web разработка

Laravel Middleware
Ограничаване на достъп

Съдържание

Laravel Auth
Middlewares
Authorisation through FB - домашно
Sending mail - домашно
Групиране на пътища

Ресурси

https://laracasts.com/

Laravel 5 beauty

Задача 1

Отпечатайте всички лекции на страница лекции с бутон за промяна и изтриване
Опечатайте списък на всички студенти с бутон за изтриване и бутон към профила им.
Ограничете достъпа -
Всеки, след като се логне може да вижда
    Списъците с лекции и студенти
    Всеки студент може да вижда и редактира профила си
Само администраторът може да
        Редактира, изтрива лекции, студенти

Laravel Auth

Laravel Auth

Изисквания към базата данни

По подразбиране Ларавел включва App\User Eloquent model в app директорията на вашия проект.. Този модел може да бъде използван с основния драйвер за проверка на Eloquent.Ако приложението ви не използва Eloquent, може да използвате вариант за проверка спрямо базата данни като използвате query builder-а на Ларавел..

Структурирайки таблицата за App\User модела в базата данни направете колоната за парола да е с дължина поне 60 знака. Може да оставите дължината по подразбиране 255 знака.

Не забравяйте, че таблицата users (или иквивалентната) съдържа колона със стойност по подразбиране null за стринга remember_token column с дължина 100 знака. Колоната е предназначена за запазване на token за потребителите, които ще селектират опцията "remember me" при вход във вашата система.

Laravel Auth

С инсталирането на Ларавел, получавате няколко предварително създадени контролери за проверка/authentication controllers/, разположени в App\Http\Controllers\Auth namespace.

RegisterController - отговаря за регистрацията на новите потребители.

LoginController - отговаря за проверката на потребителите при опит за влизане.

ForgotPasswordController - отговаря за e-mailing линковете за възстановяване/промяна на паролата.

ResetPasswordController - съдържа логика са възстановяване на пароли.

Всеки от контролерите използва trait, чрез който се включват нужните методи.

Тези контролери могат да бъдат използвани в голяма част от случаите без да е необходима модификация.

Laravel Auth

Пътища

Ларавел ви предоставя бърз начин за създаване на пътища и вюта, нужни за проверка на потребителите чрез командата

php artisan make:auth

Командата инсталира
views
layouts/app.blade.php
registration
Passwords/ reset, email
login
Home.blade.php /dashboard/
И нужните пътища за проверка на потребителите.
Добавя и HomeController, който се грижи за заявките след вход на потребителите в началната страница - т.н. dashboard.

Laravel Auth

Web.php

```
Auth::routes();

Route::get('/home', 'HomeController@index')->name('home');
```

Проверете пътищата, които са регистрирани след изпълнение на make:auth

Laravel Auth

Проверка на потребителите

Сега може да регистрирате и проверявате потребителите, искащи достъп до вашето приложение. yourdomain.app/public - ще ви препраща към yourdomain.app/public/login за проверка или регистрация.

Което все още не означава, че достъпът до останалите пътища в приложението ви е ограничен за неоторизирани потребители!

Laravel Auth

Промяна на пътищата по подразбиране

След успешна проверка, потребителя се пренасочва към /home URI.
Може да промените пътя, към който се пренасочва потребителя като дефинирате свойството redirectTo в LoginController, RegisterController, и ResetPasswordController:

```
protected $redirectTo = '/';//мястото, към което искате да насочите потребителя
```

Може да дефинирате и метод redirectTo вместо свойство, ако е необходима допълнителна логика.

```
protected function redirectTo()
    {
        return '/path';
    }
```

Laravel Auth

По подразбиране Ларавел използва email за проверка на потребителя.

Може да промените това, като дефинирате username метод в LoginController:

```
public function username()
    {
        return 'username';
    }
```

Laravel Auth

Промени във валидация/съхранявне

За да промените задължителните полета при регистрация на потребител, или за да промените начина, по който всеки нов потребител се запазва в базата данни, може да променяте RegisterController.

Този клас се грижи за валидацията и създаването на новите потребители във вашето приложение.

Методът validator в RegisterController съдържа правилата за валидация на новите потребители на вашето приложение.
Можете свободно да променяте този метод според конкретния случай.

Методът create в RegisterController се грижи за създаване на нови App\User записи в базата данни чрез Eloquent ORM.
Може свободно да променяте този метод според нуждите на базата данни на вашето приложение.

Laravel Auth

Допълнителна информация

Authentication

Достъп до Auth потребителя

/Влезлия с валидно име и парола потребител/

Потребителя ….

Имате достъп до успешно влезлия в прилоожението ви потребител чрез Auth facade:

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

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

/ /Get the currently authenticated user in the views

Auth::user()->name


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

….

Потребителя ….

След проверка на потребителя, може да достъпвате влезлия потребител чрез инстанция на Illuminate\Http\Request.

/Класовете, които са type-hinted автоматично ще бъдат инжектирани в методите на контролера/

Потребителя ….

```
namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProfileController extends Controller
{
    /**
     * Update the user's profile.
     *
     * @param  Request  $request
     * @return Response
     */
    public function update(Request $request)
    {
        // $request->user() returns an instance of the authenticated user...
    }
}
```

Потребителя ….

