

# Laravel: Requests y Responses

Peticiones y Respuestas + cookies

# Request - Sollicitudes

# Request - Solicitudes

La clase ***Illuminate\Http\Request*** de Laravel proporciona una forma orientada a objetos de **interactuar con la solicitud HTTP** actual que está siendo procesada por la aplicación, así como poder recuperar los **datos introducidos** por el usuario, **cookies** y **archivos** enviados con la solicitud.

Para poder acceder a la instancia de la petición actual, se debe hacer uso de esta clase Request al procesar rutas o llamar a métodos de de controladores:

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



# Request - Ejemplo acceso vía ruta

El siguiente ejemplo define una ruta en *web.php*:

```
<?php
```

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

```
Route::get('/', function (Request $request) {
```

```
    // procesado de Request según se requiera
```

```
    return view('welcome');    // carga de la vista que se requiera
```

```
});
```



# Request - Ejemplo acceso vía controlador

El siguiente ejemplo define un método `store()` en un controlador `UserController.php`:


```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;

class UserController extends Controller {

    public function store(Request $request): RedirectResponse {

        $name = $request->input('name');    // Extracción de datos procedentes de la petición del usuario

        // Código para guardar el usuario
        return redirect('/users');    // redirección tras finalizar el guardado
    }
}
```



# Request - Parámetros de ruta


Si el método de un controlador también espera recibir un parámetro de una ruta (ej. ID de un registro), se deben listar los parámetros de ruta después de las demás dependencias.

Por ejemplo, si la ruta está definida de la siguiente manera:

```
use App\Http\Controllers\UserController;
```

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

Desde el controlador, se puede usar la clase Request y acceder al valor del parámetro {id} definido en la ruta entrante



# Request - Ejemplo acceso a parámetro de ruta

El siguiente ejemplo define un método *update()* en un controlador *UserController.php*:

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;

class UserController extends Controller {

    public function store(Request $request, string $id): RedirectResponse {

        $name = $request->input('name');    // Extracción de datos procedentes de la petición del usuario

        $thisUser = User::findOrFail($id);    // Recuperación de instancia representada por id facilitado en /user/{id}

        // Código para comparar valores y actualizar el usuario
        return redirect('/users'); // redirección tras finalizar la actualización de datos
    }
}
```

# Request - métodos propios - Path

El método ***path()*** devuelve la **información de la ruta** de la solicitud/petición.

Por lo tanto, si la solicitud entrante está dirigida a **http://example.com/foo/bar**, el método `path()` devolverá **foo/bar**.

Ejemplo de uso:

```
$uri = $request->path();
```





# Request - métodos propios - is y routels

El método ***is()*** permite **verificar si la ruta de la solicitud entrante coincide con un patrón dado**. Se puede usar el carácter **\*** como un comodín al utilizar este método.

El siguiente ejemplo será *true* y accede al *if* si la petición entrante viene de una ruta */admin/[lo\_que\_sea]*:

```
if ($request->is('admin/*')) { // ... }
```

El método ***routels()*** permite **verificar si la ruta de la solicitud entrante coincide con un nombre de ruta dado**. Se puede usar el carácter **\*** como un comodín al utilizar este método.

El siguiente ejemplo será *true* y accede al *if* si la petición entrante viene de una ruta con *->name('admin')*:

```
if ($request->routels('admin.*')) { // ... }
```



# Request - métodos propios - URL

Para obtener la **URL completa de la solicitud entrante**, se pueden usar los métodos *url()* o *fullUrl()*.

El método *url()* devolverá la **URL sin la cadena de consulta**, mientras que el método *fullUrl()* **incluirá la cadena de consulta**.

Ejemplos:

```
$url = $request->url();
```

```
$urlWithQueryString = $request->fullUrl();
```



## Request - métodos propios - URL - añadir datos a la cadena de consulta

Con el método ***fullUrlWithQuery()*** es posible obtener una URL que combine la cadena de consulta de la petición con un array de datos.

Ejemplo:

