Laravel – ett fullskaligt MVC-ramverk

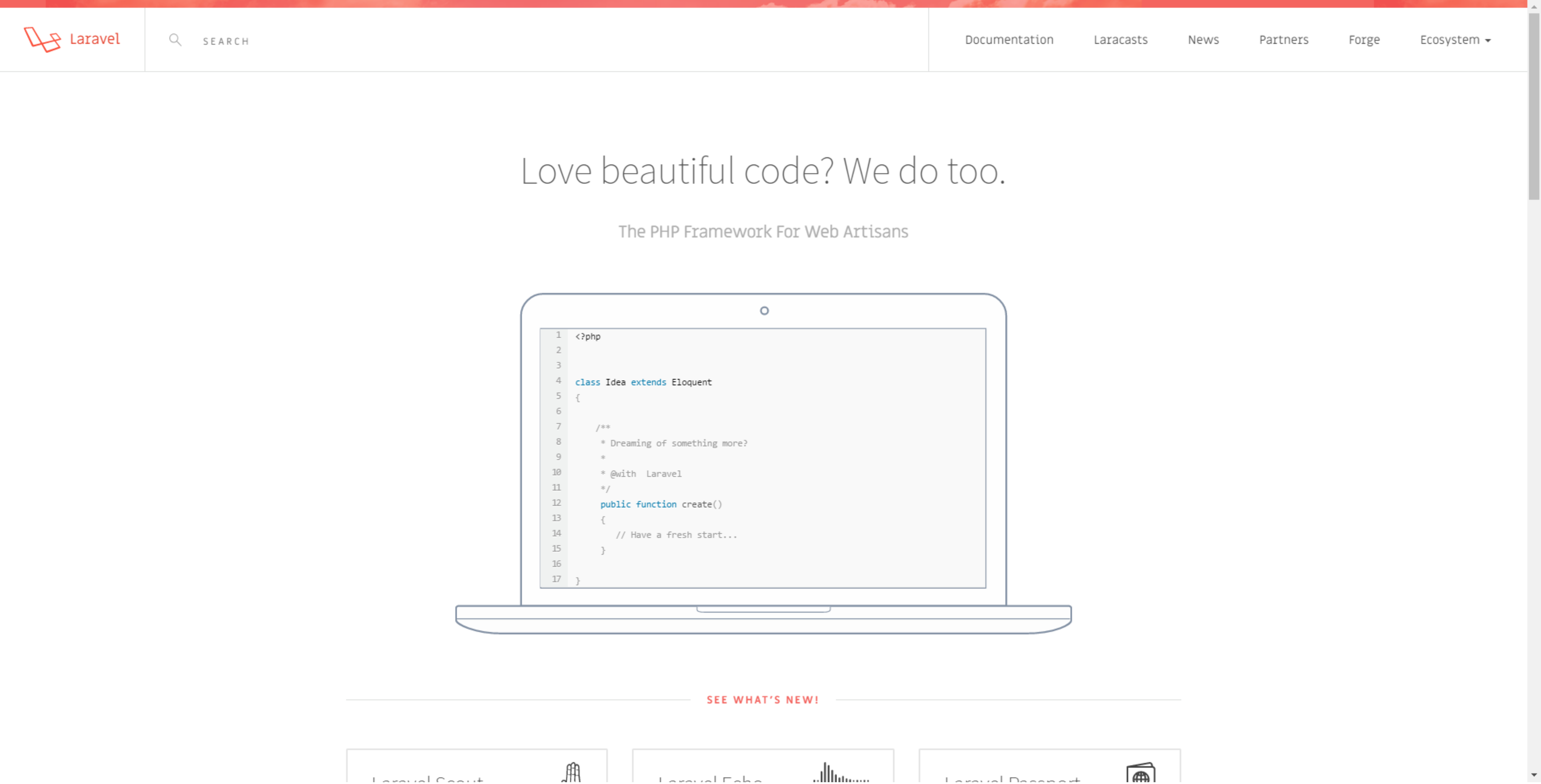
Fortsättning…

Dagens agenda

- Repetition Laravel
- Resurser i Laravel
  - JavaScript
  - CSS
  - Bilder
  - Etc.
- Middleware i Laravel
- Användarhantering i Laravel
  - Användardatabas
  - Inloggning
- Summering av denna del i kursen

Repetetion

Laravel med tillhörande koncept



Love beautiful code? We do too.

The PHP Framework For Web Artisans

```
<?php


class Idea extends Eloquent
{

    /**
     * Dreaming of something more?
     *
     * @with  Laravel
     */
     public function create()
     {
        // Have a fresh start...
     }

}
```

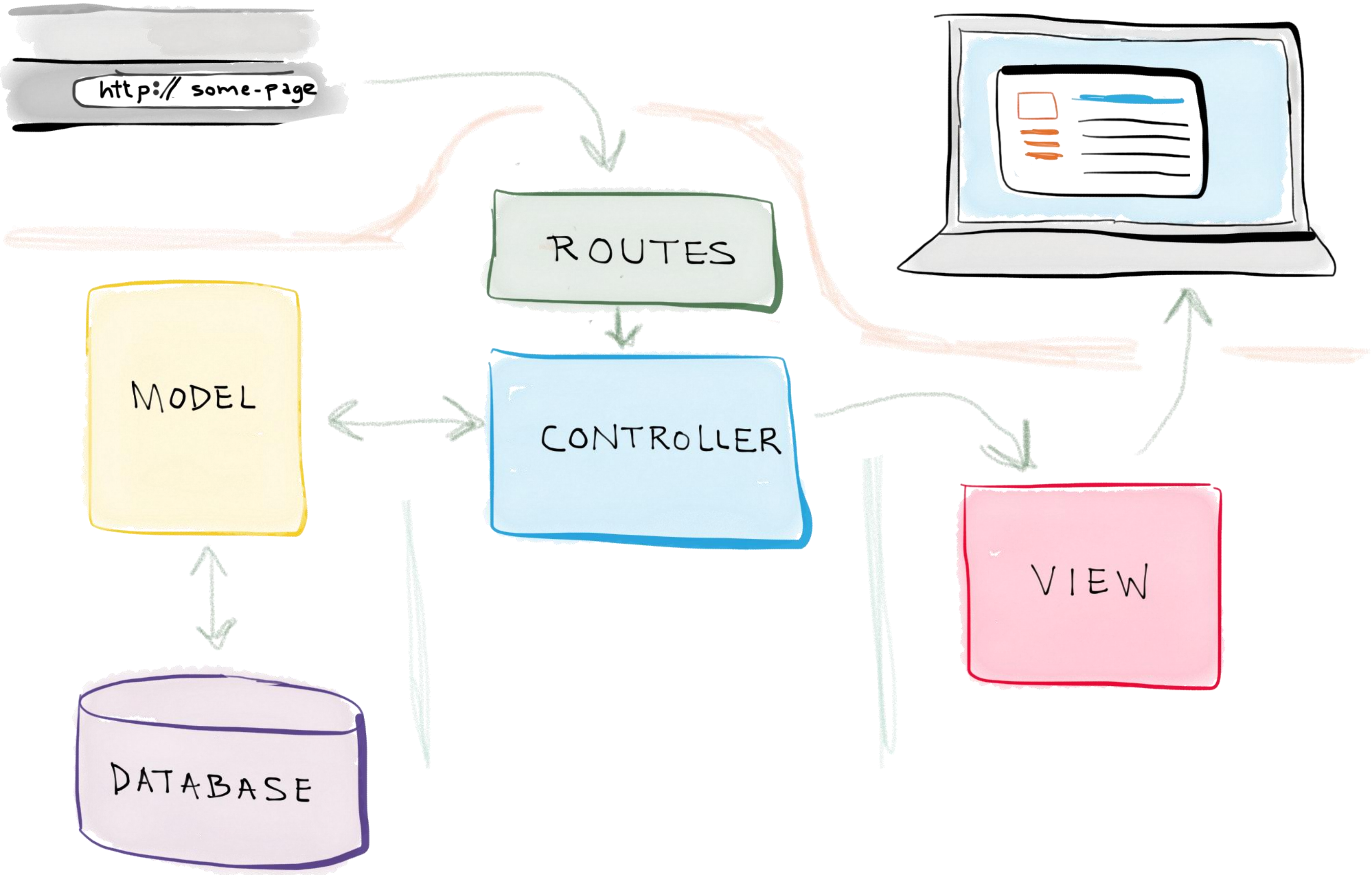
SEE WHAT'S NEW!

