# Route Middleware

## Description

- Middleware allows us to perform some action before or after executing the route.
- For example,
    - If a user needs to access a protected page then you can use the `auth` middleware to check if the user is logged in.
        - If yes then you allow access
        - Else the user is redirected to a login page.

## Example

### Two Basic Middleware Classess: `BeforeMiddleware` and `AfterMiddleware`

- `BeforeMiddleware` will execute before the route
- `AfterMiddleware` will execute after the route.

### Creating

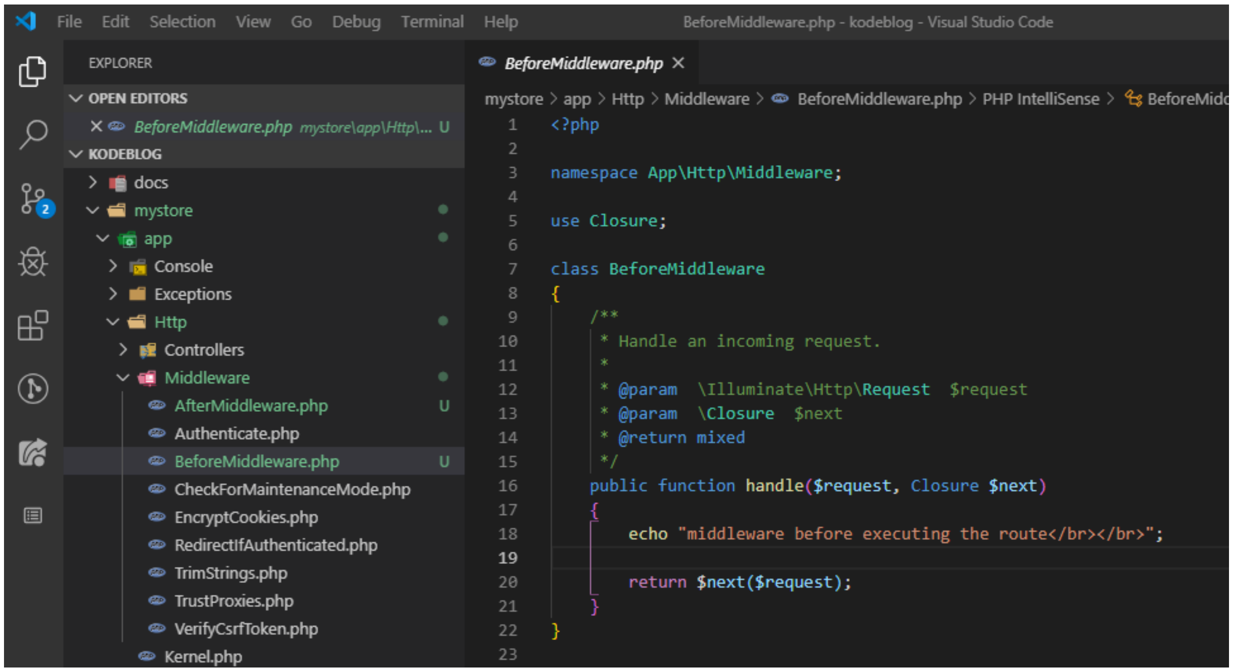
- In `cmder`

  ```
  cd mystore
  php artisan make:middleware BeforeMiddleware
  php artisan make:middleware AfterMiddleware
  ```