```
$request->fullUrlWithQuery(['type' => 'phone']);
```



# Request - métodos propios - method e isMethod

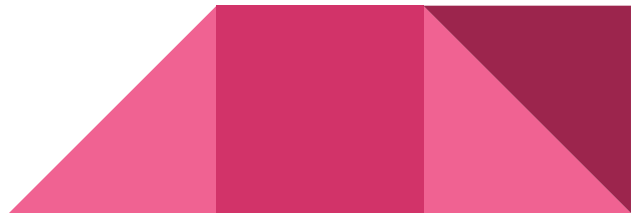
El método ***method()*** devolverá el tipo de la solicitud HTTP realizada (ej. Get, Post, Put...).

Por otro lado, se puede usar el método ***isMethod()*** para verificar que el tipo de solicitud HTTP coincide con una cadena dada.

Ejemplos:

```
$method = $request->method(); // devuelve el tipo de la solicitud
```

```
if ($request->isMethod('post')) {  
    // Accede al if si la petición es tipo POST  
}
```



# Request - Headers

Se puede **obtener un encabezado de la solicitud** desde la instancia de *Illuminate\Http\Request* utilizando el método ***header()***.

Si el encabezado no está presente en la solicitud, se devolverá *null*.

Sin embargo, el método `header()` acepta un segundo argumento opcional que será devuelto en caso de que el encabezado no esté presente en la solicitud.

También se puede verificar si la solicitud contiene un determinado encabezado, con el método ***hasHeader()*** e indicando la cabecera a buscar.

Ejemplo:

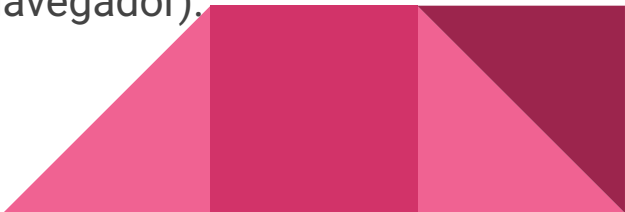
```
$value = $request->header('X-Header-Name');
```



# Encabezados HTTP [RECORDATORIO]

Los encabezados ayudan a personalizar, autenticar y estructurar la **comunicación entre cliente y servidor**.

Transmiten detalles importantes como el tipo de contenido enviado, el formato esperado de la respuesta, credenciales de autenticación, información sobre la conexión, entre otros. Por ejemplo:

- **Content-Type:** Indica el tipo de contenido (por ejemplo; JSON, HTML).
  - **Authorization:** Incluye credenciales para autenticar al cliente.
  - **User-Agent:** Identifica el software que realiza la solicitud (navegador).
  - **Accept:** Define los formatos que el cliente puede procesar.
- 

# Encabezados HTTP [RECORDATORIO]

**method**

**path**

**protocol**

GET /tutorials/other/top-20-mysql-best-practices/ HTTP/1.1

```
Host: net.tutsplus.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: PHPSESSID=r2t5uvjq435r4q7ib3vtdjq120
Pragma: no-cache
Cache-Control: no-cache
```

**HTTP headers as Name: Value**

# Request - Dirección IP

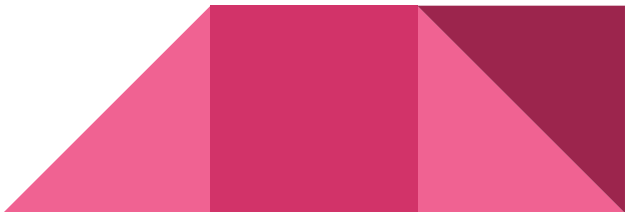
El método *ip()* puede ser utilizado para **obtener la dirección IP del cliente** que realizó la solicitud a la aplicación.

También se puede usar *ips()* para obtener un array de IPs que incluye todas las direcciones IP del cliente que fueron reenviadas por proxies.

En general, las direcciones IP **deben considerarse como datos no confiables, controlados por el usuario**, y solo utilizarse con fines informativos.

Ejemplo:

```
$ipAddresses = $request->ips();
```





# Request - Input - Recuperar datos del cliente/usuario

Se pueden **recuperar todos los datos** pasados con la solicitud entrante como un array utilizando el método ***all()***.

Ejemplo:

```
$input = $request->all();
```

Alternativamente, también se puede llamar al método ***input()*** sin argumentos para **obtener todos los valores de entrada como un array asociativo**.

Ejemplo:

