web разработка

Controllers
Views
Routes

Съдържание

Предназначение, създаване и връзка между
- Controllers
- Views
- Routes

Plain controllers

Blade engine

Master.blade.php

@include

@extends

@section(‘section_name’)

@yield

Задача

Създайте основните страници във вашето приложение и организирайте достъпа до тях през началната страница

- Начална - Регистрация/логин
  - Профилна
  - домашни

Въведение

Въведение

Преди да правим каквито и да е промени в Ларавел

на адрес domain.name, когато приложението е онлайн

Или localhost/път-до-приложението/public се зарежда началната страница на Ларавел

Какъв е механизмът, който ползва Ларавел за да зареди тази страница?

Въведение

- app/routes/web.php - route декларация, задължителна за Ларавел, за да зареди

```
Route::get('/', function () {
        return view('welcome');
});
```

- app/resources/views/welcome.blade.php - view файл - това, което потребителя ще види на екрана.

Въведение

Обърнете внимание

```
return view('welcome');
```

сочи към

welcome.blade.php във папка resources/view

Blade engine

Blade engine

Blade е прост, но мощен генератор на съдържание / templating engine/ съпътстващ Laravel.

Blade не ви ограничава да пишете PHP код във вютата - може да смесвате типичния за blade синтаксис и php.

Blade файлове се компилират в php код и се кешират, докато не бъдат променени.

Blade файловете използват .blade.php разширение и се съхраняват в resources/views директорията.

PHP Artisan

PHP Artisan

Artisan е CLI/command-line interface/, включен в Ларавел или езика, който ще ползваме за да работим през конзолата с Ларавел.

Пълен списък на artisan командите -

php artisan list

Списък на команди от определена категория /migrate/

php artisan help migrate

Най-често ще извиквате - php artisan route:list

Контролери

Контролери

Контролерите групират методите, отговарящи за поведението на приложението в отделни класове.

Отговарят за извикането и отговора на потребителските заявки.

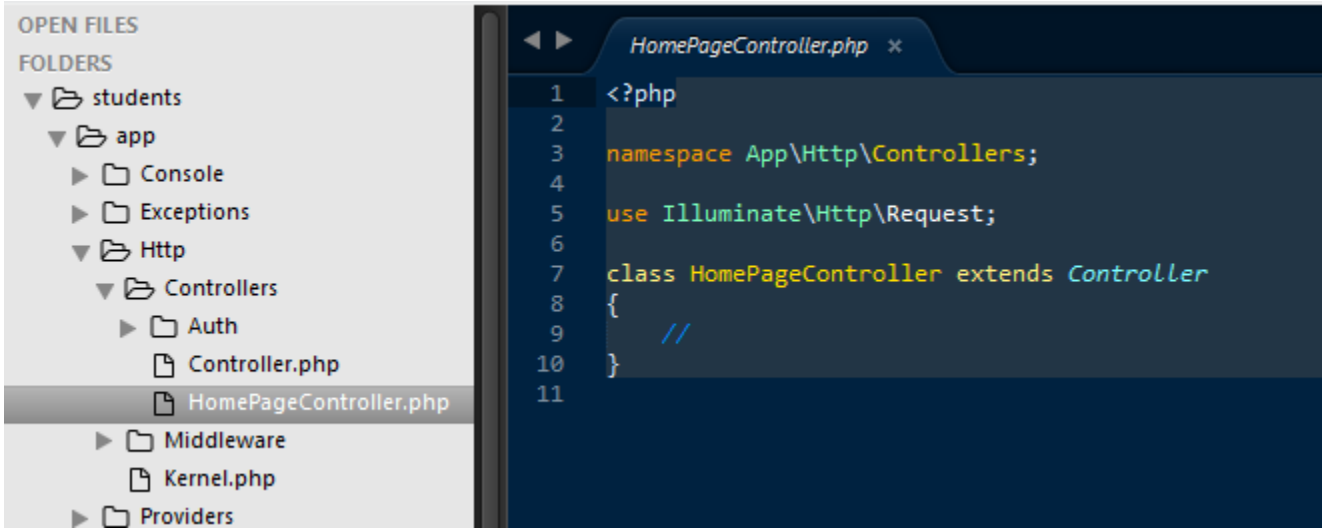
Съхраняват се в app/Http/Controllers directory.

Контролери

Създаваме т.нар. plain controller с командата

php artisan make:controller HomePageController

В резултат -



Контролери

Първата задача, която ще поверим на новосъздадения контролер е да зарежда началната страница - Home Page, при въвеждане на адреса в браузъра.