### Fill-in Middleware Classes

#### `BeforeMiddleware` class

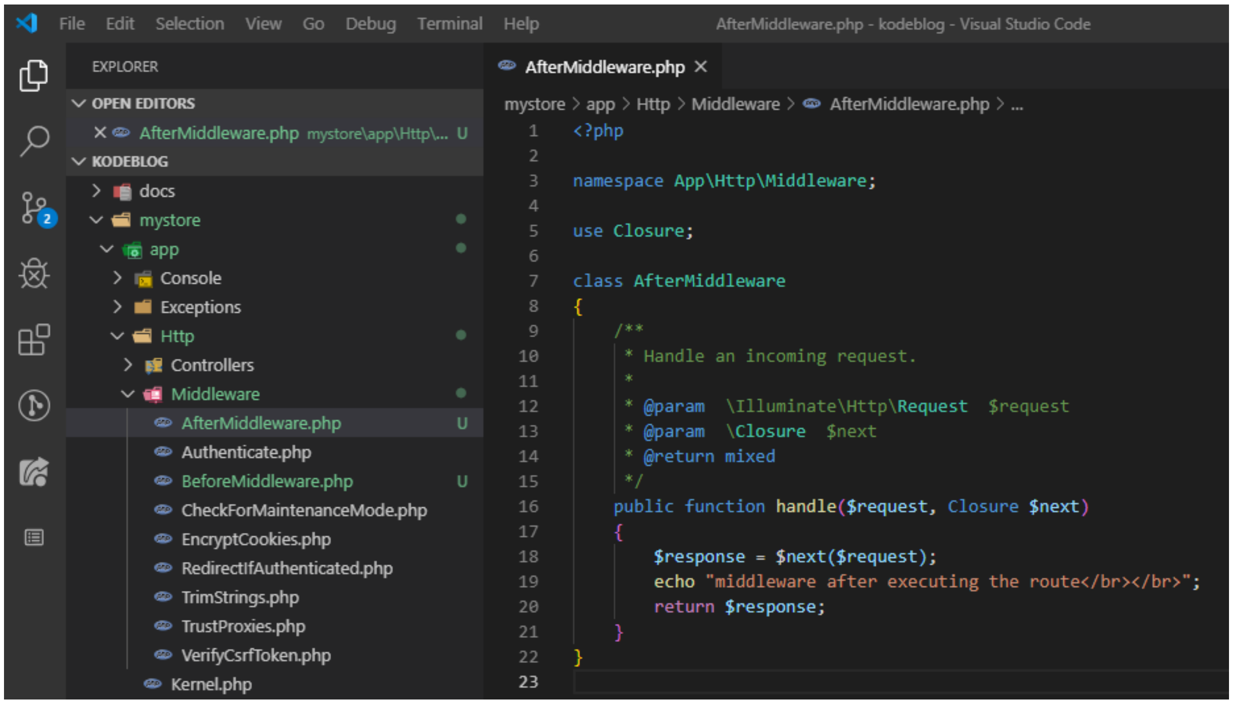
- In VSC, edit file `kodeblog\mystore\app\Http\Middleware\BeforeMiddleware.php`


File Edit Selection View Go Debug Terminal Help BeforeMiddleware.php - kodeblog - Visual Studio Code
EXPLORER
OPEN EDITORS
BeforeMiddleware.php mystore\app\Http\... U
KODEBLOG
docs
mystore
app
Console
Exceptions
Http
Controllers
Middleware
AfterMiddleware.php U
Authenticate.php
BeforeMiddleware.php U
CheckForMaintenanceMode.php
EncryptCookies.php
RedirectIfAuthenticated.php
TrimStrings.php
TrustProxies.php
VerifyCsrfToken.php
Kernel.php
mystore > app > Http > Middleware > BeforeMiddleware.php > PHP IntelliSense > BeforeMidd

```php
<?php

namespace App\Http\Middleware;

use Closure;

class BeforeMiddleware
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @return mixed
     */
    public function handle($request, Closure $next)
    {
        echo "middleware before executing the route</br></br>";

        return $next($request);
    }
}

```

`AfterMiddleware`

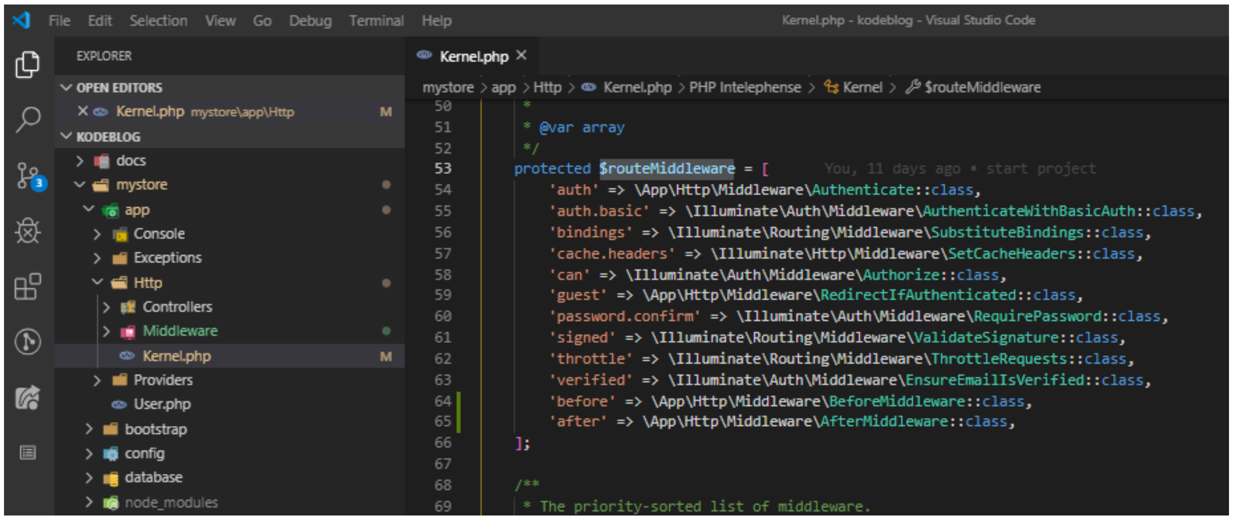
- In VSC, edit file `kodeblog\mystore\app\Http\Middleware\AfterMiddleware.php`