Laravel Scout

Laravel Echo

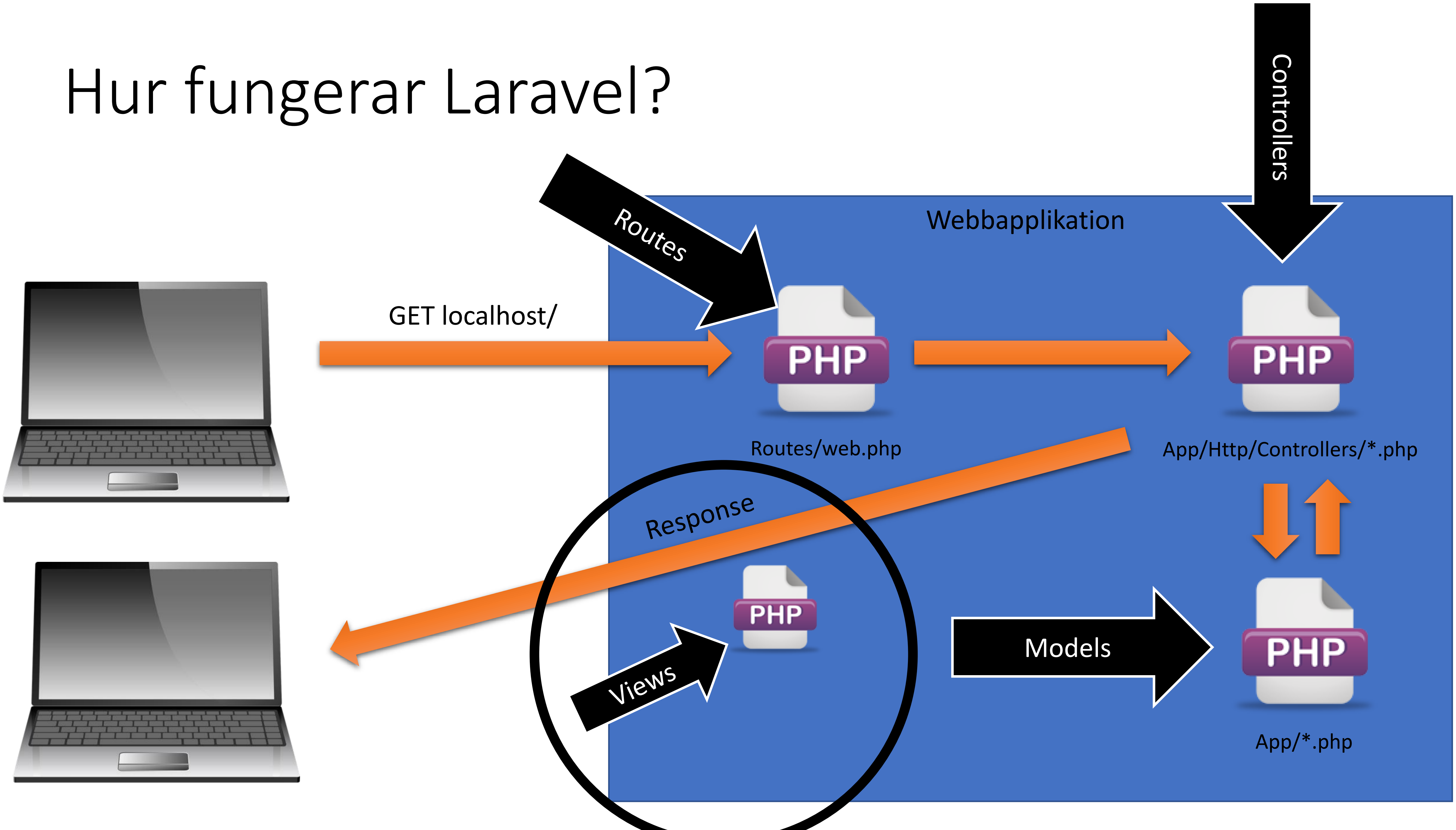
Laravel Passport

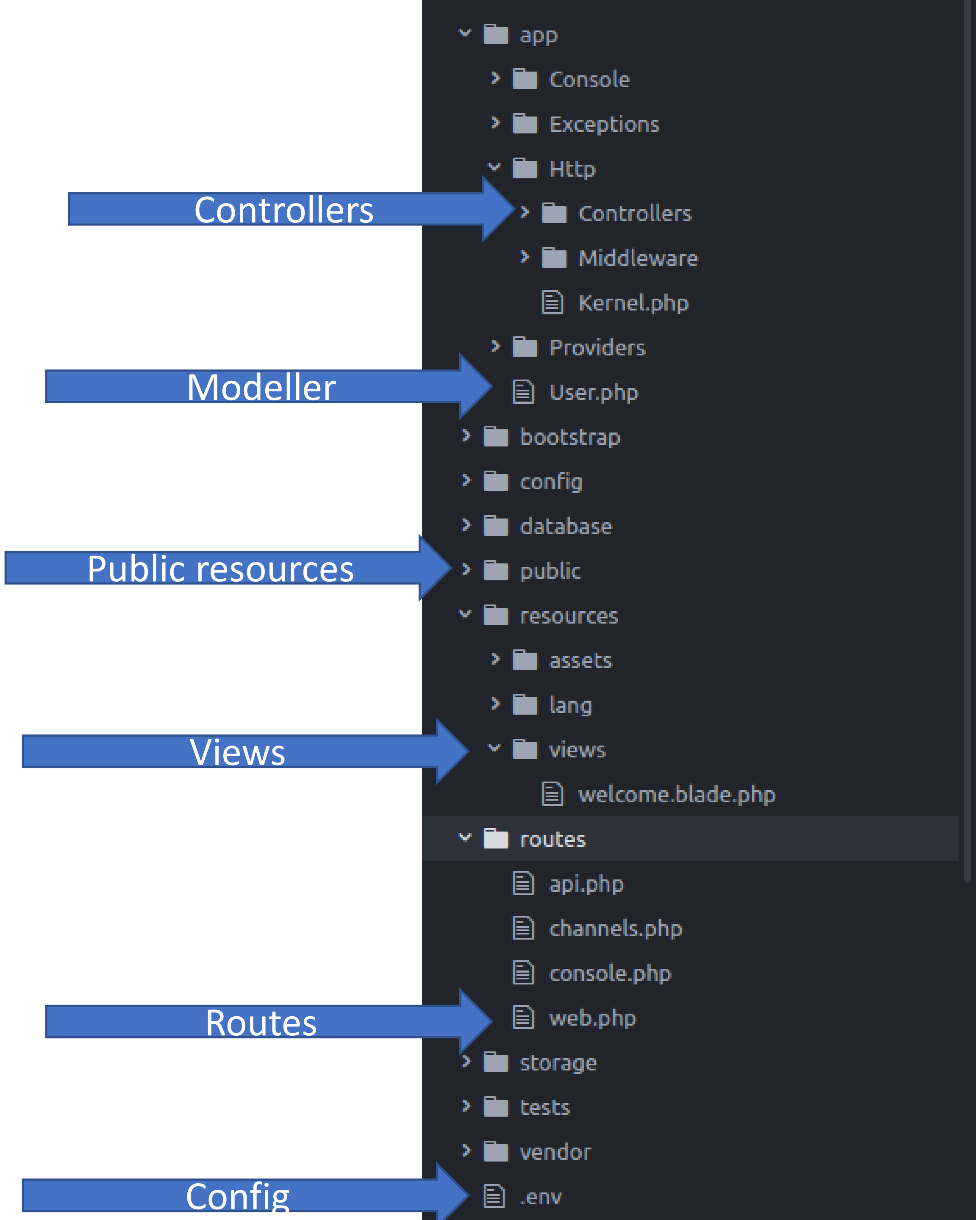
https://laravel.com/



https://selftaughtcoders.com/wp-content/uploads/2015/07/mvc_diagram_with_routes.png

Hur fungerar Laravel?





```
app
  Console
  Exceptions
  Http
    Controllers
    Middleware
    Kernel.php
  Providers
  User.php
bootstrap
config
database
public
resources
  assets
  lang
  views
    welcome.blade.php
routes
  api.php
  channels.php
  console.php
  web.php
storage
tests
vendor
.env
```

Controllers

Modeller

Public resources

Views

Routes

Config

Lumen =>

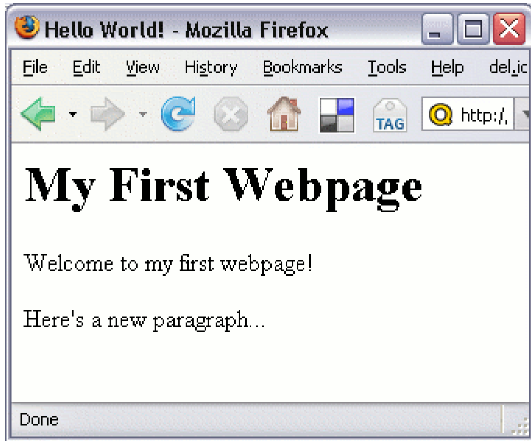
Routes
Controllers
Models
Database

=> JSON-data

Laravel =>

Routes
Controllers
Models
Database

=> Webbsida

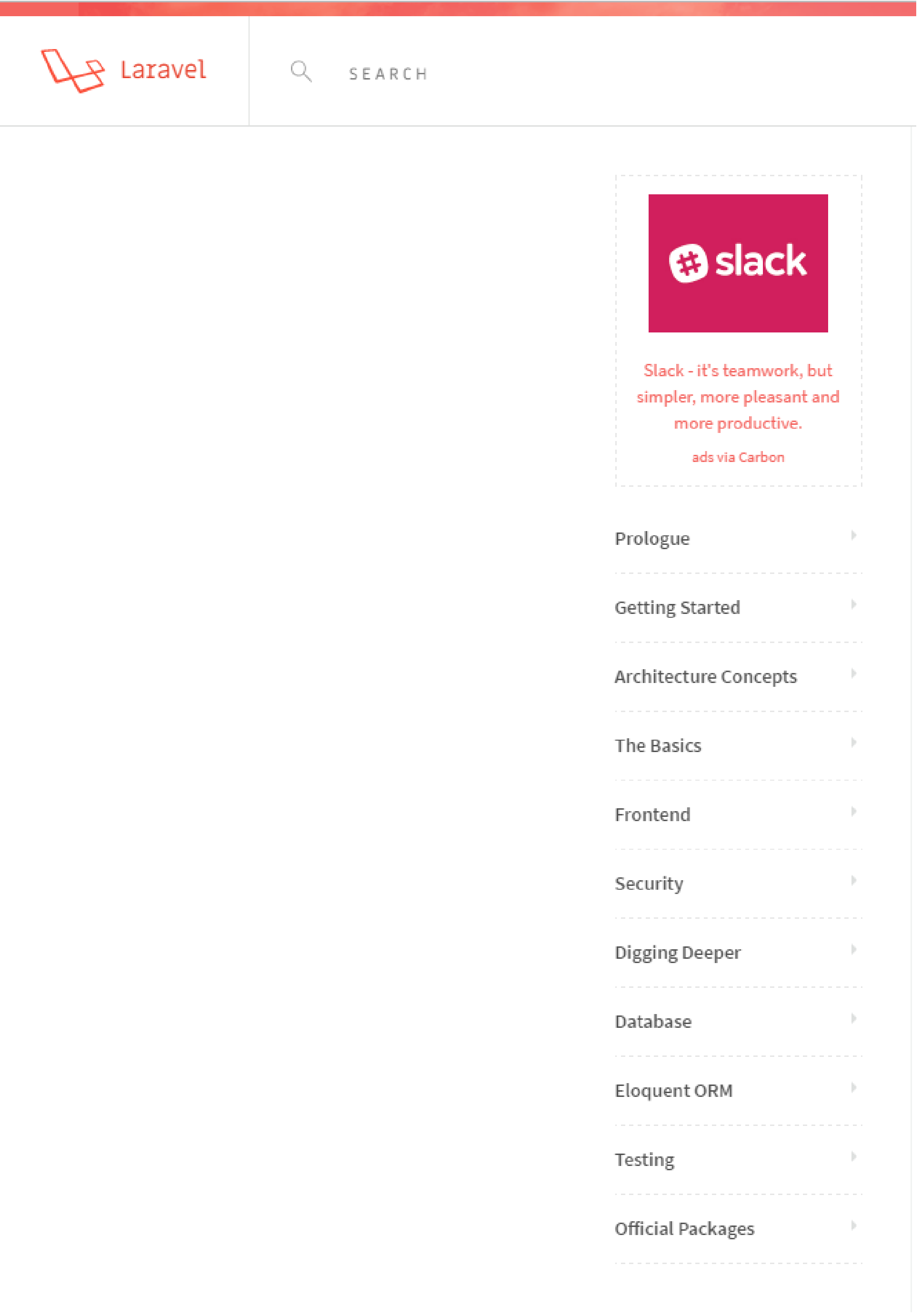


My First Webpage

Welcome to my first webpage!

Here's a new paragraph...

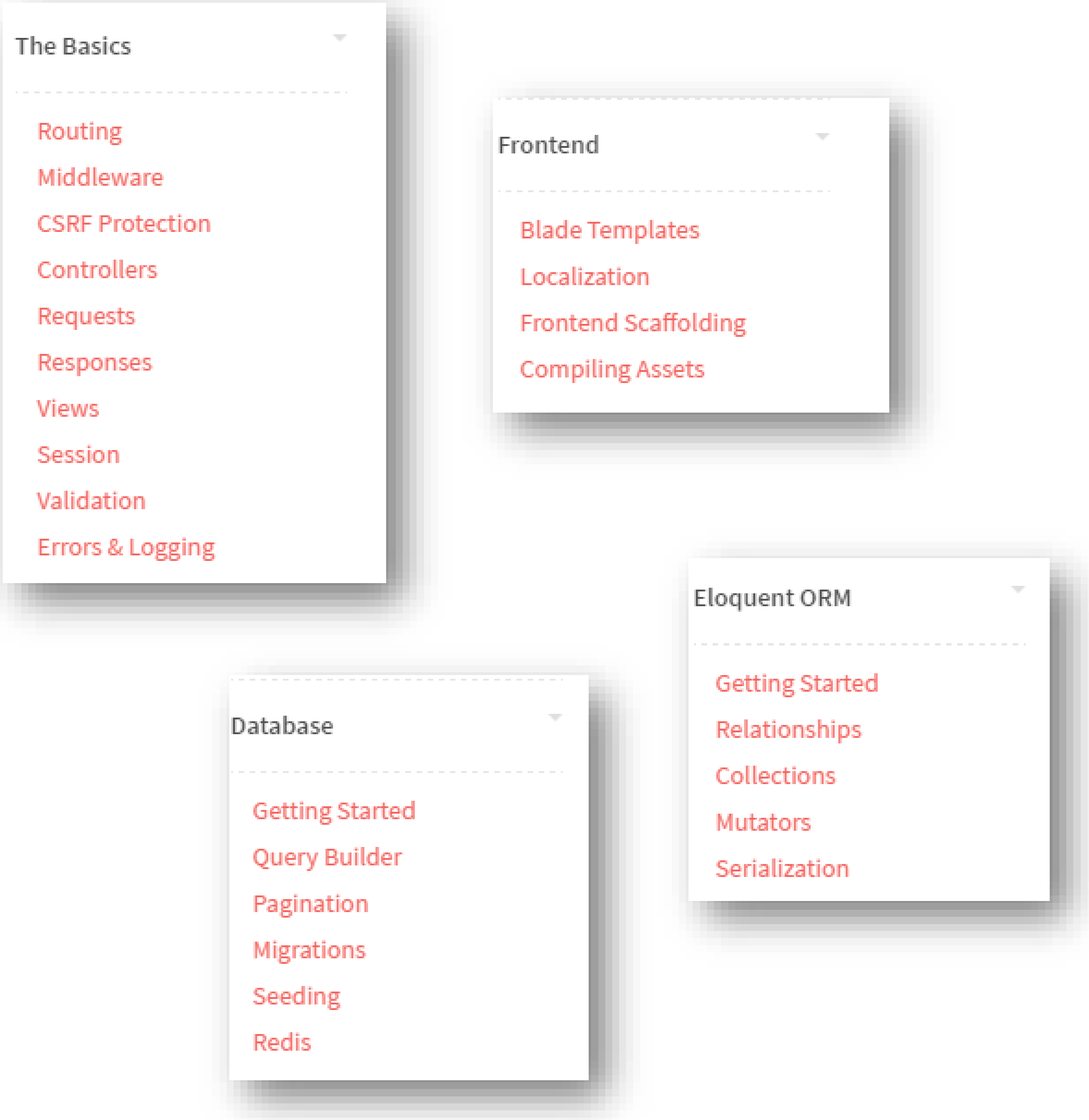
Done



Laravel

SEARCH

- Prologue
- Getting Started
- Architecture Concepts
- The Basics
- Frontend
- Security
- Digging Deeper
- Database
- Eloquent ORM
- Testing
- Official Packages



The Basics

- Routing
- Middleware
- CSRF Protection
- Controllers
- Requests
- Responses
- Views
- Session
- Validation
- Errors & Logging

Frontend

- Blade Templates
- Localization
- Frontend Scaffolding
- Compiling Assets

Database

- Getting Started
- Query Builder
- Pagination
- Migrations
- Seeding
- Redis

Eloquent ORM

- Getting Started
- Relationships
- Collections
- Mutators
- Serialization

php artisan

Ett hjälpmedel för att snabba upp utvecklingen av webbapplikationer

Resurser till vår webbplats

CSS / JavaScript / Bilder / etc.