Добавяме

```
public function index() {

        return view('home_page');

}
```

Views

Views

В app/resources/views създаваме

home_page.blade.php

В този файл ще бъде разположено съдържанието, което ще вижда потребителя.

Routes

Routes

За да изпълни задачата си контролера - да зареди home page - поставена в index метода, трябва да регистрираме пътя, който съответства на тази задача.

В **routes/web.php**

Route::get('/', 'HomePageController@index');

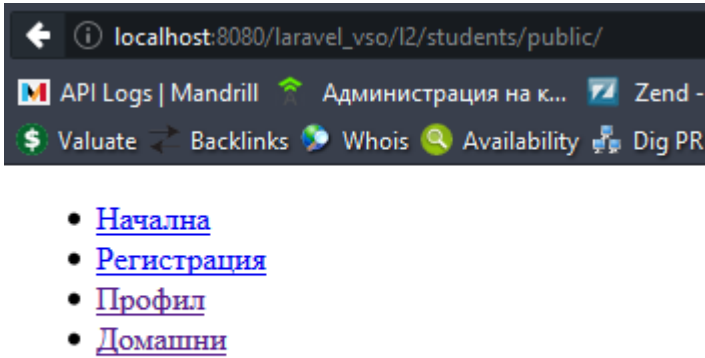
Декларираме, че при въвеждане на ...students/ в браузъра, ще бъде изпълнен методът index от контролера HomePage /резултатът от изпълнението на този метод е зареждането на Home Page в браузъра/

Routes

Задача Повторете стъпките, които направихме

за началната страница,

за всяка от другите страници в приложението.



Как да заработят линковете - да зареждат съответните страници, без да се налага ние да въвеждаме пътя в браузъра?

Routes - route()

За да заработи менюто на началната страница, посочваме пътищата към съответните страници като използваме функцията route().

Като параметър подаваме името на пътя /route/.

Routes - route()

Има много варианти за посочване на пътя към съответната страница.

Когато пътищата в route/web.php се натрупат е добре да дадем на всеки уникално име -

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

Route::get('profile', 'ProfileController@get_profile')->name('profile');

Route::get('homework', 'HomeworkController@index')->name('homework');
```

Routes - route()

HomePageController.php × home_page.blade.php ×

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Students</title>
6  </head>
7  <body>
8      <ul>
9          <li><a href="#">Начална</a></li>
10         <li><a href="#">Регистрация</a></li>
11         <li><a href="">Профил</a></li>
12         <li><a href="">Домашни</a></li>
13     </ul>
14 </body>
15 </html>
```

```
</head>
<body>
    <ul>
        <li><a href="{{ route('home') }}">Начална</a></li>
        <li><a href="#">Регистрация</a></li>
        <li><a href="{{ route('profile') }}">Профил</a></li>
        <li><a href="{{ route('homework') }}">Домашни</a></li>
    </ul>
</body>
</html>
```

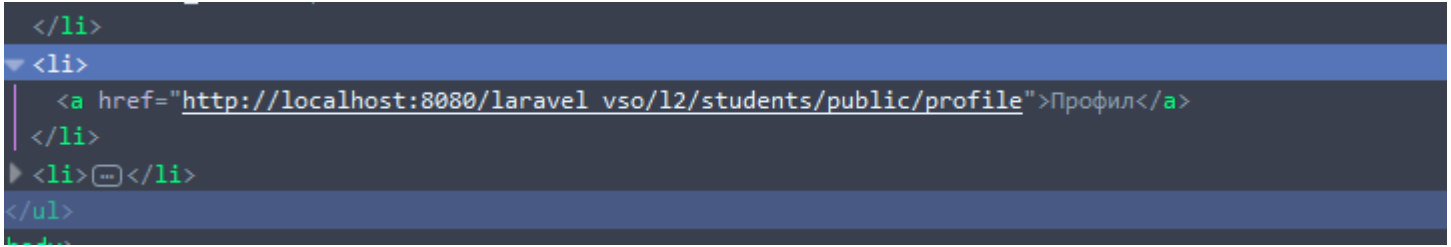
Routes - route()

Благодарение на Blade

Вместо **<?php echo route('home') ?>**

Посочваме пътя с **{{ route('home') }}**

В браузъра - резултатът от функцията route()



master.blade.php

master.blade.php

Файл, съдържащ повтарящата се във всички/група от вюта информация.