File Edit Selection View Go Debug Terminal Help AfterMiddleware.php - kodeblog - Visual Studio Code
EXPLORER
OPEN EDITORS
AfterMiddleware.php mystore\app\Http\... U
KODEBLOG
docs
mystore
app
Console
Exceptions
Http
Controllers
Middleware
AfterMiddleware.php U
Authenticate.php
BeforeMiddleware.php U
CheckForMaintenanceMode.php
EncryptCookies.php
RedirectIfAuthenticated.php
TrimStrings.php
TrustProxies.php
VerifyCsrfToken.php
Kernel.php
mystore > app > Http > Middleware > AfterMiddleware.php > ...

```php
<?php

namespace App\Http\Middleware;

use Closure;

class AfterMiddleware
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @return mixed
     */
    public function handle($request, Closure $next)
    {
        $response = $next($request);
        echo "middleware after executing the route</br></br>";
        return $response;
    }
}

```

## Register Custom Middleware Classes

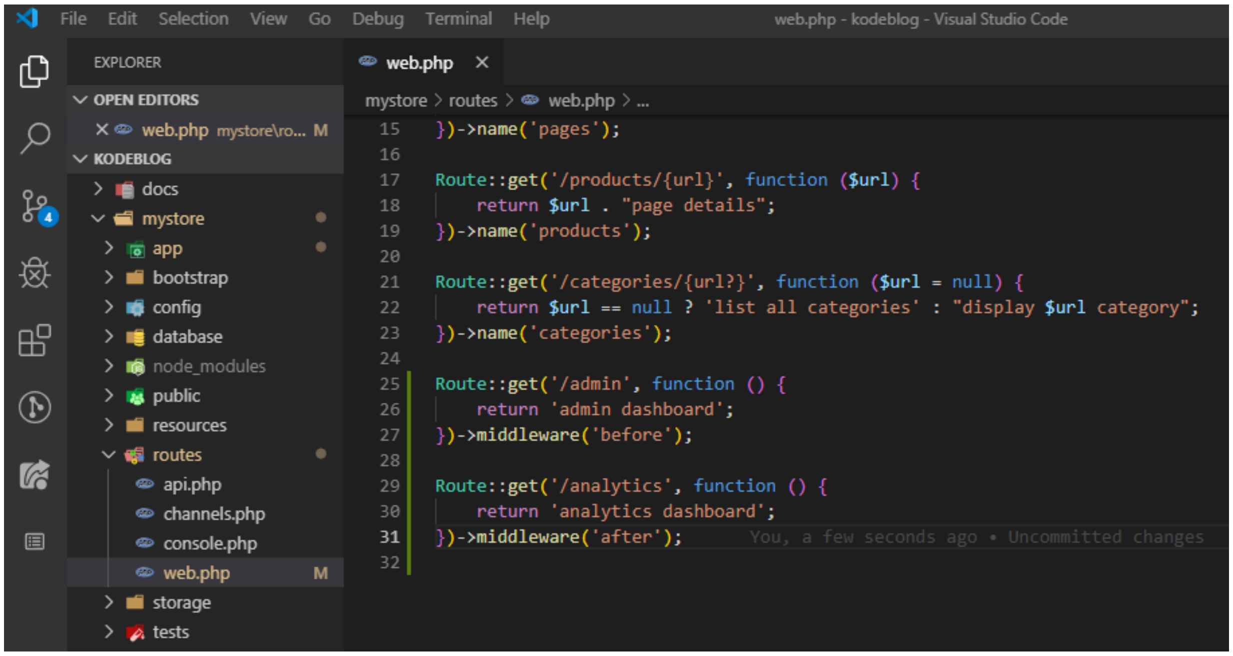
- In VSC, open file `kodeblog\mystore\app\Http\Kernel.php`
- Register `before` and `after` in list `$routeMiddleware`



```php
     *
     * @var array
     */
    protected $routeMiddleware = [        You, 11 days ago • start project
        'auth' => \App\Http\Middleware\Authenticate::class,
        'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
        'bindings' => \Illuminate\Routing\Middleware\SubstituteBindings::class,
        'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,
        'can' => \Illuminate\Auth\Middleware\Authorize::class,
        'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
        'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,
        'signed' => \Illuminate\Routing\Middleware\ValidateSignature::class,
        'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
        'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,
        'before' => \App\Http\Middleware\BeforeMiddleware::class,
        'after' => \App\Http\Middleware\AfterMiddleware::class,
    ];

    /**
     * The priority-sorted list of middleware.
```