```
$input = $request->input();
```



# Request - Input - Recuperar datos del cliente/usuario

Para recuperar UN valor de entrada específico, se puede acceder a los datos introducidos por el usuario desde la instancia de Illuminate\Http\Request sin preocuparse por qué tipo de solicitud HTTP se utilizó. Esto se realiza mediante el método ***input()*** y **el nombre del campo buscado**. También sirve para JSON.

Ejemplo:

```
$name = $request->input('name');
```

Se puede pasar un valor predeterminado como segundo argumento: dicho valor será devuelto si el valor buscado sobre la solicitud no está presente o es null.

Ejemplo:

```
$name = $request->input('name', 'Nombre_por_defecto');
```



# Request - Collection - Recuperar datos del cliente/usuario

Usando el método **collect()**, también se puede obtener todos los datos de entrada de la solicitud entrante como una colección de datos de Laravel (clase propia *Collection*).

Ejemplo:

```
use Illuminate\Support\Collection;
```

```
$input = $request->collect();
```

```
$input->each(function ($item) {
```

```
    echo $item . '<br>';
```

```
});
```



# Request - Query - Recuperar datos del cliente/usuario

Es posible recuperar datos de la cadena de la consulta mediante el método *query()*.

Ejemplo:

```
$name = $request->query('name');
```

También se puede llamar al método *query()* sin argumentos para **obtener todos los valores de la cadena de consulta como un array asociativo**.



# Request - Recuperar datos del cliente/usuario por tipo

Si los valores entrantes son de un tipo específico, se pueden recoger como tal:

- **string**

```
$name = $request->string('name')
```

- **integer**

```
$numPage = $request->integer('num_page');
```

- **boolean**

```
$valid = $request->boolean(valid);
```

- **date** (segundo parametro define formato, tercero la zona horaria)

```
$elapsed = $request->date('elapsed', '!H:i', 'Europe/Madrid');
```



# Request - Condicionales según los datos de la solicitud

- Método ***has()***: **Determina si un valor está presente** en la solicitud. Ejemplo:

```
if ($request->has('name')) { // ... }
```

- ❑ **Con un array, determina si todos los valores especificados están presentes:**

```
if ($request->has(['name', 'email'])) { // ... }
```



# Request - Condicionales según los datos de la solicitud

- Método ***hasAny()***: Devuelve **true** si alguno de los valores especificados está **presente** en la solicitud.

```
if ($request->hasAny(['name', 'email'])) { // ... }
```

- Método ***whenHas()***: Ejecuta un cierre (closure) si un valor está presente. Puede verse como un `has()` en un *if*:

```
$request->whenHas('name', function (string $input) { // ... })
```



# Request - Condicionales según los datos de la solicitud

- Método ***filled()***: Determina si un **valor está presente y no es una cadena vacía**. Equivalente a ***!is\_empty()*** de PHP:

```
if ($request->filled('name')) { // ... }
```

- Método ***isNotFilled()***: Determina si un **valor falta o es una cadena vacía**. Equivalente a ***is\_empty()*** de PHP:

```
if ($request->isNotFilled('name')) { // ... }
```





# Request - Condicionales según los datos de la solicitud

- Método ***anyFilled()***: Devuelve **true** si alguno de los valores especificados no es una cadena vacía.

```
if ($request->anyFilled(['name', 'email'])) { // ... }
```

- Método ***whenFilled()***: Ejecuta un cierre (closure, entra en la función declarada) si un valor está presente y no es una cadena vacía:

```
$request->whenFilled('name', function (string $input) { // ... });
```

- Métodos ***missing()*** y ***whenMissing()***: Para determinar si una clave dada esta ausente dentro del Requesta.

```
if ($request->missing('name')) { // ... }
```



# Request - Cookies

Todas las **cookies creadas por el framework Laravel** están **cifradas y firmadas con un código de autenticación**, lo que significa que serán consideradas inválidas si han sido modificadas por el cliente.

Para obtener el valor de una cookie de la solicitud, se utiliza el método `cookie` en una instancia de *Illuminate\Http\Request*:

```
$value = $request->cookie('name');
```

NOTA: En Laravel las cookies se crean utilizando el método `cookie()` de la clase *Illuminate\Http\Response*.



# Request - Files - Ficheros


Se pueden recuperar archivos subidos desde una instancia de `Illuminate\Http\Request` utilizando el método ***file()***.

El método *file()* **devuelve una instancia de la clase `Illuminate\Http\UploadedFile`**, la cual extiende la clase PHP *SplFileInfo* y proporciona una variedad de métodos para interactuar con el archivo.

Ejemplo de recuperación de un fichero:

