
## What is Middleware

I like to think of middleware as an forced interruption in our code. So in all languages we have our inputs and outputs. When we hit a url at the browser we have an input that is expecting an output. What if we wanted certain criteria for the url that was hit to be met and if not met then force some user behavior to happen.

Think of something simple like a user loggin in. So a user needs to login to view a section of our site. Let's say they visit `http://blog.app/admin` and we don't want to give access to that area unless they are logged in. We would need to interrupt the application and route them to the area for logging in before proceeding to the admin area.

Also think of what if the person logged in is not an admin and only an editor who shouldn't be able to delete posts. We would need to interrupt that route and route them back to the dashboard or something similar.

### Creating middleware

To create our first middleware you would need to do the following.

```
cd ~/Homestead
vagrant ssh
cd  /home/vagrant/projects/blog.app
php artisan make:middleware AgeMiddleware
```

This is the first time we see the php artisan tool. Php artisan is a tool we will be using through the building of our application. Whenever we do `php filename.php` that will run that php file. So `artisan` is just a php file with no extension.

Technically, the `artisan` file is *almost* identical to the `public/index.php` file (which handle the HTTP request). The former passes a console handler to the application instance to interact with the terminal, and the latter passes an HTTP handler to handle the browser HTTP requests.

The above will create an empty middleware class that we need to fill in with the logic we are checking for.

Below is an example from the Laravel documentation of something you would probably do for an age middleware. We are saying that you must be over the age of 200 years to access this part of the site.

```
<?php

namespace App\Http\Middleware;
```

```
use Closure;

class AgeMiddleware
{
    /**
     * Run the request filter.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @return mixed
     */
    public function handle($request, Closure $next)
    {
        if ($request->input('age') <= 200) {
            return redirect('home');
        }

        return $next($request);
    }

}
```

### Giving our middleware a short name

We need to give our route a short name to be able to use it in our routes. We would need to add a line to our `app/Http/Kernel.php` file.

```
protected $routeMiddleware = [
    'auth' => \App\Http\Middleware\Authenticate::class,
    'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
    'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
    'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
    'ageCheck' => \App\Http\Middleware\AgeMiddleware::class,
];
```

Where `ageCheck` is the key will use to reference the middleware we created.

### Adding the middleware to our route

So now that we created the middleware we need to let the application know where we will be adding this to. To take advantage of this interrupt we need to do so in the routes. This is easily done by adjust our route like the following.

```
Route::get('fountain-of-youth', ['middleware' => 'ageCheck', function () {
    //
}]);
```

Now we have added our middleware (interrupt) to our route and when we hit the url `blog.dev/fountain-of-youth` you will not be allowed to that path unless you are over 200 years old.

Of course we would somehow need to collect this age information from the user at some point so we can accurately check if they can access this area.

Our current middleware is called before Laravel handles the request, but you can change that by calling the `$next` pipeline and return the response.

```
public function handle($request, Closure $next)
{
   $response = $next($request);

   // Perform action

   return $response;
}
```