

Laravel: Session


Sesiones de cliente

Sesiones de Laravel

Dado que las aplicaciones basadas en HTTP no tienen estados (presentan una lectura de lenguaje de marcas), **las sesiones ofrecen una forma de almacenar información del cliente** a través de múltiples solicitudes.

Con Laravel, esta información del usuario generalmente se guarda en un almacenamiento persistente o backend al que se puede acceder desde solicitudes posteriores.

Laravel incluye una variedad de backends o “drivers” de sesión a los que se accede a través de una **API unificada**.



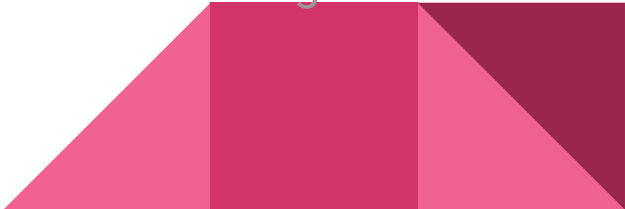
Sesiones de Laravel

El archivo de configuración de sesiones de la aplicación se encuentra en ***config/session.php***.

Se recomienda revisar las opciones disponibles en este archivo. Ver listado de drivers de la siguiente diapositiva.


De manera **predeterminada**, Laravel está configurado para usar el driver de **sesiones de base de datos**.

Nota: Sin embargo, para los test, a fin de evitar almacenar información generada por estos, se suele emplear el driver de array.



Drivers de sesión

La opción de configuración **session driver** define dónde se almacenarán los datos de la sesión para cada solicitud. Laravel incluye los siguientes drivers:

- **file**: las sesiones se almacenan en **storage/framework/sessions**.
 - **cookie**: las sesiones se almacenan en **cookies seguras y cifradas**.
 - **database**: las sesiones se almacenan en una **base de datos relacional**.
 - **memcached / redis**: las sesiones se almacenan en uno de estos almacenes rápidos basados en caché.
 - **dynamodb**: las sesiones se almacenan en AWS DynamoDB.
 - **array**: las sesiones se almacenan en un **array PHP** y no se persisten.
- 

Drivers de sesión - Requisito de sesiones en BD

Cuando se utiliza el driver de sesiones de base de datos, es necesario asegurarse de tener una tabla en la base de datos para contener los datos de la sesión.

Normalmente, esto se incluye en la migración predeterminada `0001_01_01_000000_create_users_table.php` de Laravel; sin embargo, **si por alguna razón no se tiene una tabla de sesiones, se puede usar el comando `make:session-table` de Artisan para generar esta migración:**

```
php artisan make:session-table
```

```
php artisan migrate
```



Session - Extraer datos

Existen dos formas de trabajar con datos de sesión en Laravel:

- A través de una **instancia de *Request*** (solicitud)
- Mediante el helper ***session()***




Session - Extraer datos mediante instancia de Request

Se puede acceder a la sesión a través de una **instancia de Request**, ya sea en una ruta o en un método de controlador. Sobre esta instancia de sesión se puede encadenar el método **get()** con la clave del dato a extraer.

El siguiente ejemplo extrae el valor correspondiente a una clave 'user_name' definida dentro de la sesión a través de la instancia de la solicitud:

```
use Illuminate\Http\Request;
```

```
Route::get('/user_name', function (Request $request) {  
    $userName = $request->session()->get('user_name'); // $userName = $_SESSION['user_name'];  
    // Devuelve el valor almacenado en la sesión con clave 'user_name'  
    return $userName;  
});
```



Session - Extraer datos mediante instancia de Request

Adicionalmente, también se puede:

- **Extraer toda la información** encadenando ***all()***:

```
$data = $request->session()->all();
```

- Revisar si una sesión **tiene un valor declarado** para una clave con ***has()***. Este método devuelve true si un dato está presente Y no es NULL:

```
if ($request->session()->has('users')) { /* ... */ }
```

- Usar el metodo ***missing()*** para determinar si un dato NO está presente en la sesión (devuelve true si no se encuentra):

```
if ($request->session()->missing('users')) { /* ... */ }
```



Session - Extraer datos mediante Request en Controlador

Para entender el siguiente ejemplo sobre extracción de datos de sesión vía Request, es importante recordar que las dependencias (argumentos) de los métodos de los controladores se inyectan automáticamente en Laravel.