## Create Routes for `BeforeMiddleware` and `AfterMiddleware`

- In VSC, open file `kodeblog\mystore\routes\web.php`
- Add two routes

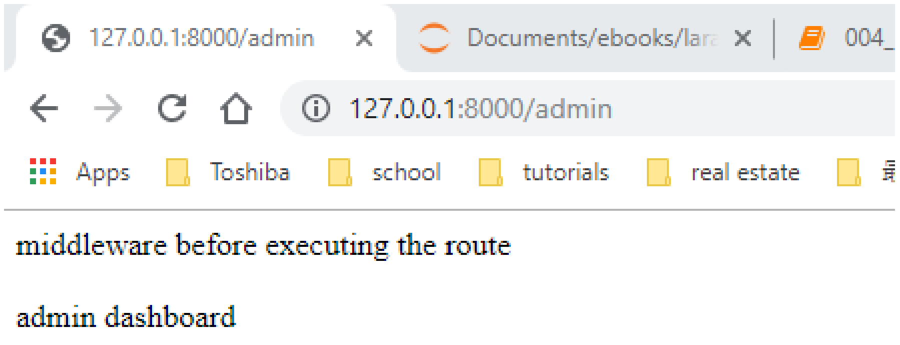


```php
})->name('pages');

Route::get('/products/{url}', function ($url) {
    return $url . "page details";
})->name('products');

Route::get('/categories/{url?}', function ($url = null) {
    return $url == null ? 'list all categories' : "display $url category";
})->name('categories');

Route::get('/admin', function () {
    return 'admin dashboard';
})->middleware('before');

Route::get('/analytics', function () {
    return 'analytics dashboard';
})->middleware('after');        You, a few seconds ago • Uncommitted changes
```