Да определим дали текущия потребител е влязъл с валидно име и парола /с цела да покажем различно съдържание или друга логика/

Може да използвате методът check от Auth facade. Връща true/false в зависимост от това дали потребителят е влязъл с парола и потребителско име.

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

if (Auth::check()) {
    // The user is logged in...
}
```

Middleware - Предпазване на пътищата в приложението

Middleware

Добавянето на възможност за проверка на потребителско име и парола, не означава, че сме защитили пътищата си от нежелан достъп.

Една част от потребителите трябва да имат достъп, други не до определени страници /пътища в приложението ни/.

Middleware

Middleware ви осигурява удобен механизъм за филтриране на HTTP заявките към/във вашето приложение.

Например - Ларавел включва по подразбиране middleware - **auth**, проверяващ дали потребителя достъпващ приложението ви е оторизиран с парола и потребителско име/email.

Ако потребителят не е оторизиран, middleware пренасочва потребителя към страницата за login.

Ако потребителят е оторизиран middleware ще позволи заявката да продължи нататък в приложението/ще я допусне в приложението.

Middleware

Разбира се, може да дефинирате допълнителни middleware класове, изпълняващи различни задачи извън проверката дали потребителя е влязъл в приложението ви с потребителско име и парола /или други данни за проверка на потребителя/.

Съществуват няколко middleware, включени в основната инсталация на Laravel framework, в това число auth и middleware за предпазване от CSRF.

Всички те и тези, които дефинирате в последствие се разполагат в **app/Http/Middleware directory.**

Middleware

Route middleware - прилагане на middleware към пътищата.

Използват се, например, за да се позволи само на проверени с парола потребители да достъпват даден път.

При инсталирането на Laravel получавате **auth middleware**, дефиниран в Illuminate\Auth\Middleware\Authenticate.

Middleware

auth middleware е регистриран в HTTP kernel и може да го прилагате към желаната дефиниция за път -

```
Route::group(['middleware' => 'auth'], function () {

    Route::get('/home', 'HomeController@index')->name('home');

    Route::resource('course', 'CoursesController');

});
```

Middleware

Може да свържете необходимия middleware, чрез метода middleware в конструктора на конкретен контролер, вместо към дефиницията на конкретния път/конкретните пътища.

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

Middleware

Прилагане на middleware към един път -

```
Route::get('movies/create', ['middleware' => 'admin', 'uses' => 'MoviesController@create']);
```

Middleware

Допълнителна информация

Как да дефинираме Middleware - стъпка по стъпка

Задача - дефинирайте Middleware Admin и го приложете върху част от дефинираните пътища.

Login with Facebook

Login with Facebook

https://github.com/laravel/socialite

https://blog.damirmiladinov.com/laravel/laravel-5.2-socialite-facebook-login.html#.WbDcXOlRXcs
https://scotch.io/tutorials/laravel-social-authentication-with-socialite

Sending Mail

Sending Mail

https://laravel.com/docs/5.4/mail

Групиране на пътища

Grouping Routes

Чрез групиране на пътища имате възможност да споделяте атрибути на пътища като например middleware, представки, namespaces, между голям брой пътища без да се налага да дефинирате тези атрибути за всеки отделен път.

Споделените атрибути се задават в масив като първи параметър към метода Route::group method.

Grouping Routes

Споделяне на Middleware

Може да задавате middleware /няколко middleware/ към групирани пътища.
Middleware се изпълняват по реда, в който са подредени в масива.

```
Route::middleware(['first', 'second'])->group(function () {
        Route::get('/', function () {
            // Uses first & second Middleware
        });

        Route::get('user/profile', function () {
            // Uses first & second Middleware
        });
    });
```

Grouping Routes

Route Prefixes

The prefix method may be used to prefix each route in the group with a given URI. For example, you may want to prefix all route URIs within the group with admin:

```
Route::prefix('admin')->group(function () {
        Route::get('users', function () {
            // Matches The "/admin/users" URL
        });
    });
```

Grouping Routes

Други примери за групиране

```
//пътища достъпни за всички
 Route::resource('portfolio', 'PortfoliosController', ['only'=>'show']);
 Route::resource('team', 'PeoplesController', ['only'=>'show']);

 // пътища достъпни за оторизирани потребители
 Route::group(['middleware' => ['auth']], function () {
     Route::get('contact', 'ContactsController@index')->name('contact');
     Route::post('contact', 'ContactsController@sendMail');
      //пътища достъпни за администратр
     Route::group(['prefix'=>'admin', 'middleware' => ['admin']], function(){
       Route::get('/', 'PagesController@adminIndex')->name('admin');
       ……...
       Route::resource('users', 'UsersController');
     });
});
```

Grouping Routes

Вариант на синтаксис за групиране

```
Route::group([
    'prefix'=>'/admin',
    'middleware' => 'auth'
    ], function(){
      Route::get('/', [
                'uses'    =>'AdminController@getIndex',
                'as'    =>'admin.index'
                ]);
      Route::get('blog/posts/create', [
                'uses'    => 'PostController@getCreatePost',
                'as'    => 'admin.blog.create_post'
                ]);
……
});
```

Задача 2

1. Ограничете достъпа до администраторските страници чрез групиране на пътищата и задаване на съотвения middleware guard.
2. Групирайте пътищата според представката в пътя. /admin или student/