```
$file = $request->file('photo');
```

La clase `UploadedFile` incluye métodos para guardar el fichero; *store()* y *storeAs()*.



# Response - Respuestas

# Respuesta básica a Solicitud

Todas las **rutas y controladores deben devolver una respuesta** que será enviada **al navegador** del cliente.

Laravel ofrece varias formas diferentes de devolver respuestas.

La respuesta más básica consiste en devolver una cadena de caracteres (string) desde una ruta o un controlador.

El framework convertirá automáticamente este string en una respuesta HTTP completa. Por ejemplo, desde web.php:

```
Route::get('/', function () {  
    return 'Hola Mundo';  
});
```




# Response - Instancia de Respuesta a Solicitud

Sin embargo, tras procesar una solicitud, una aplicación no devolverá solo un string o array; devolverá vistas (*views*) o instancias de ***Illuminate\Http\Response***.

Devolver una instancia completa de ***Response*** permite personalizar el código de estado HTTP y los encabezados de la respuesta.

Por ejemplo, en *web.php* se puede definir una ruta con la siguiente respuesta:

```
Route::get('/home', function () {  
    return response('Hello World', 200)  
        ->header('Content-Type', 'text/plain');  
});
```




# Response - Instancia de Respuesta a Solicitud - header

En el ejemplo anterior, mediante el método ***header()*** se puede modificar los datos de la cabecera de la respuesta.

Este método **se puede encadenar** para aplicar múltiples configuraciones sobre el encabezado:

```
return response($content)
    ->header('Content-Type', $type)
    ->header('X-Header-One', 'Header Value')
    ->header('X-Header-Two', 'Header Value');
```



# Encabezados HTTP [RECORDATORIO 2]

En una **respuesta**, los encabezados pueden incluir el **tipo de contenido**, el **código de estado HTTP**, o la **fecha de expiración de los datos**.

**protocol**      **status code**

```
HTTP/1.x 200 OK
Transfer-Encoding: chunked
Date: Sat, 28 Nov 2009 04:36:25 GMT
Server: LiteSpeed
Connection: close
X-Powered-By: W3 Total Cache/0.8
Pragma: public
Expires: Sat, 28 Nov 2009 05:36:25 GMT
Etag: "pub1259380237;gz"
Cache-Control: max-age=3600, public
Content-Type: text/html; charset=UTF-8
Last-Modified: Sat, 28 Nov 2009 03:50:37 GMT
X-Pingback: http://net.tutsplus.com/xmlrpc.php
Content-Encoding: gzip
Vary: Accept-Encoding, Cookie, User-Agent
```

**HTTP headers as Name: Value**

## HTTP Status Codes

- 200s solicitud satisfactoria.
- 300s redirecciones.
- 400s problema con la solicitud.
- 500s problema co el servidor.



# Response - Ficheros (visualización o descarga)

El método ***file()*** permite **mostrar un fichero directamente en navegador, sin realizar la descarga**. Esto es útil para ficheros como imágenes o PDFs. Se debe pasar la **ruta al fichero a descargar**. Sigue el formato:

```
return response()->file($pathToFile);
```

Por otro lado, el método ***download()*** genera una respuesta que **fuerza al navegador del cliente la descarga de un fichero**. Se debe pasar la **ruta al fichero a descargar**. Este método acepta un nombre de fichero como segundo parámetro, para renombrar el fichero a descargar. Sigue el formato:

```
return response()->download($pathToFile);
```



# Response - Cookies

Se pueden pasar **cookies** de Laravel con las instancias de tipo *Response*. Se debe indicar su **nombre**, un **valor** y los **minutos** de validez. Estas cookies están encriptadas; no pueden ser modificadas por el cliente.

*return response('Hello World')->cookie( 'name', 'value', \$minutes);*

Tambien se pueden añadir argumentos propios de la funcion de PHP `setcookie()`.



# Cookies de Laravel

Para crear una cookie de Laravel, se puede usar la función de ayuda “helper” **`cookie()`**. Se deben indicar los argumentos de nombre, valor y minutos de validez.

Esta cookie **NO se envía de vuelta al cliente** automáticamente; se debe pasar vía instancia de Response con el método **`->cookie()`** visto previamente.

*// Crear cookie*