Es decir; si tenemos un método:

show(Request \$request, ***string*** \$id)

entonces **el valor de \$id se pasa automáticamente al método *show()*** del controlador cuando la ruta siguiente recibe una solicitud:

```
Route::get('user/{id}', [UserController::class, 'show']);
```



Session - Extraer datos mediante Request en Controlador

```
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use Illuminate\View\View;

class UserController extends Controller {


    // Mostrar el perfil de un usuario dado su id.
    public function show(Request $request, string $id): View {

        $value = $request->session()->get('key');      // key se ha definido con el valor del id del usuario

        // Recupera listado de usuarios en colección "users" de BD, empleando $value.
        // Solo debería contener el usuario con id facilitado
        //...

        // filtra y extrae el usuario buscado vía el ID de la sesión (a modo de doble verificación)
        $user = $this->users->find($id);

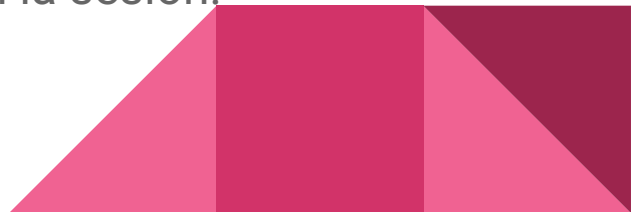
        return view('user.profile', ['user' => $user]);
    }
}
```



Session - Extraer datos mediante helper session()

La función global ***session()*** permite tanto extraer como almacenar datos en sesiones:

- Cuando se llama al helper con **un único argumento de tipo cadena (string)**, **devolverá el valor de esa clave** de sesión.
- ¡OJO! el mismo método, cuando se llama con **un array asociativo** de pares clave => valor, hace que estos **valores se guarden** en la sesión.



Session - Extraer datos con helper session()

Ejemplo de extracción y guardado de datos en sesión al declarar una ruta:

```
Route::get('/home', function () {  
    // Recuperar dato de la sesión  
    $value = session('key');  
});
```



Session - Guardar datos de session

Como antes, se pueden guardar datos de sesión en Laravel:

- A través de una **instancia de *Request*** (solicitud) y método ***put()***
- Mediante el helper ***session()***

Por ejemplo:

// Via instancia de Request

```
$request->session()->put('key', 'value');
```

// Via método helper "session()" con array asociativo (clave => valor)

```
session(['key' => 'value']);
```



Session - push sobre instancia de Request->session()

El método ***push()*** encadenado al método *session()* sobre una instancia de Request sirve para **añadir datos a un array almacenado dentro de la sesión**. No sobrescribe valores previos. Si la clave no existe, crea un nuevo array.

Por ejemplo: si tenemos el array:

```
$user = ['teams' => ['designers', 'managers']];
```

Tras ejecutar:

```
$request->session()->push('user.teams', 'developers');
```

El array contendrá:

```
$user = ['teams' => ['designers', 'managers', 'developers']];
```



Session - pull sobre instancia de Request->session()

El método ***pull()*** sirve para **extraer Y eliminar un dato** de la sesión.

Por ejemplo, la siguiente instrucción elimina el valor de 'key' guardado:

```
$value = $request->session()->pull('key');
```

La clave 'key' se elimina de la sesión tras obtener su valor.



PULL



PUSH



Session - Eliminar datos de sesión

El método ***forget()*** permite **borrar uno o varios datos de la sesión**. Por ejemplo:

// Eliminar una clave y su valor de una sesión:

```
$request->session()->forget('name');
```

// Eliminar múltiples claves (y sus valores) de una sesión vía array de claves:

```
$request->session()->forget(['name', 'status']);
```

Por otro lado, el método **flush** elimina TODA la información de una sesión:

```
$request->session()->flush();
```



Session - Flash() - datos de un solo uso

Para dar un feedback rápido y sencillo, mostrar información condicional o avisos, puede ser útil guardar datos “flash” en la sesión específicos para la siguiente solicitud.

Esto se puede hacer usando el método ***flash()*** de `session()` sobre instancia de Request.

Los **datos guardados en la sesión mediante este método estarán disponibles** de inmediato y **sólo durante la siguiente solicitud HTTP**. Después de esta solicitud subsecuente, los datos se eliminarán.

```
$request->session()->flash('status', '¡Trabajo finalizado!');
```



Session - Regenerar o Reiniciar una sesión

Laravel regenera automáticamente el ID de sesión durante la autenticación si se está utilizando uno de los kits de inicio (como Breeze).

Sin embargo, si se necesita **regenerar manualmente el ID de sesión**, se puede utilizar el método ***regenerate()***:

```
$request->session()->regenerate();
```

Si se necesita **regenerar el ID de sesión Y eliminar todos los datos de la sesión**, se puede utilizar el método ***invalidate()***:

```
$request->session()->invalidate();
```

