
UA . MASTER MOVILES

MÁSTER UNIVERSITARIO EN DESARROLLO DE SOFTWARE PARA DISPOSITIVOS MÓVILES

# PROGRAMACIÓN HIPERMEDIA PARA DISPOSITIVOS MÓVILES

## Laravel 2 - Controladores, filtros y formularios

# CONTENIDO

1. Controladores
2. *Middleware* o Filtros
3. Rutas avanzadas
4. Redirecciones
5. Formularios

# CONTROLADORES

- Permiten separar mucho mejor el código y crear clases (controladores) que agrupen toda la funcionalidad de un determinado recurso.
- Son el punto de entrada de las peticiones de los usuarios y contendrán toda la lógica asociada para:
  - Acceder a la base de datos si fuese necesario.
  - Preparar los datos.
  - Llamar a la vista con los datos asociados.
- Se almacenan en la carpeta "`app/Http/Controllers`" como ficheros PHP.
- Se les añade el sufijo "`Controller`", por ejemplo "`UserController.php`" o "`MoviesController.php`".

# CONTROLADORES

- Ejemplo de controlador básico almacenado en el fichero "`app/Http/Controllers/UserController.php`":

```php
<?php
namespace App\Http\Controllers;

use App\User;


class UserController extends Controller
{
    public function showProfile($id)
    {
        $user = User::findOrFail($id);
        return view('user.profile', ['user' => $user]);
    }
}
```

Es importante el espacio de nombre y añadir las clases a usar.

Controlador base situado en app/Http/Controllers

# CONTROLADORES

- Para **crear un nuevo controlador** podemos usar el siguiente comando de Artisan:

```
php artisan make:controller MoviesController
```

- Este comando creará el controlador `MoviesController.php` dentro de la ruta "`app/Http/Controllers`".
- Lo completará con el código básico para un controlador que hemos visto antes.
- Por defecto el controlador no contendrá ningún método. Más adelante veremos cómo generar los métodos de un controlador tipo RESTful.

# CONTROLADORES

- Para enlazar el controlador con una ruta tenemos que modificar el fichero “routes/web.php”:

```
Route::get('user/{id}', 'UserController@showProfile');
```

- Para generar la URL que apunte a una acción de un controlador tenemos dos opciones:

```
action('UserController@showProfile', [1])
url(“user/”, [1])
```

- Ejemplo de uso en una plantilla:

```
<a href="{{ action('UserController@showProfile') }}">
    ¡Aprieta aquí!
</a>
```

# *MIDDLEWARE* O FILTROS

- Permiten realizar **validaciones antes o después** de que se ejecute el código asociado a una ruta.
- Los filtros se definen como una clase PHP almacenada dentro de la carpeta "`app/Http/Middleware`".
- Cada *middleware* aplicará un filtro concreto sobre una petición y podrá permitir su ejecución, dar un error o redireccionar.
- Laravel incluye por defecto algunos filtros "`auth`", "`guest`", "`csrf`", ...
- Para crear nuestros propios filtros podemos usar el comando de Artisan:

```
php artisan make:middleware MyMiddleware
```

# *MIDDLEWARE* O FILTROS

- Ejemplo de código de un *middleware* propio:

```php
<?php
namespace App\Http\Middleware;
use Closure;

class MyMiddleware {
    public function handle($request, Closure $next) {
        return $next($request);
    }
}
```

Parámetros de entrada

Método que tiene que procesar la petición

- Podemos devolver:
  - Para que continúe la petición: `return $next($request);`
  - Redirección a otra ruta: `return redirect('home');`
  - Lanzar excepción o abortar: `abort(403, 'Unauthorized');`

# *USO DE MIDDLEWARE* O FILTROS

- Para poder utilizar un *middleware* propio tendremos que añadirlo al fichero "`app/Http/Kernel.php`".
- Esta clase define tres arrays:
  - `$middleware`: Si lo añadimos a este array se ejecutará en TODAS las peticiones.
  - `$middlewareGroups`: Para que se ejecute solo dentro de un grupo de rutas (`web` o `api`).
  - `$routeMiddleware`: Si lo añadimos aquí lo podremos usar de forma separada para las rutas que indiquemos. Ejemplo:

```
protected $routeMiddleware = [
    'auth' => \App\Http\Middleware\Authenticate::class,
    'auth.basic' => \Illuminate\Auth\Middleware\AuthBasic::class,
    'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
    'old' => \App\Http\Middleware\MyMiddleware::class,
];
```

# *USO DE MIDDLEWARE* O FILTROS

- Una vez añadido a `$routeMiddleware` podemos proteger las rutas individualmente.
- Para asociar un filtro a una ruta tenemos que modificar el fichero de rutas "`routes/web.php`":

```
Route::get('user', function()
{
    return 'Has superado el filtro!';
})->middleware('old');

// Podemos indicar varios filtros...
Route::get('user', 'UserController@showProfile')
  ->middleware('auth', 'old');
```