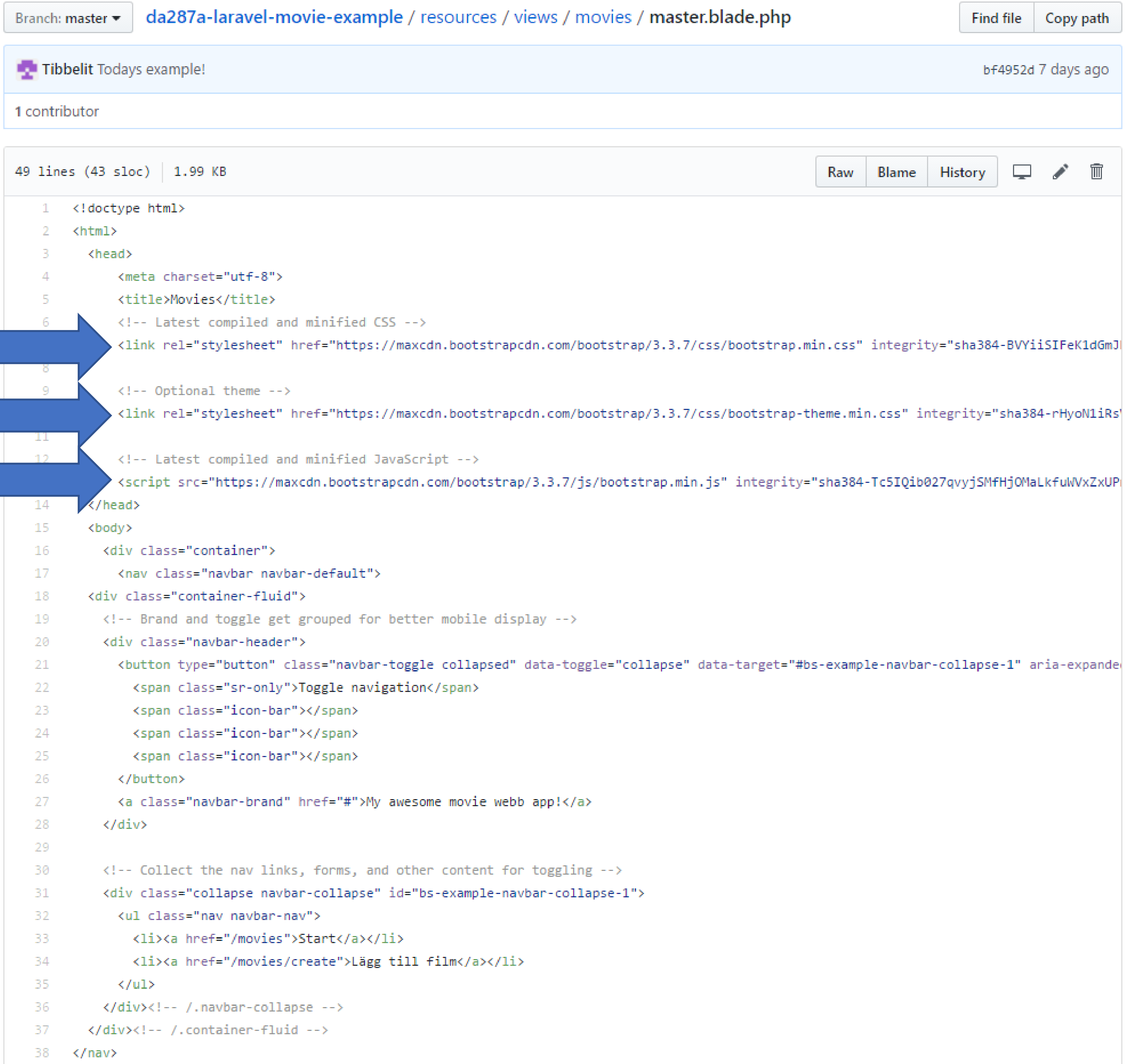
Hur s en?

https://githu
example/blc

CSS

CSS

JavaScript



da287a-laravel-movie-example / resources / views / movies / master.blade.php

Tibbelit Todays example! bf4952d 7 days ago

1 contributor

49 lines (43 sloc) | 1.99 KB

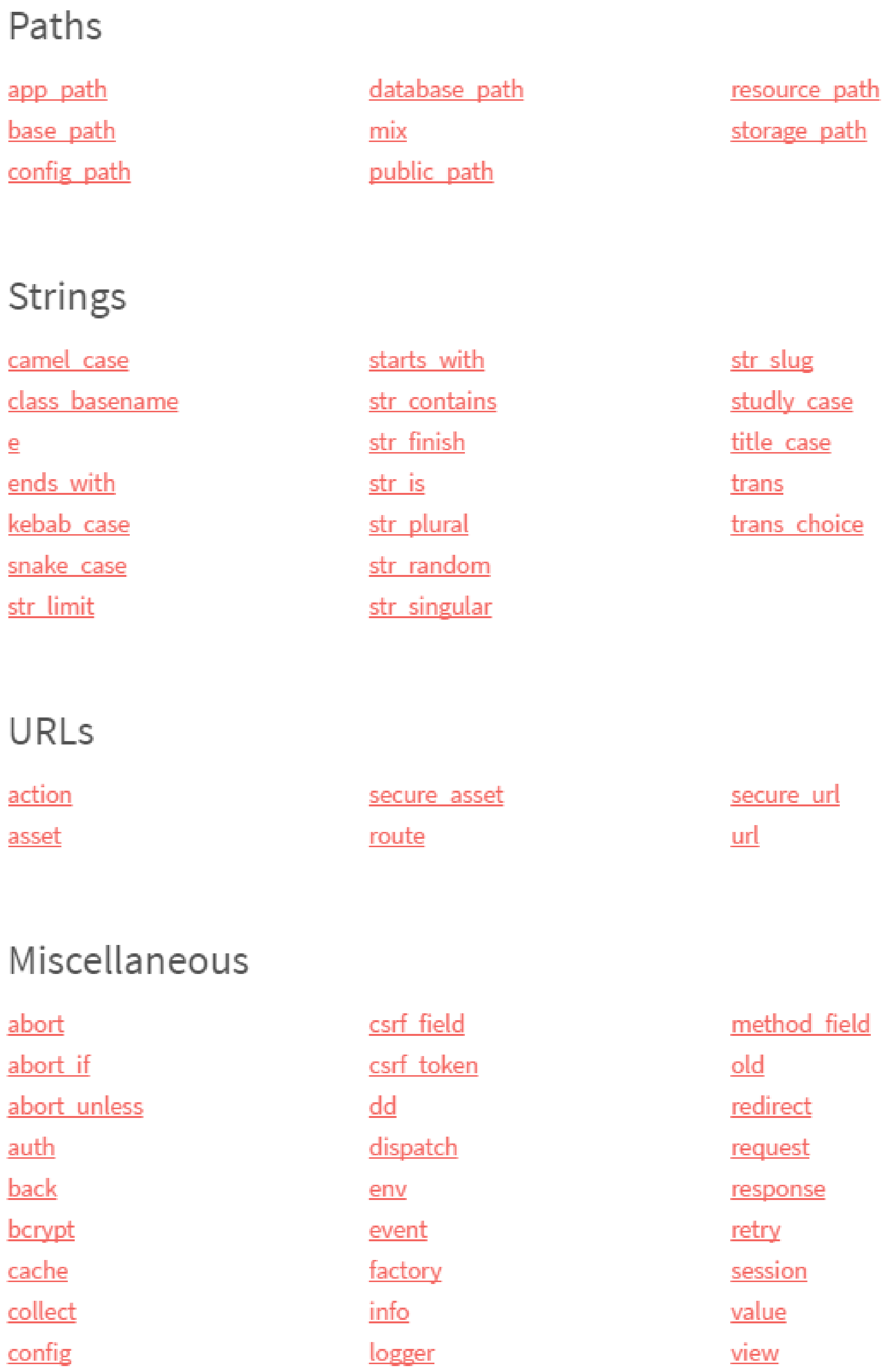
```
<!doctype html>
<html>
  <head>
      <meta charset="utf-8">
      <title>Movies</title>
      <!-- Latest compiled and minified CSS -->
      <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJ

      <!-- Optional theme -->
      <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css" integrity="sha384-rHyoN1iRs

      <!-- Latest compiled and minified JavaScript -->
      <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUP
  </head>
  <body>
    <div class="container">
      <nav class="navbar navbar-default">
  <div class="container-fluid">
    <!-- Brand and toggle get grouped for better mobile display -->
    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1" aria-expande
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="#">My awesome movie webb app!</a>
    </div>

    <!-- Collect the nav links, forms, and other content for toggling -->
    <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
      <ul class="nav navbar-nav">
        <li><a href="/movies">Start</a></li>
        <li><a href="/movies/create">Lägg till film</a></li>
      </ul>
    </div><!-- /.navbar-collapse -->
  </div><!-- /.container-fluid -->
</nav>
```

Bootstrap

Vi länkar in CSS & JavaScript från Bootstraps webbplats

För- och nackdelar med detta?