```
$cookie = cookie('name', 'value', $minutes);
```

*// Enviar cookie*

```
return response('Hello World')->cookie($cookie);
```



# Cookies de Laravel

Para **eliminar** una cookie de Laravel, se debe hacer lo mismo que se hace con cookies de PHP (Creadas con `setcookie()`): se debe **forzar su expiración** o validez.

Con Laravel, este paso se puede realizar con la instancia de Respuesta del usuario, mediante el método ***withoutCookie()*** e indicando el nombre:

```
return response('Hello World')->withoutCookie('name');
```

Alternativamente, se puede forzar la expiración directamente, mediante el método ***expire()*** de la clase propia `Cookie`, e indicando el nombre:

```
Cookie::expire('name');
```




# RedirectResponse

Las **respuestas de redirección** son instancias de la clase ***Illuminate\Http\RedirectResponse*** y contienen los **encabezados necesarios para redirigir al cliente a otra URL**.

Existen varias formas de generar una instancia de *RedirectResponse*. El método más simple es utilizar el asistente global (helper) ***redirect()***.

Un ejemplo de ruta redireccionada que podría declararse en web.php:

```
Route::get('/dashboard', function () {  
    return redirect('/home/dashboard');  
});
```



# RedirectResponse - redirigir a rutas con nombre

Cuando se llama al helper **redirect()** **sin parámetros**, se devuelve una instancia de *Illuminate\Routing\Redirector* (redirector de rutas), lo que permite invocar cualquier método en dicha instancia.

Esto significa que, para generar una redirección, **RedirectResponse**, a una ruta con nombre, se puede utilizar el método **route()**.

Por ejemplo:

```
return redirect()->route('login');
```

Si la ruta contiene parámetros, estos se pueden pasar como segundo argumento del método:

```
// Para una ruta con la siguiente URI: /profile/{id} donde id = 1
```

```
return redirect()->route('profile', ['id' => 1]); // ruta: profile/1
```



# RedirectResponse - redirigir a método de controlador

También es posible redirigir a una acción o método de un controlador, para ello, se emplea ***redirect()*** sin argumentos y el método ***action()*** con llamada a la clase del controlador y su método.


Ejemplo usando el controlador *UserController.php* en una redirección (en *web.php*, en un controlador, ...):

```
use App\Http\Controllers\UserController;
```

```
return redirect()->action([UserController::class, 'index']);
```

Si el método requiere parámetros, se deben pasar como segundo parámetro de *action()*:

```
return redirect()->action(  
    [UserController::class, 'profile'], ['id' => 1]  
);
```




# RedirectResponse - redirigir con información adicional

Redirigir a una nueva ruta y agregar datos para la nueva vista a cargar suele hacerse al mismo tiempo.

Esto es común después de realizar una acción con éxito, como por ejemplo tras un store o update (guardado en BD) finalizado correctamente.

Para realizar este paso de datos, es posible crear una instancia de *RedirectResponse* y agregar dicha información a la sesión de Laravel mediante el método encadenado ***with()***. Por ejemplo:

```
Route::post('/user/profile', function () {  
    // Acción a realizar...  
  
    return redirect('/dashboard')->with('status', '¡Perfil actualizado!');  
});
```





# RedirectResponse - redirigir con información adicional

Una vez **pasados datos adicionales vía sesión dentro de una redirección**, es posible capturarlos haciendo referencia al **nombre del datos con session() utilizando sintaxis de Blade**.

Por ejemplo, continuando con la vista a cargar tras guardar datos en BD se puede recoger el estado 'status' pasado mediante *with()* con el siguiente bucle:

```
@if (session('status'))
```

```
    <div class="alert alert-success">
```

```
        {{ session('status') }}    {{-- Esto imprime "¡Perfil actualizado! --}}
```

```
    </div>
```

```
@endif
```



# RedirectResponse - redirigir fuera de la app web

A veces es necesario redirigir a un dominio fuera de la aplicación.

Esto se puede lograr utilizando el método ***away()***, que crea una *RedirectResponse* sin aplicar codificación de URL adicional, validación o verificación.

El siguiente ejemplo manda al famoso buscador google:

```
return redirect()->away('https://www.google.com');
```

