01

LARAVEL: SESIONES, Manteniendo información en el servidro

Http un protocolo sin estado

Usando variables de sesión podemos mantener información de acciones anteriores del cliente en el servidor

Sesiones con laravel

Las sesiones son elementos fundamentales en el desarrollo web. Permiten mantener información y estado entre diferentes solicitudes realizadas por un mismo cliente al servidor.

El tiempo en el cual el servidor retiene y administra estos valores asociados a la sesión se queda establecido y depende de diferentes parámetros. También el hecho de que fluyan mensajes entre cliente y servidor lo cual mantiene la sesión activa y permite actualizar esta información

Sesiones con laravel

Laravel inicia de forma automática las sesiones, por lo que nuestra función **session_start** de php no hay que invocar de forma explícita a esta función:

Al hacer una solicitud a una aplicación Laravel, el middleware **StartSession** se encarga de inicializar la sesión antes de que llegue a tu controlador. Este middleware se registra automáticamente en el kernel de la aplicación y se ejecuta en cada solicitud entrante.

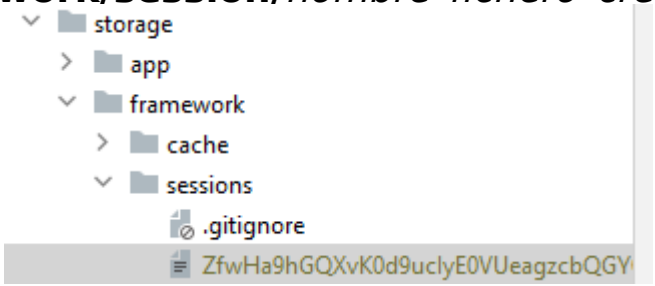
App/Http/Kernel.php

```
. . .
protected $middlewareGroups = [
      'web' => [
            \App\Http\Middleware\EncryptCookies::class,
            \Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::class,
            \Illuminate\Session\Middleware\StartSession::class,
            . . .
```

Sesiones con laravel

Para configurar las sesiones en laravel, existe un fichero de configuración **config/session.php**
En laravel podemos establecer dónde queremos guardar esta información (ficheros, bases de datos, memcache, ...) siendo el fichero el valor tomado por defecto

Por defecto se almacena en el directorio especificado en **storage/framework/session/***nombre_fichero_creado_por_laravel*



https://laravel.com/docs/10.x/session#accessing-the-session

Sesiones con laravel

Para gestionar las sesiones, podemos hacerlo con el helper **session()** o a través del método de un objeto de la clase **Request, session()**
Dentro de un fichero **blade**, puedo acceder a través del helper **session()**
A continuación un listado de acciones sobre sesiones

- session()->has('key'): Verifica si la clave **key** existe en la sesión.
- session()->get('key'): Obtiene el valor de la clave **key** en la sesión.
- session()->put('key', 'value'): Establece un valor para la variable de sesión **key**
- session()->forget('key'): Elimina la variable de sesión **key** (y su valor).
- session()->flush(): Elimina todos los datos de la sesión.
- session()->all(): Obtiene todos los datos almacenados en la sesión como un array.
- session()->pull('key',)'default': Obtiene y elimina un valor de la sesión. Si la clave no existe, se retorna el valor por defecto especificado.
- session()->flash('key', 'value'): Establece un valor flash en la sesión, que estará disponible solo durante la siguiente solicitud. (ideal para pasar menajes)

Generando una variable en el controlador

En nuestra aplicación cuando realizamos una acción, vamos a guardarla en

```
public function store(StoreAlumnoRequest $request)
{

    $valores =  $request->input();
    $alumno = new Alumno($valores);
    $alumno->save();

    session()->flash("status", "Se ha creado el alumno $alumno->nombre");

    $alumnos = Alumno::all();
    return view ("alumnos.listado",["alumnos"=>$alumnos]);


}
```

Leyendo en el servidor

En la vista, si existe la visualizamos

```
@if(session("status"))
        <span>{{session("status")}}</span>
@endif
```

Podemos usar un componente de daisyui

```
<div role="alert" class="alert alert-success">
   <svg xmlns="http://www.w3.org/2000/svg"...
            <!-- Resto del código svg .. -->
  </svg>
   <span>{{session("status")}}</span>
</div>
```

Leyendo en el servidor

Podemos establecer un tiempo para que el alert desaparezca
Esto implica un poco de javascript. Ponemos un id al elemento

```
<div id=”alertSession” role="alert" class="alert alert-success">
   <svg xmlns="http://www.w3.org/2000/svg"...
            <!-- Resto del código svg .. -->
  </svg>
   <span>{{session("status")}}</span>
</div>
```

Y añadimos al final el script

```
<script>
   window.onload=()=>
      setTimeout(()=>
         document.getElementById('alertSesion').style.display='none', 1000);
</script>
```

Leyendo en el servidor

Usando la librería sweet (https://sweetalert2.github.io/)

Para ello podemos instalar el paquete o cargarlo con un CDN

Como CDN, Escribimos en nuestro **layout** , en el head la ubicación de la librería

```
<script src="//cdn.jsdelivr.net/npm/sweetalert2@11"></script>
```

Alternativamente Instalamo2 con **npm**

```
npm install sweetalert2
```

E importamos el fichero en nuestro fichero app.js

```
import Swal from 'sweetalert2'
```

Para la inserción o actualización

Ahora es cuestión de utilizar la librería con la gran variedad de opciones que nos ofrece.

La función de alerte es **swal.fire(texto-opciones)**

Vamos a crear funciones de js para nuestro cometido

```
@if(session("status"))
        <script>
          showAlert("{{session("status")}}")
        </script>
@endif
```

```
<script>
   function showAlert(mensaje) {
      Swal.fire({
         title:"Éxito!!",
         text:mensaje,
         icon: 'info',
         timer:2000
      });
   }
```

Para el borrado

Asociamos a la acción onclick del botón una función de js

```
<form action="/alumnos/{{$alumno->id}}" method="POST">
    @csrf
    @method("DELETE")
    <button class="btn" onclick="confirmDeletion(event, this)" type=" submit">
```

Para el borrado

En la función invocamos a Swal.fire y en las opciones recogemos el resultado de la ventana de confirmación que nos pide y seguimos con la acción

```
function confirmDeletion(event, button) {
    event.preventDefault();
    Swal.fire({
        title: '¿Estás seguro?',
        text: "No podrás revertir esto!",
        icon: 'warning',
        showCancelButton: true,
        confirmButtonColor: '#3085d6',
        cancelButtonColor: '#d33',
        confirmButtonText: 'Sí, borrarlo!'
    }).then((result) => {
        if (result.isConfirmed) {
            // Buscar el formulario más cercano y enviarlo
            button.closest('form').submit();
        }
    });
}
```