Hjälpfunktioner i Laravel



Paths

- app_path
- base_path
- config_path
- database_path
- mix
- public_path
- resource_path
- storage_path

Strings

- camel_case
- class_basename
- e
- ends_with
- kebab_case
- snake_case
- str_limit
- starts_with
- str_contains
- str_finish
- str_is
- str_plural
- str_random
- str_singular
- str_slug
- studly_case
- title_case
- trans
- trans_choice

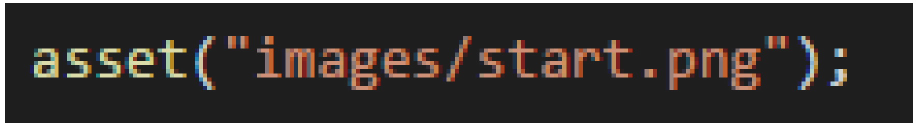
URLs

- action
- asset
- secure_asset
- route
- secure_url
- url

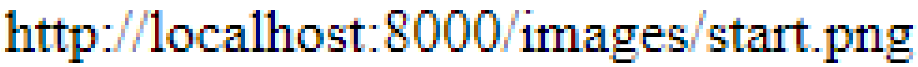
Miscellaneous

- abort
- abort_if
- abort_unless
- auth
- back
- bcrypt
- cache
- collect
- config
- csrf_field
- csrf_token
- dd
- dispatch
- env
- event
- factory
- info
- logger
- method_field
- old
- redirect
- request
- response
- retry
- session
- value
- view

Använd ”asset()”-funktionen



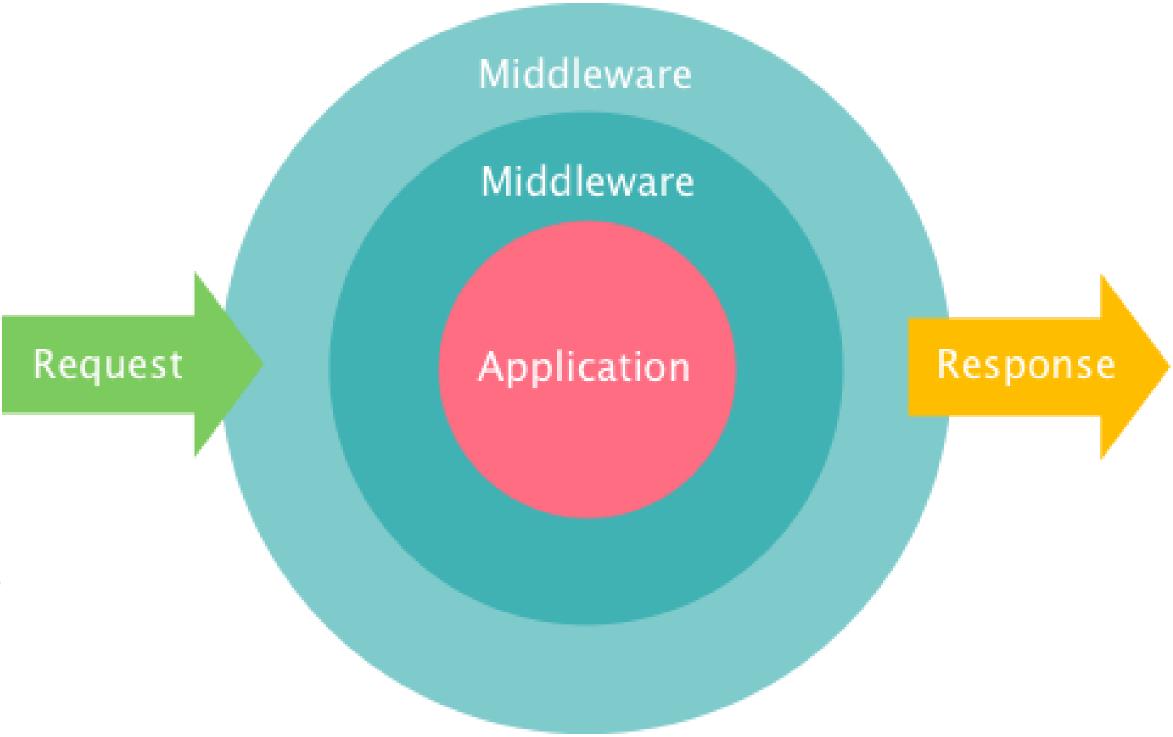
```
asset("images/start.png");
```



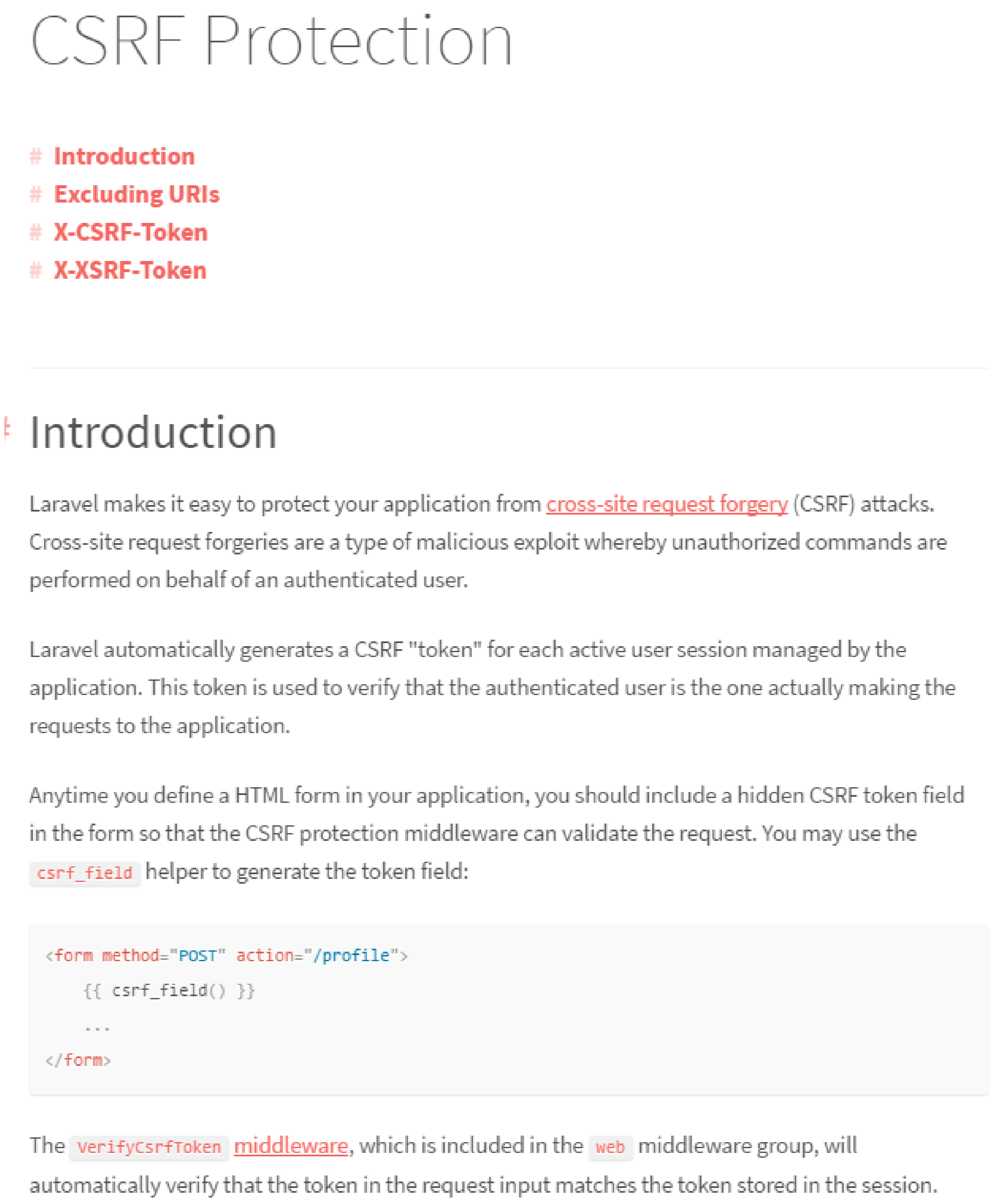
http://localhost:8000/images/start.png

Middleware

https://laravel.com/docs/5.6/middleware



Middleware
- Vi har vi sett det innan



CSRF Protection

Introduction

Laravel makes it easy to protect your application from cross-site request forgery (CSRF) attacks. Cross-site request forgeries are a type of malicious exploit whereby unauthorized commands are performed on behalf of an authenticated user.

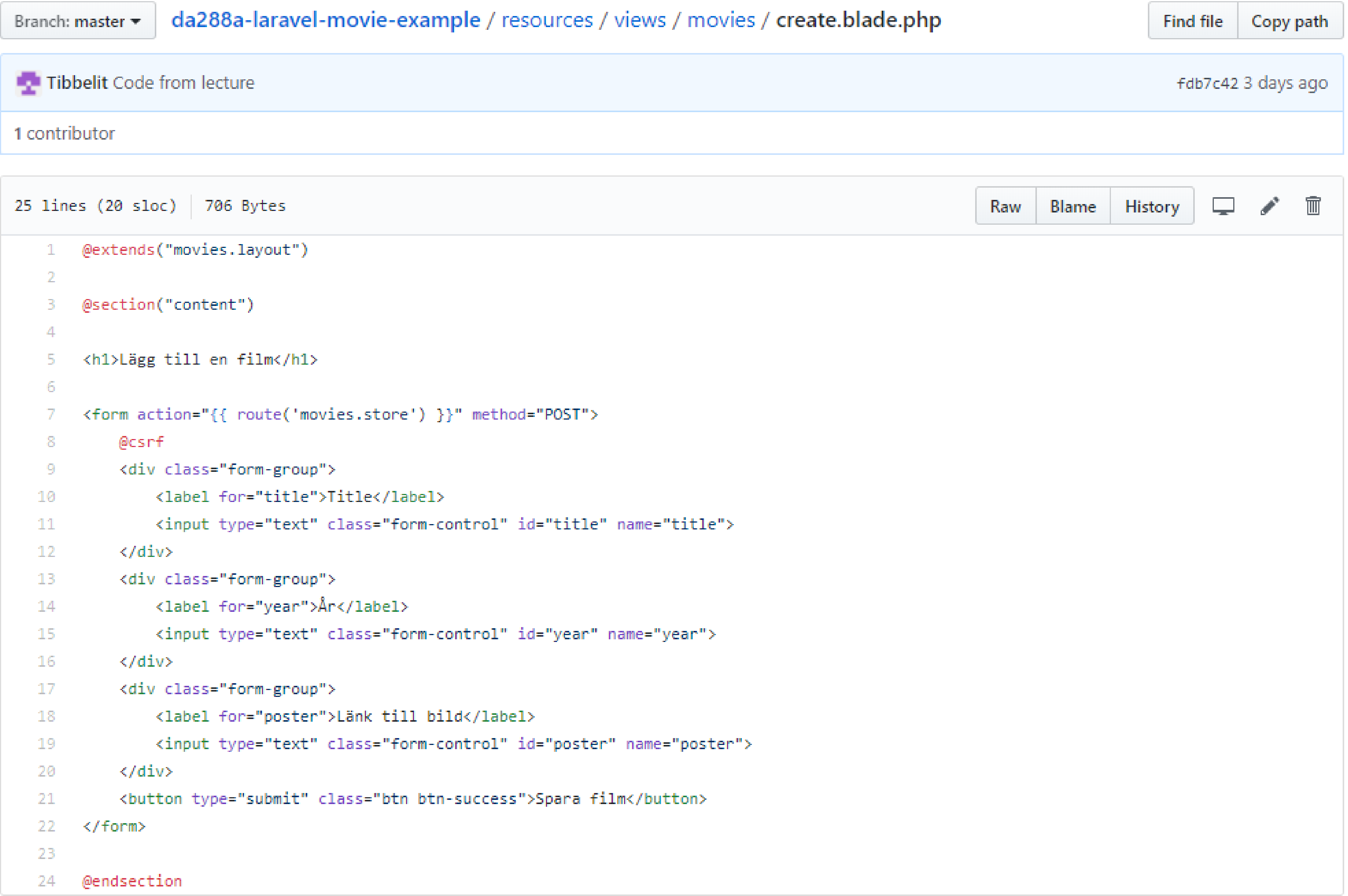
Laravel automatically generates a CSRF "token" for each active user session managed by the application. This token is used to verify that the authenticated user is the one actually making the requests to the application.