## Testing

### Test `before` Middleware

```
http://127.0.0.1:8000/admin
```

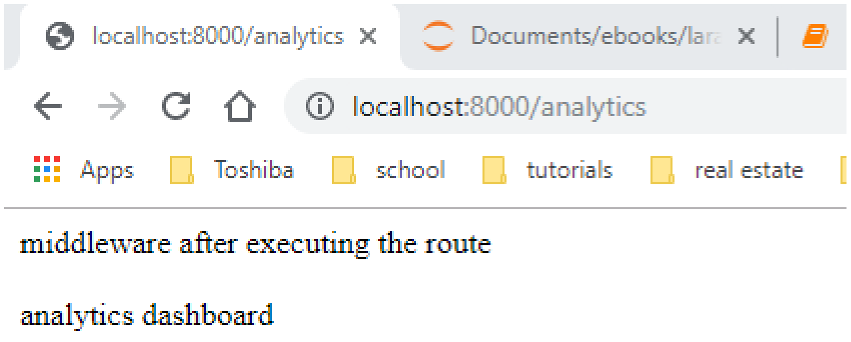


middleware before executing the route

admin dashboard

### Test `after` Middleware

```
http://localhost:8000/analytics
```



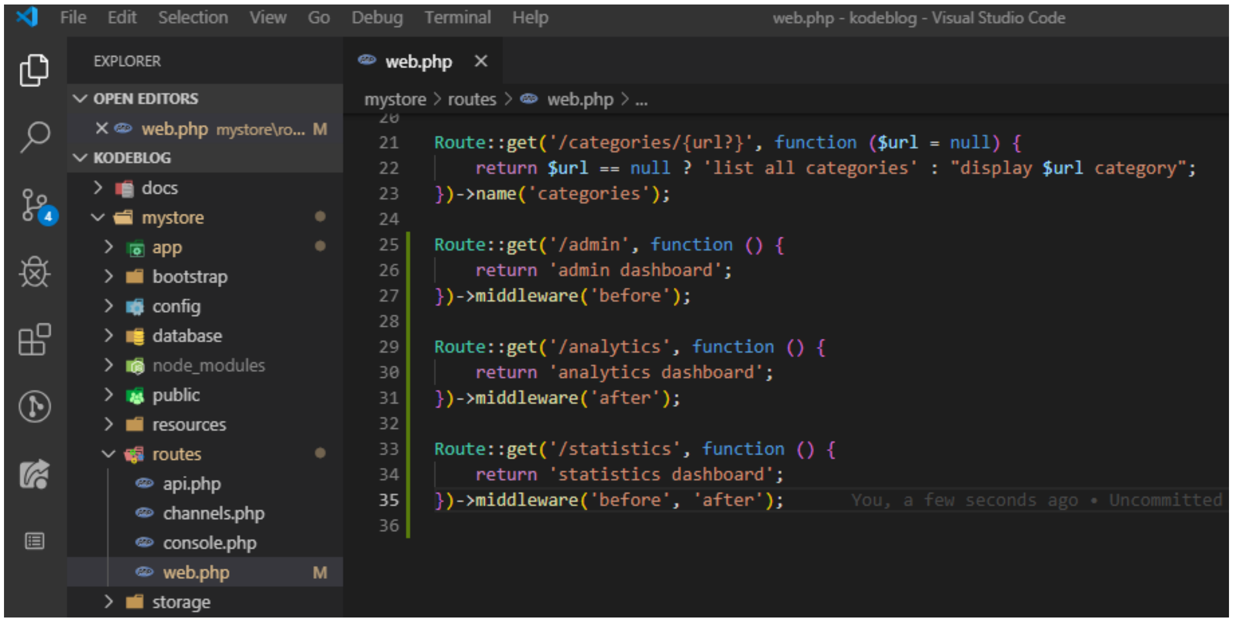
middleware after executing the route

analytics dashboard

## Combied Middleware

### Creating Route

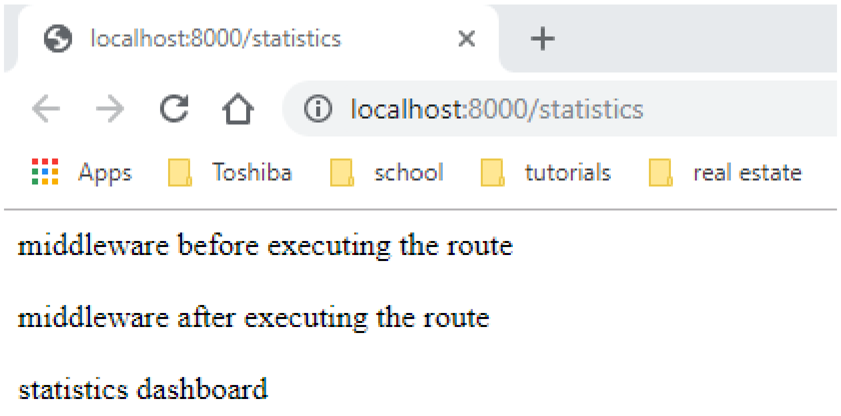
- In VSC, open file `kodeblog\mystore\routes\web.php`
- Add combined middleware route



```php
Route::get('/categories/{url?}', function ($url = null) {
    return $url == null ? 'list all categories' : "display $url category";
})->name('categories');

Route::get('/admin', function () {
    return 'admin dashboard';
})->middleware('before');

Route::get('/analytics', function () {
    return 'analytics dashboard';
})->middleware('after');

Route::get('/statistics', function () {
    return 'statistics dashboard';
})->middleware('before', 'after');
```

### Testing

```
http://localhost:8000/statistics
```



middleware before executing the route

middleware after executing the route

statistics dashboard

## Commit Step

In [ ]: