y la recibo con $request.

```
publicfunctionrecibirPost(Request $request){
// código de mi método
}
```

Como tercer paso, ya en el código de implementación del método, puedo acceder a uno de los campos que me envíen por post, de manera independiente.

```
publicfunctionrecibirPost(Request $request){
echo$request->input('id');
}
```

En el caso anterior estamos accediendo a un campo enviado por formulario llamado "id". Pero también existen métodos para recuperar de una sola vez todos los datos enviados en la solicitud.

```
publicfunctionrecibirPost(Request $request){
        $todos_los_datos=$request->all();
}
```

En este caso recuperas todos los datos en forma de array, por lo que si deseas ver su contenido tendrás que usar una función como print_r() o var_dump(), o quizás mejor dd() que es una función específica de Laravel que muestra el contenido de una variable y a continuación para la ejecución del script.

```
publicfunctionrecibirPost(Request $request){
        dd($request->all());
}
```

Nota: Para poner en marcha este ejemplo de una manera cómoda tendrás que usar una extensión como postman, que te permite generar formularios al vuelo. Algo que se explicó en el artículo Verbos en las rutas de Laravel. Otra cosa que tendrás que hacer será comentar la línea donde se accede al middleware de protección csrf, también explicado en el mencionado artículo.

Recibir parámetros de la ruta

Aunque estemos inyectando en el controlador el objeto de la solicitud, clase HTTP Request, no impide recibir otros parámetros en el método de la acción. Esto lo podemos conseguir simplemente creando los parámetros en la acción como siempre.

Nuestra ruta sería algo así:

```
Route::post('editar/{id}','PrimerController@editar');
```

Ahora podremos recibir tanto la request como el parámetro de la ruta {id} en el método de la acción del controlador.

```
publicfunctioneditar(Request $request,$id){
        echo"Recibo $id como parámetro de la ruta.";
        echo"Además recibimos estos datos por formulario: ".implode(',
',$request->all());
}
```

Conclusión

Con lo que hemos visto tenemos material suficiente para poder continuar viendo otras utilidades del frameworkLaravel. Por supuesto, más adelante volveremos sobre el tema de las solicitudes, pues es muy importante.