Anytime you define a HTML form in your application, you should include a hidden CSRF token field in the form so that the CSRF protection middleware can validate the request. You may use the `csrf_field` helper to generate the token field:

```
<form method="POST" action="/profile">
    {{ csrf_field() }}
    ...
</form>
```

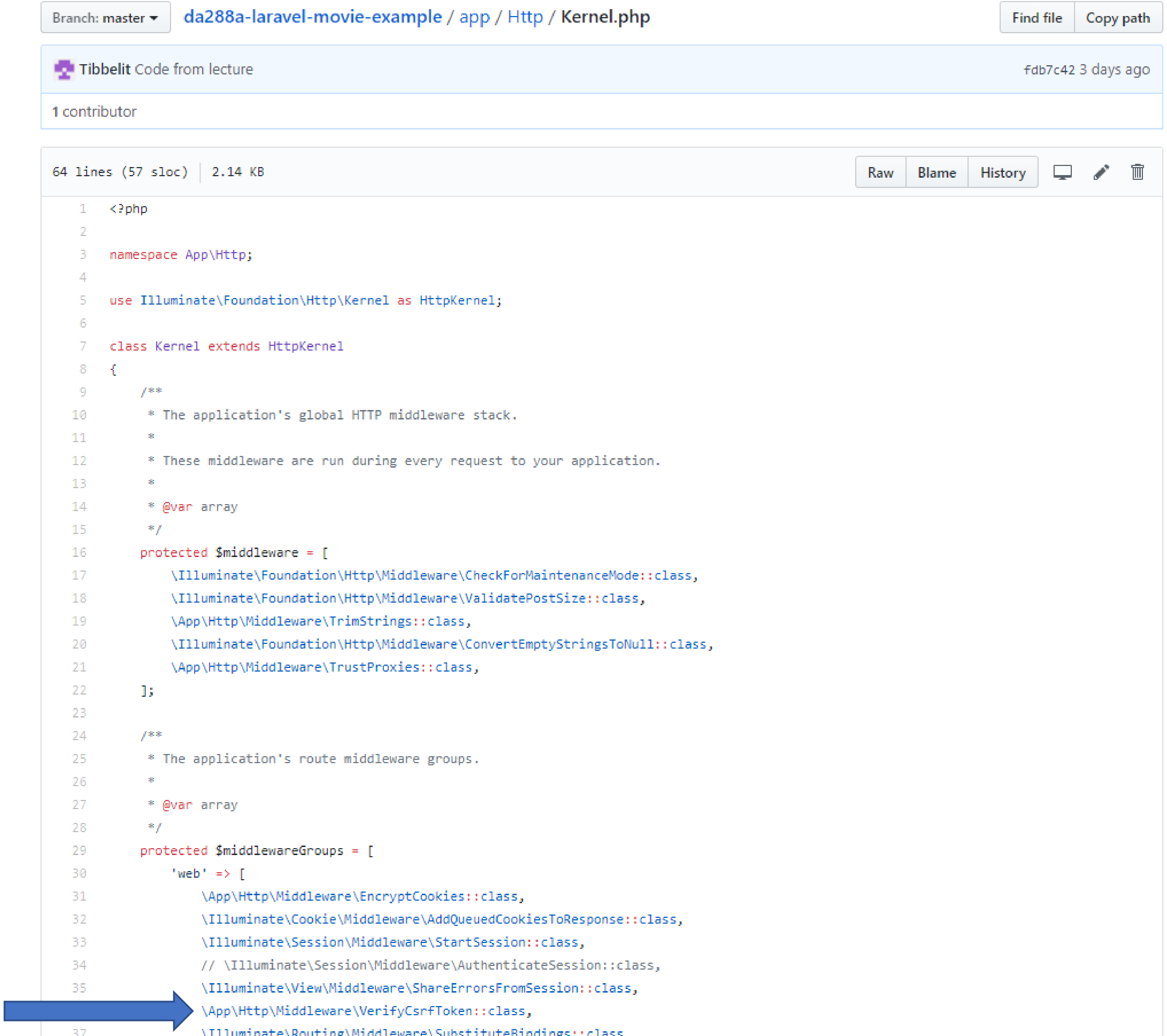
The `VerifyCsrfToken` middleware, which is included in the `web` middleware group, will automatically verify that the token in the request input matches the token stored in the session.

Exempelanvändning av middleware - CSRF



```
@extends("movies.layout")

@section("content")

<h1>Lägg till en film</h1>

<form action="{{ route('movies.store') }}" method="POST">
    @csrf
    <div class="form-group">
        <label for="title">Title</label>
        <input type="text" class="form-control" id="title" name="title">
    </div>
    <div class="form-group">
        <label for="year">År</label>
        <input type="text" class="form-control" id="year" name="year">
    </div>
    <div class="form-group">
        <label for="poster">Länk till bild</label>
        <input type="text" class="form-control" id="poster" name="poster">
    </div>
    <button type="submit" class="btn btn-success">Spara film</button>
</form>

@endsection
```

Registrera middleware



da288a-laravel-movie-example / app / Http / Kernel.php

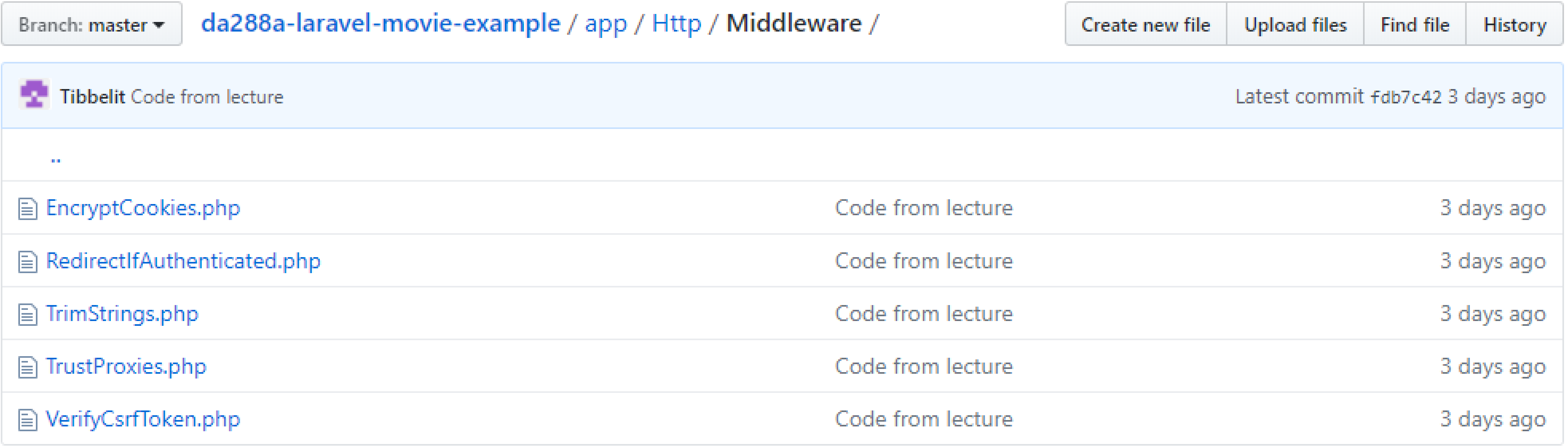
Tibbelit Code from lecture fdb7c42 3 days ago

1 contributor

64 lines (57 sloc) | 2.14 KB

```php
<?php

namespace App\Http;

use Illuminate\Foundation\Http\Kernel as HttpKernel;

class Kernel extends HttpKernel
{
    /**
     * The application's global HTTP middleware stack.
     *
     * These middleware are run during every request to your application.
     *
     * @var array
     */
    protected $middleware = [
        \Illuminate\Foundation\Http\Middleware\CheckForMaintenanceMode::class,
        \Illuminate\Foundation\Http\Middleware\ValidatePostSize::class,
        \App\Http\Middleware\TrimStrings::class,
        \Illuminate\Foundation\Http\Middleware\ConvertEmptyStringsToNull::class,
        \App\Http\Middleware\TrustProxies::class,
    ];

    /**
     * The application's route middleware groups.
     *
     * @var array
     */
    protected $middlewareGroups = [
        'web' => [
            \App\Http\Middleware\EncryptCookies::class,
            \Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::class,
            \Illuminate\Session\Middleware\StartSession::class,
            // \Illuminate\Session\Middleware\AuthenticateSession::class,
            \Illuminate\View\Middleware\ShareErrorsFromSession::class,
            \App\Http\Middleware\VerifyCsrfToken::class,
            \Illuminate\Routing\Middleware\SubstituteBindings::class
```

Placera middleware

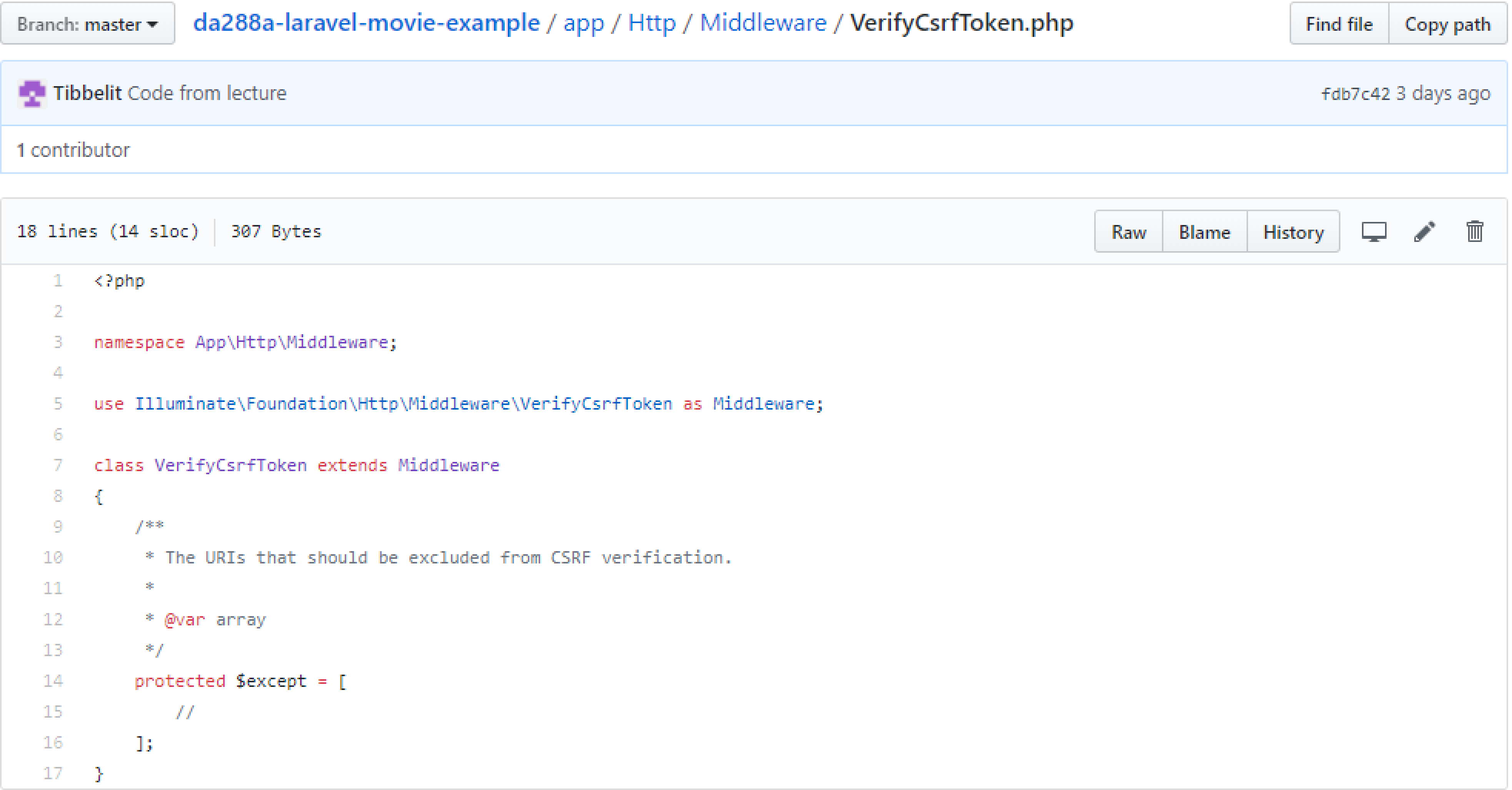


da288a-laravel-movie-example / app / Http / Middleware /

Tibbelit Code from lecture Latest commit fdb7c42 3 days ago

..		
EncryptCookies.php	Code from lecture	3 days ago
RedirectIfAuthenticated.php	Code from lecture	3 days ago
TrimStrings.php	Code from lecture	3 days ago
TrustProxies.php	Code from lecture	3 days ago
VerifyCsrfToken.php	Code from lecture	3 days ago