# FILTRAR GRUPOS DE RUTAS

- También podemos agrupar varias rutas y aplicar un filtro a todas ellas a la vez:

```
Route::group(['middleware' => 'auth'], function()
{
    Route::get('/', function()
    {
        //
    });

    Route::get('user/profile', function()
    {
        //
    });
});
```

# GRUPOS DE RUTAS CON PREFIJO

- También podemos usar los grupos de rutas para aplicar un prefijo a todas ellas, por ejemplo:

```
Route::group(['prefix' => 'api'], function()
{
    Route::group(['prefix' => 'v1'], function()
    {
        Route::get('recurso',   'ControllerAPIv1@index');
        Route::post('recurso',  'ControllerAPIv1@store');
        Route::get('recurso/{id}','ControllerAPIv1@show');
    });
    Route::group(['prefix' => 'v2'], function()
    {
        Route::get('recurso',   'ControllerAPIv2@index');
        Route::post('recurso',  'ControllerAPIv2@store');
        Route::get('recurso/{id}','ControllerAPIv2@show');
    });
});
```

# REDIRECCIONES

Podemos redirigir de ruta a otra si por ejemplo hay un error de validación o de permisos, volver a la ruta anterior o incluso devolver los valores de la petición con la redirección:

```
return redirect('user/login');
return back();       // Volver a la ruta anterior

// O redirigir a un método de un controlador:
return redirect()->action('HomeController@index');

// Para añadir parámetros:
return redirect()->action('UserController@profile', [1]);

// Para devolver los valores de entrada del usuario:
return redirect('form')->withInput();
return back()->withInput();

// O para reenviar los datos de entrada excepto algunos:
return redirect('form')->withInput($request->except('password'));
```

# FORMULARIOS

# FORMULARIOS

- Laravel 5 NO incluye métodos para la generación de formularios.
- Solo incluye algunas funciones de ayuda.
- En una vista, para abrir y cerrar formularios, haríamos:

```
<form action="{{ url('foo/bar') }}" method="POST">
    ...
</form>
```

- Para cambiar el **método** de envío (por defecto HTML solo acepta GET y POST):

```
<form action="{{ url('foo/bar') }}" method="POST">
    @method('PUT')
    ...
</form>
```

# FORMULARIOS: PROTECCIÓN CONTRA CSRF

- Protección contra CSRF (*Cross-site request forgery* o falsificación de petición en sitios cruzados):
  - Tipo de *exploit* malicioso de un sitio web en el que comandos no autorizados son transmitidos por un usuario en el cual el sitio web confía.
- Laravel proporciona una forma fácil de protegernos:

```
<form action="{{action('Controller@method')}}"
                                        method="POST">
    @csrf
    @method('PUT')
    ...
</form>
```

# FORMULARIOS: CAMPOS DE TEXTO

- Campos de texto:

```
<input type="text" name="nombre" id="nombre">
{{--También podemos especificar un valor por defecto--}}
<input type="text" name="edad" id="edad" value="18">
```

- *Textarea, password, hidden*:

```
<input type="password" name="password" id="password">
<input type="hidden" name="oculto" value="valor">
<textarea name="texto" id="texto" rows="4" cols="50">
    Texto por defecto
</textarea>
```

- También podemos crear otro tipo de *inputs* (email, number, tel, etc.).

# FORMULARIOS: ETIQUETAS Y BOTONES

- Añadir las etiquetas de un formulario:

```
<label for="correo">Correo electrónico:</label>

<input type="email" name="correo" id="correo">
```

  - Es importante que el valor del atributo "`for`" coincida con el identificador (id) del campo asociado.

- Podemos añadir tres tipos de **botones** a un formulario:

```
<button type="submit">Enviar</button>

<button type="reset">Borrar</button>

<button type="button">Volver</button>
```

# FORMULARIOS: *CHECKBOX Y RADIO BUTTONS*

- *Checkbox*:

```
<label for="terms">Aceptar términos</label>
<input type="checkbox" name="terms" id="terms" value="1">
```

- *Radio buttons*:

```
<label for="color">Elige un color:</label>

<input type="radio" name="color" id="color" value="red">Rojo
<input type="radio" name="color" id="color" value="blue">Azul
<input type="radio" name="color" id="color" value="green">Verde
```

- Es importante que todos tengan el mismo nombre en la propiedad "`name`".
- Además podemos marcar alguno por defecto añadiendo "`checked`"

# FORMULARIOS: LISTAS DESPLEGABLES

- Listas desplegables tipo "*select*":

```
<select name="marca">
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="mercedes">Mercedes</option>
  <option value="audi" selected>Audi</option>
</select>
```

- Podemos marcar una opción por defecto añadiendo el atributo "`selected`".

# ¿PREGUNTAS?