

# **Laravel - Middleware**

**Topics**: Laravel

Written on December 21, 2023

In Laravel, middleware acts as a filter for HTTP requests entering your application. It can perform various tasks such as authentication, logging, modifying request or response data, and more. Middleware is executed in a sequential manner, and you can apply it globally to all routes, to specific routes, or within a controller.

Here's a basic overview of how middleware works in Laravel:

# **Creating Middleware:**

You can create a middleware using the artisan command:

## php artisan make:middleware MyMiddleware

This will create a new middleware class in the app/Http/Middleware directory.

#### **Middleware Structure:**

A middleware class in Laravel typically contains a handle method. This method is called for each incoming HTTP request. Here's a simple example:

```
namespace App\Http\Middleware;
use Closure;
class MyMiddleware
{
   public function handle($request, Closure $next)
   {
      // Perform actions before the request is handled by the application.
      $response = $next($request);
      // Perform actions after the request is handled by the application.
      return $response;
   }
}
```

# **Registering Middleware:**

You can register middleware in the app/Http/Kernel.php file. The \$middleware property contains a list of middleware that will be run on every request:

```
protected $middleware = [
    // ...
    \App\Http\Middleware\MyMiddleware::class,
];

You can also apply middleware to specific routes in the web.php or api.php route files:

Route::get('/example', function () {
    // Your route logic here
})->middleware('my_middleware');
```

## **Middleware Parameters:**

You can pass parameters to middleware if needed. Modify the handle method to accept additional parameters:

```
public function handle($request, Closure $next, $parameter)
{
    // Access the parameter here.
    // ...
}
And in the web.php or api.php file, you can pass parameters like this:
Route::get('/example', function () {
    // Your route logic here
})->middleware('my middleware:param value');
```

#### **Global Middleware:**

Global middleware is run on every HTTP request to your application. You can add them to the \$middleware property in the Kernel class.

## **Terminable Middleware:**

If you need to perform actions after the response has been sent to the browser, you can implement the TerminableMiddleware interface and add the terminate method to your middleware.

```
public function terminate($request, $response)
{
    // Perform actions after the response is sent.
}
```