Exempelmiddleware



da288a-laravel-movie-example / app / Http / Middleware / VerifyCsrfToken.php

Tibbelit Code from lecture

fdb7c42 3 days ago

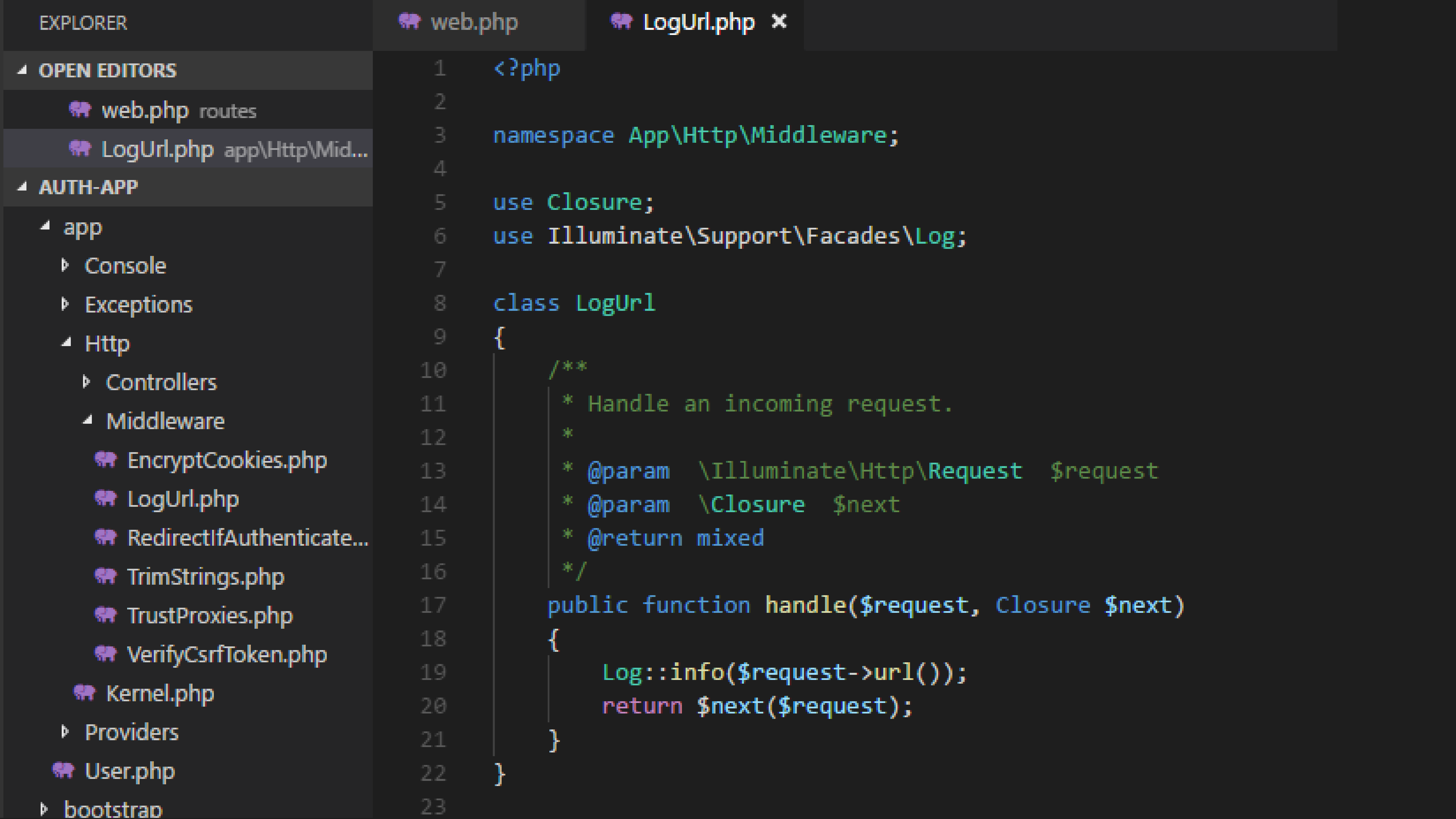
1 contributor

18 lines (14 sloc) | 307 Bytes

```php
<?php

namespace App\Http\Middleware;

use Illuminate\Foundation\Http\Middleware\VerifyCsrfToken as Middleware;

class VerifyCsrfToken extends Middleware
{
    /**
     * The URIs that should be excluded from CSRF verification.
     *
     * @var array
     */
    protected $except = [
        //
    ];
}
```

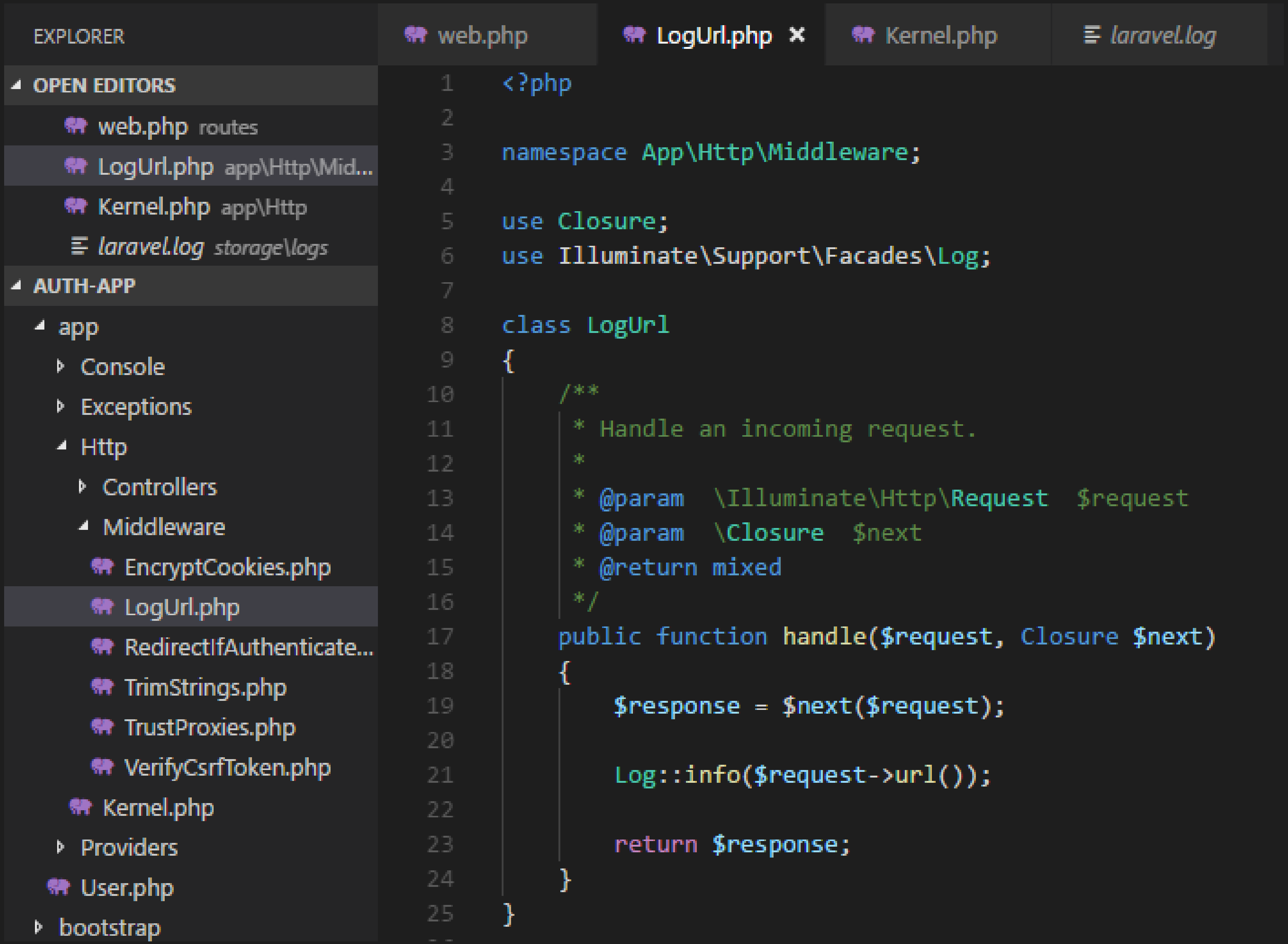
php artisan make:middleware ExampleMiddleware

Skapa en middleware



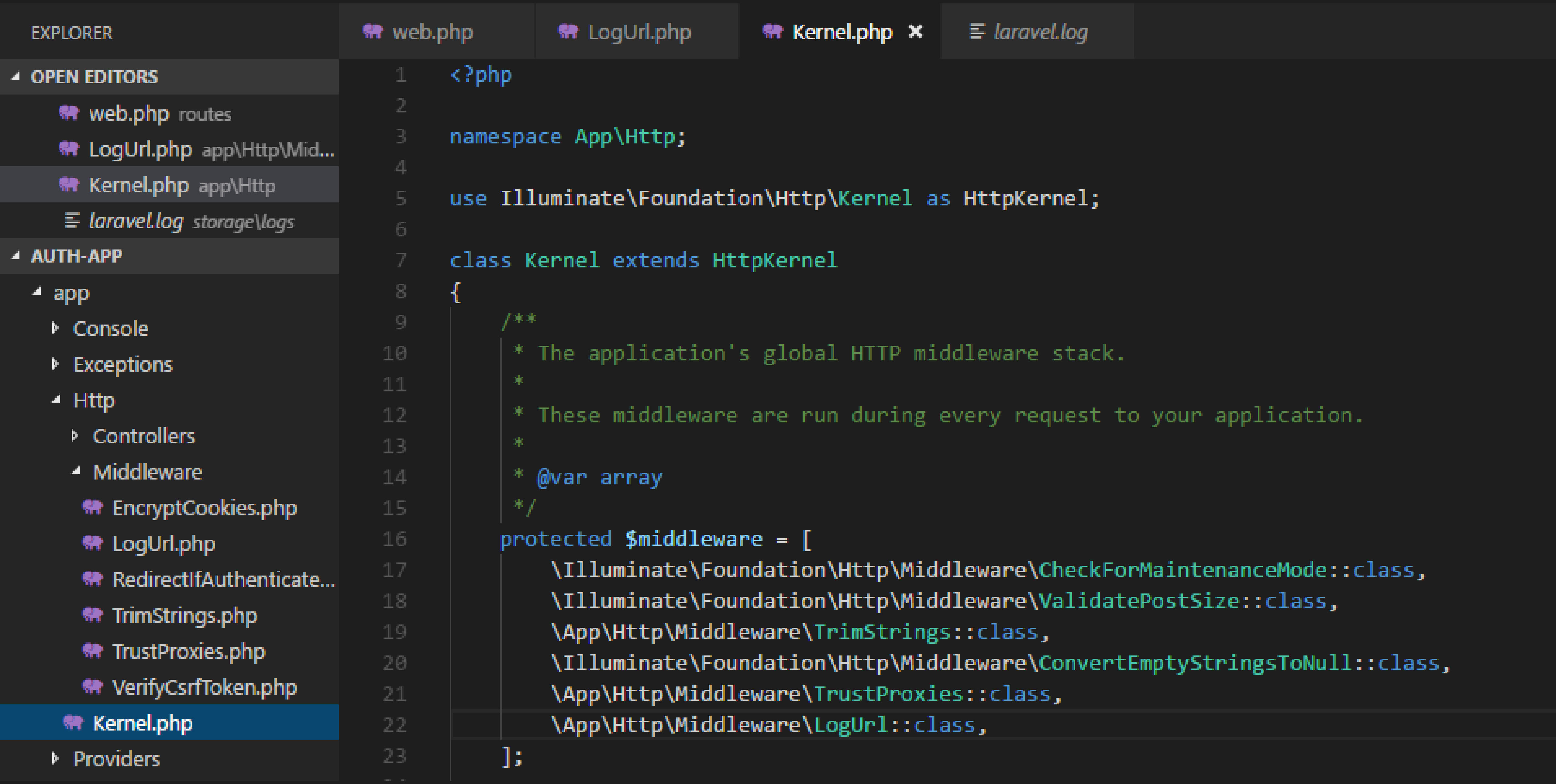
```php
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Support\Facades\Log;

class LogUrl
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @return mixed
     */
    public function handle($request, Closure $next)
    {
        Log::info($request->url());
        return $next($request);
    }
}

```

Efter ett request?



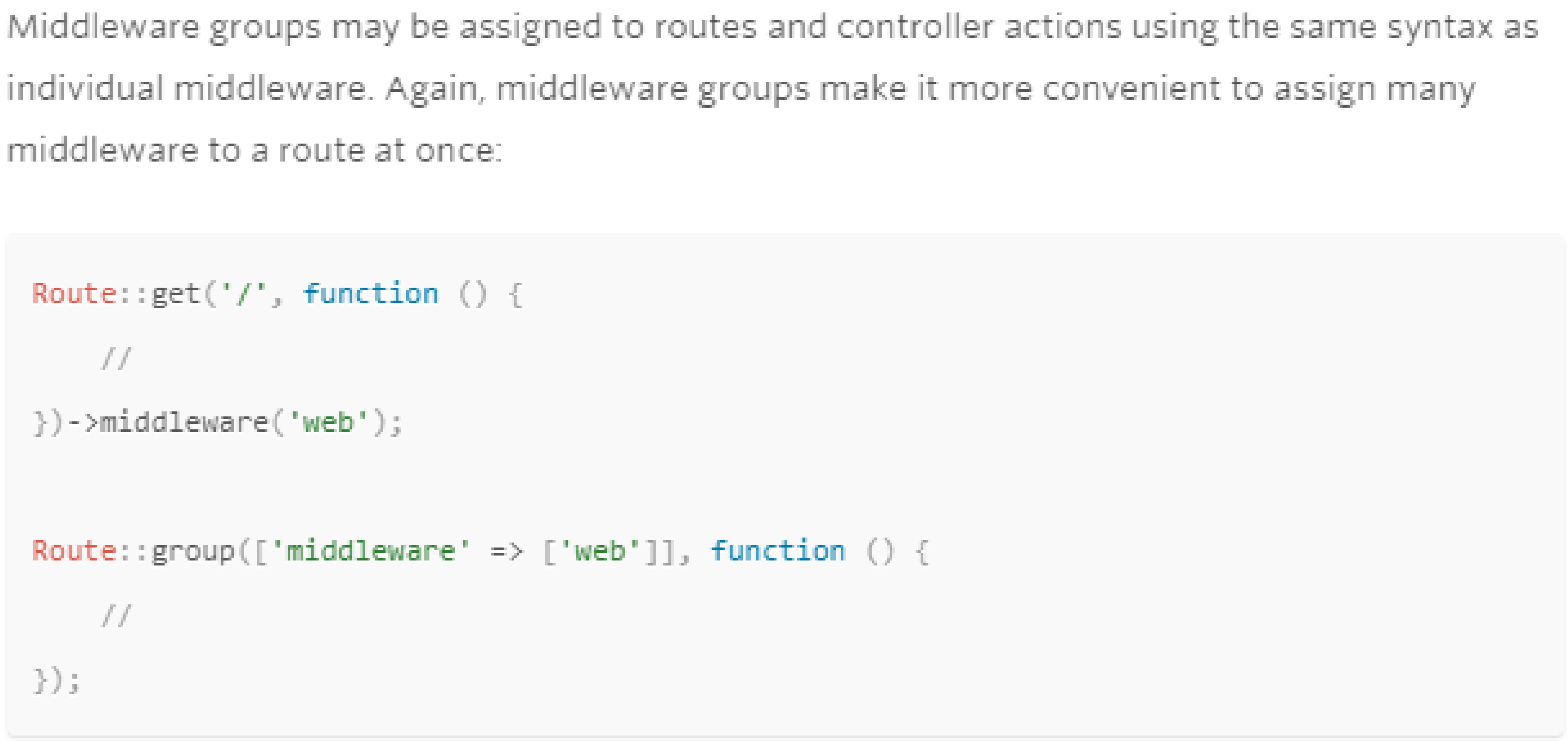
```php
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Support\Facades\Log;

class LogUrl
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @return mixed
     */
    public function handle($request, Closure $next)
    {
        $response = $next($request);

        Log::info($request->url());

        return $response;
    }
}
```

Registrera en middleware



```php
<?php

namespace App\Http;

use Illuminate\Foundation\Http\Kernel as HttpKernel;

class Kernel extends HttpKernel
{
    /**
     * The application's global HTTP middleware stack.
     *
     * These middleware are run during every request to your application.
     *
     * @var array
     */
    protected $middleware = [
        \Illuminate\Foundation\Http\Middleware\CheckForMaintenanceMode::class,
        \Illuminate\Foundation\Http\Middleware\ValidatePostSize::class,
        \App\Http\Middleware\TrimStrings::class,
        \Illuminate\Foundation\Http\Middleware\ConvertEmptyStringsToNull::class,
        \App\Http\Middleware\TrustProxies::class,
        \App\Http\Middleware\LogUrl::class,
    ];
```

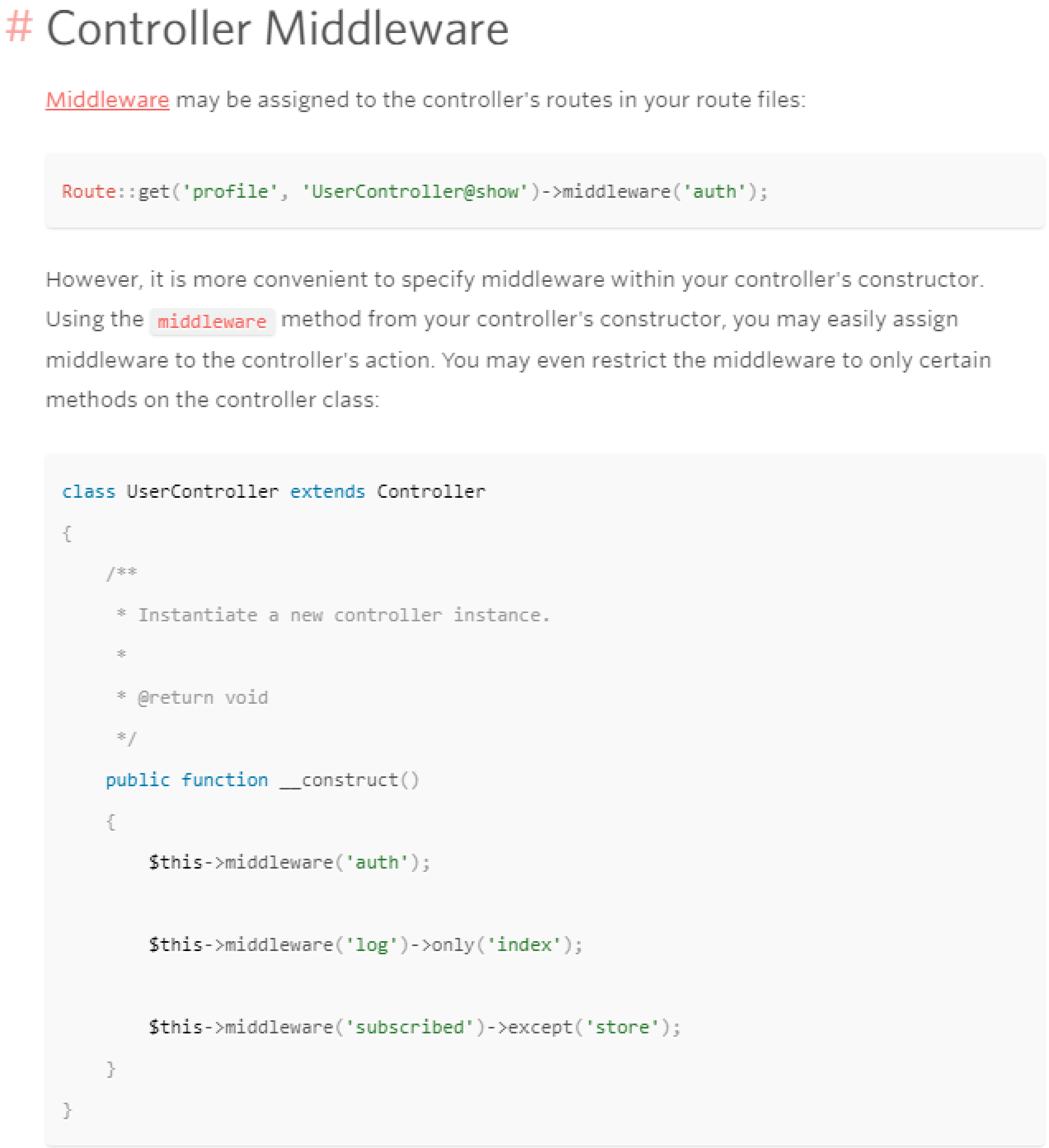
Middleware som ej gäller alla routes



Middleware groups may be assigned to routes and controller actions using the same syntax as individual middleware. Again, middleware groups make it more convenient to assign many middleware to a route at once:

```
Route::get('/', function () {
    //
})->middleware('web');

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

https://laravel.com/docs/5.6/middleware



Controller Middleware

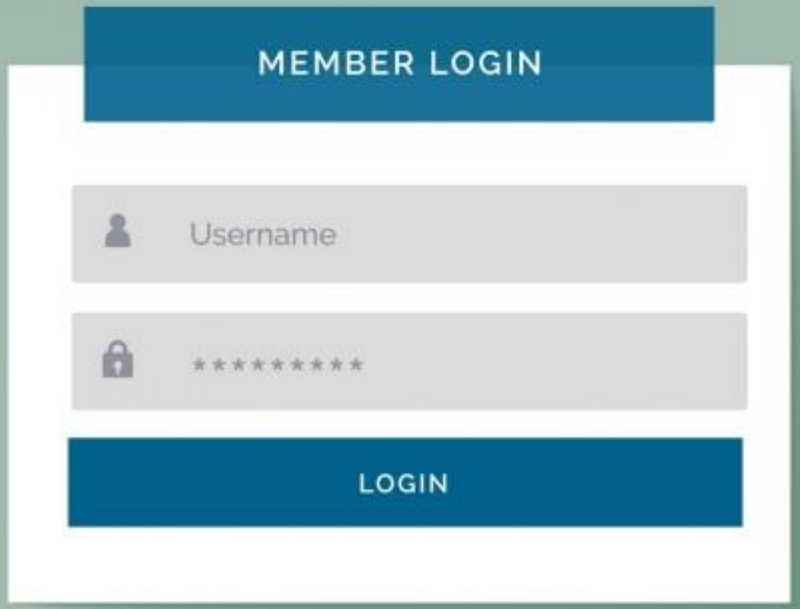
Middleware may be assigned to the controller's routes in your route files:

```
Route::get('profile', 'UserController@show')->middleware('auth');
```

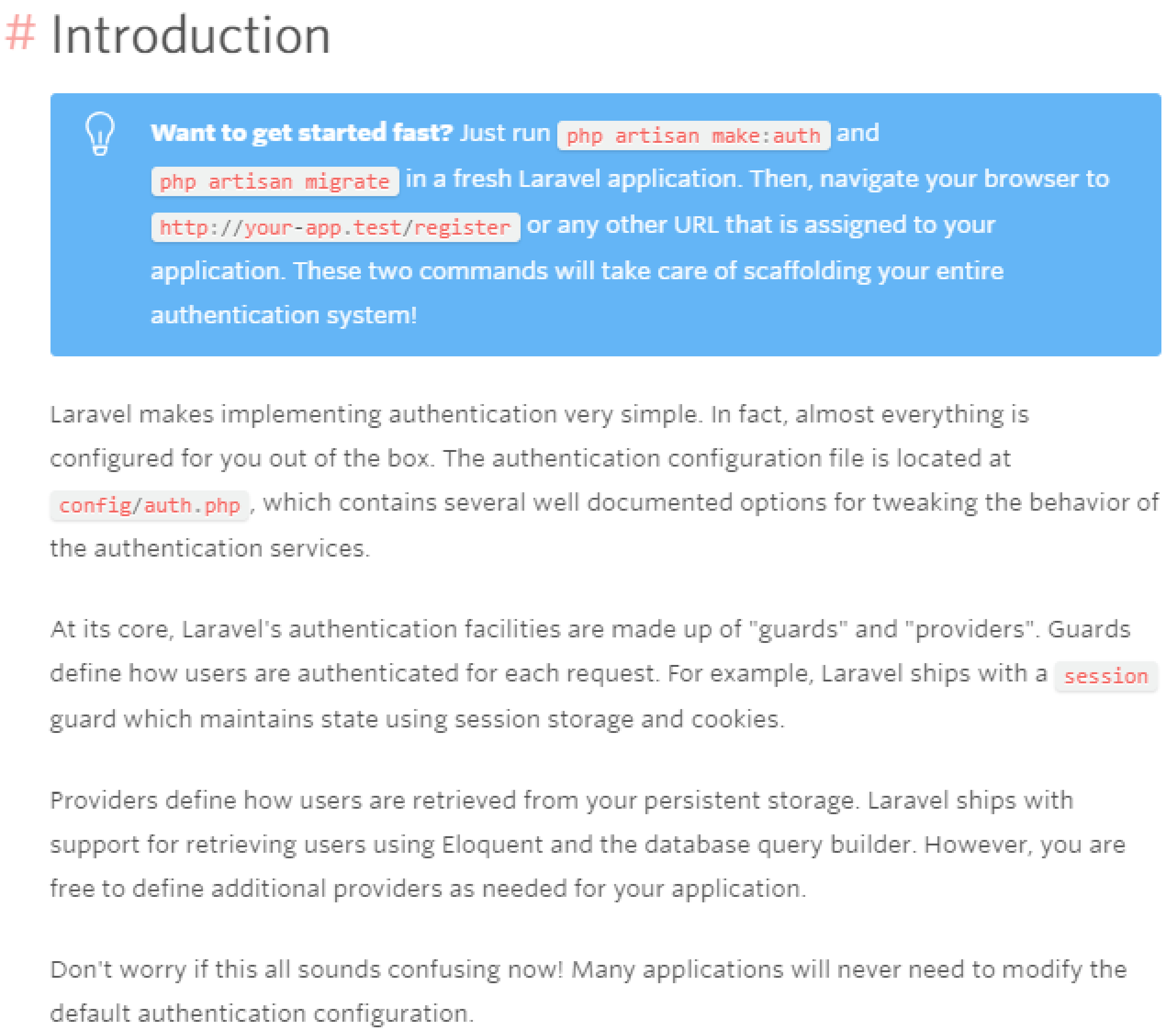
However, it is more convenient to specify middleware within your controller's constructor. Using the `middleware` method from your controller's constructor, you may easily assign middleware to the controller's action. You may even restrict the middleware to only certain methods on the controller class:

```
class UserController extends Controller
{
    /**
     * Instantiate a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('auth');

        $this->middleware('log')->only('index');

        $this->middleware('subscribed')->except('store');
    }
}
```

Demo - middleware



MEMBER LOGIN
Username
****
LOGIN



Introduction

> **Want to get started fast?** Just run `php artisan make:auth` and `php artisan migrate` in a fresh Laravel application. Then, navigate your browser to `http://your-app.test/register` or any other URL that is assigned to your application. These two commands will take care of scaffolding your entire authentication system!

Laravel makes implementing authentication very simple. In fact, almost everything is configured for you out of the box. The authentication configuration file is located at `config/auth.php`, which contains several well documented options for tweaking the behavior of the authentication services.

At its core, Laravel's authentication facilities are made up of "guards" and "providers". Guards define how users are authenticated for each request. For example, Laravel ships with a `session` guard which maintains state using session storage and cookies.

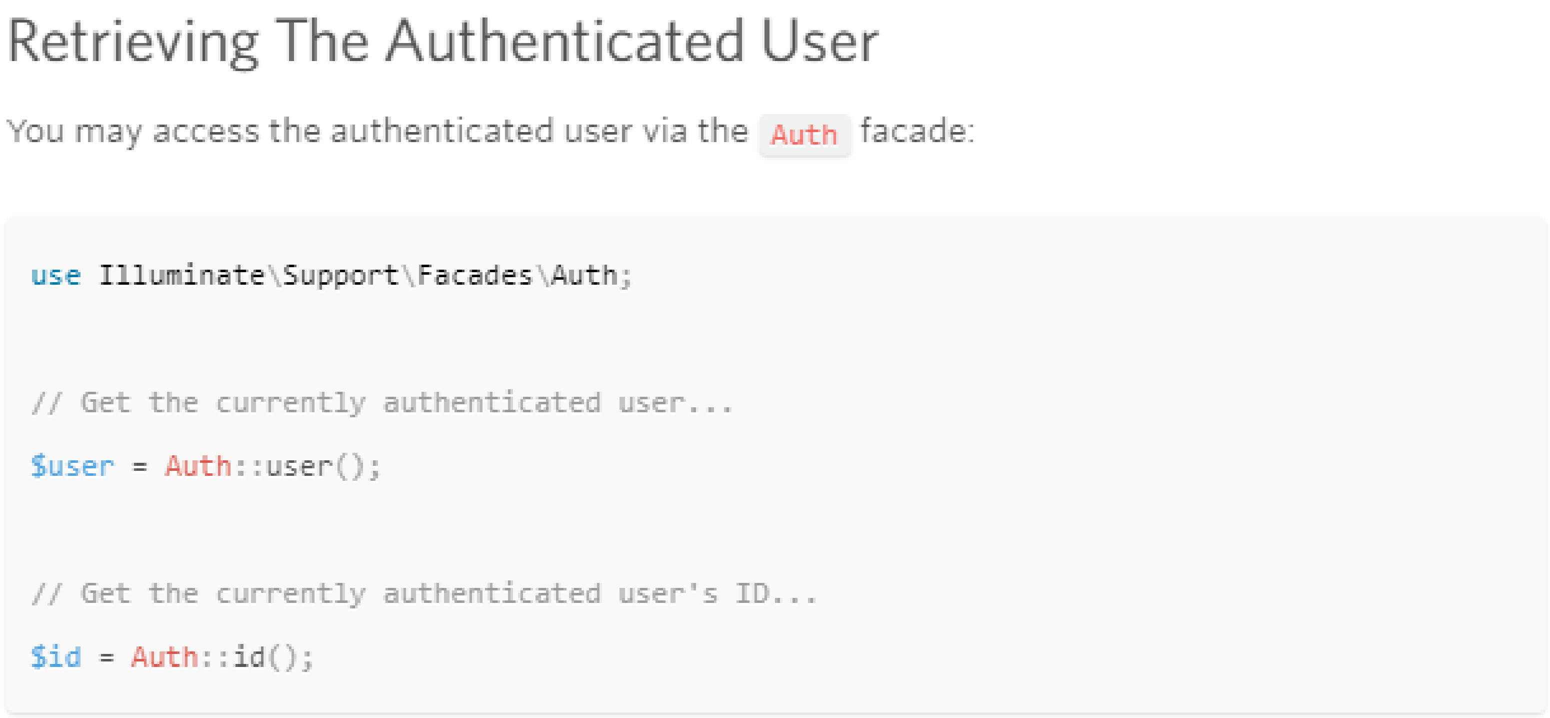
Providers define how users are retrieved from your persistent storage. Laravel ships with support for retrieving users using Eloquent and the database query builder. However, you are free to define additional providers as needed for your application.

Don't worry if this all sounds confusing now! Many applications will never need to modify the default authentication configuration.

php artisan make:auth

- Skapar routes för inloggning
    - Kontrollera routes genom kommando: *php artisan route:list* i terminalen
- Skapar Controller för *Auth* (mappen *Auth* under *Controllers*)
- Skapar vyer för inloggning

- Vi behöver
    - Skapa en databas
    - Uppdatera vår .env-fil så att den stämmer överns med databasen
    - Migrera databas

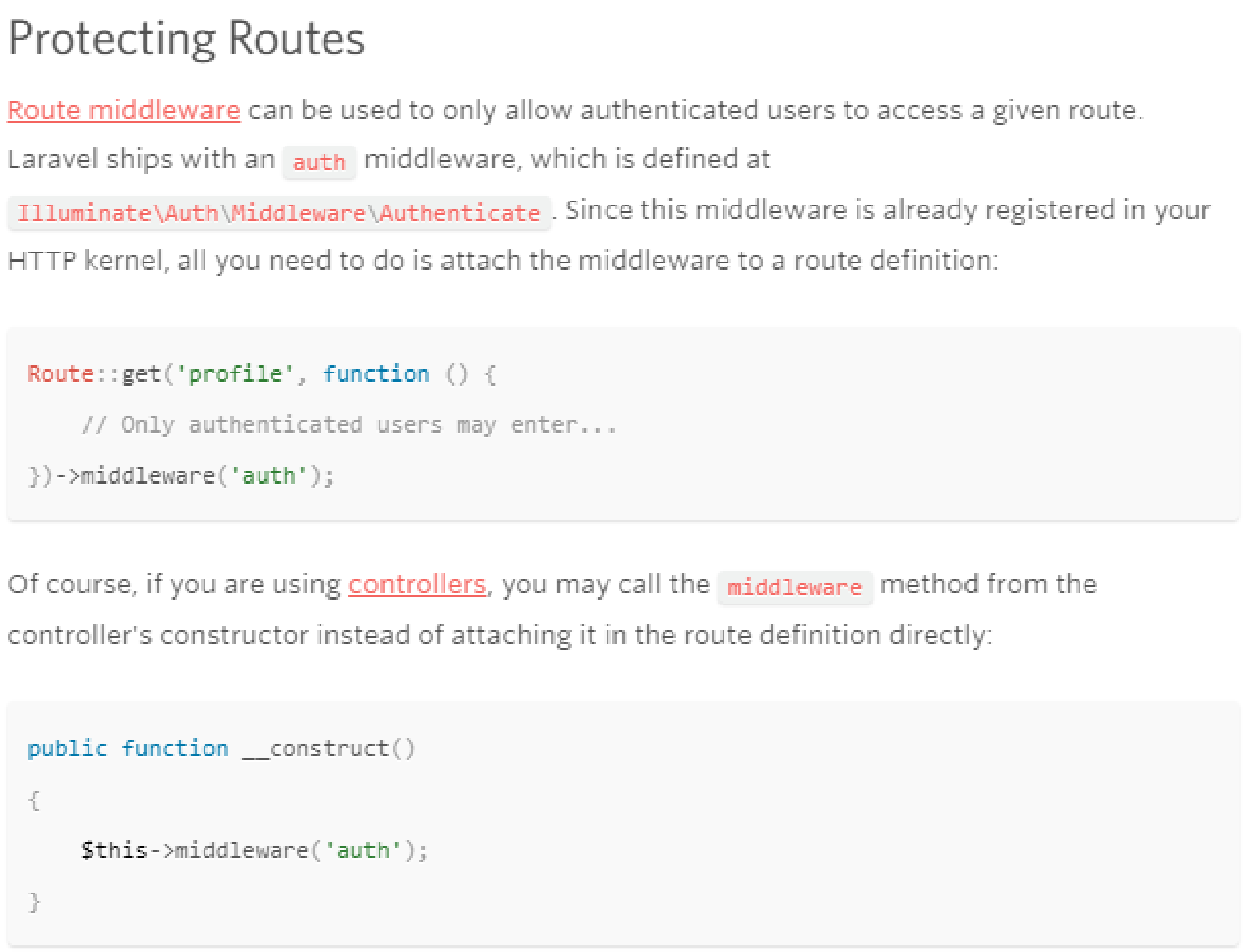


Retrieving The Authenticated User

You may access the authenticated user via the `Auth` facade:

```php
use Illuminate\Support\Facades\Auth;

// Get the currently authenticated user...
$user = Auth::user();

// Get the currently authenticated user's ID...
$id = Auth::id();
```

https://laravel.com/docs/5.6/authentication



Protecting Routes

Route middleware can be used to only allow authenticated users to access a given route. Laravel ships with an `auth` middleware, which is defined at `Illuminate\Auth\Middleware\Authenticate`. Since this middleware is already registered in your HTTP kernel, all you need to do is attach the middleware to a route definition:

```
Route::get('profile', function () {
    // Only authenticated users may enter...
})->middleware('auth');
```

Of course, if you are using controllers, you may call the `middleware` method from the controller's constructor instead of attaching it in the route definition directly:

```
public function __construct()
{
    $this->middleware('auth');
}
```

https://laravel.com/docs/5.6/authentication

Demo!

Sammanfattning av denna del i kursen

Reflektion i kursen

- Beskriv följande delar i kursen:
  - PHP
  - Pakethanterare
  - Versionshantering
  - Lumen
  - Laravel
- Inom MVC och webbramverk beskriv följande delar i kursen:
  - Routes
  - Controllers
  - Models
  - Views
  - ORM
  - Migration & seeds
  - Authentisering
  - Middleware

Kommande delar i kursen

- Testning
- Refaktorisering
- Integration
- Deployment
- Delivery
- => Att publicera ett projekt