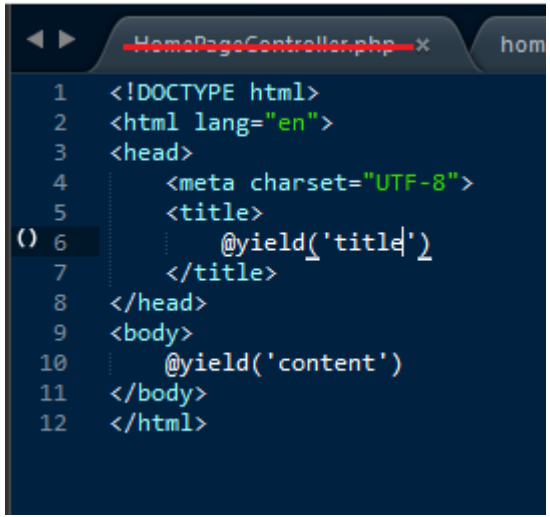
Макет, в който чрез

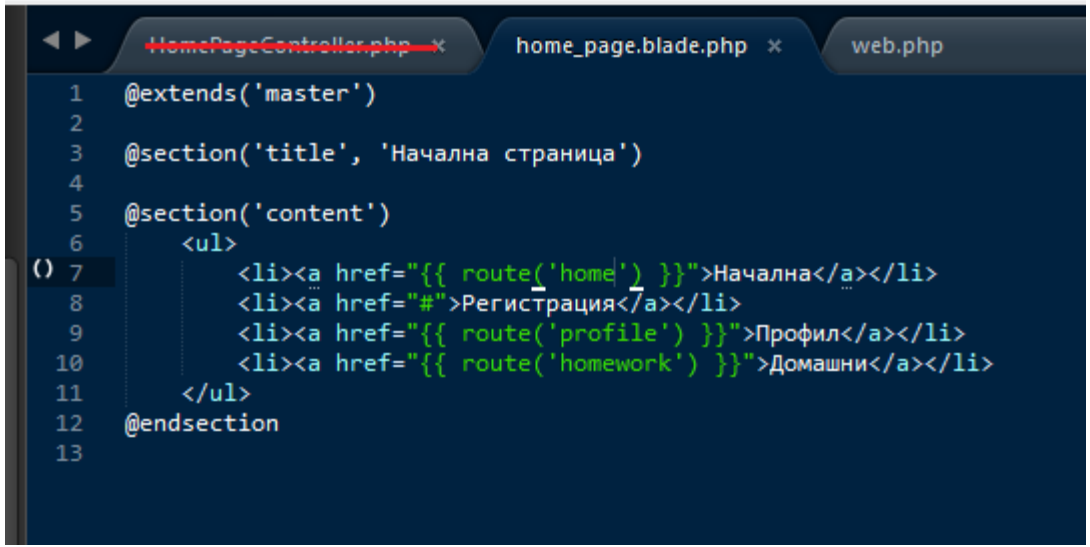
@yield('section_name')

Включваме различната информация за всяко вю, което наследява макета.

Различната информация може да бъде - стилови файлове, джаваскрипт, html и т.н

master.blade.php





Master.blade.php

home_page.blade.php

Организация на blade файловете

Организация на blade файловете

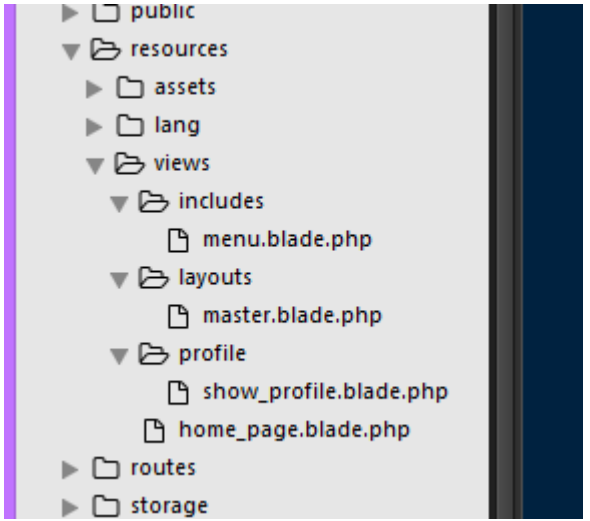
Най-вероятно файловете в проекта ни ще се натрупат.

Добре е да ги организираме в папки.

Когато blade файла е в папка се извиква

@extends(‘layouts.master’)
@extends(‘foldername.filename’)

view(‘profile.show_profile’)
view(‘foldername.filename’)



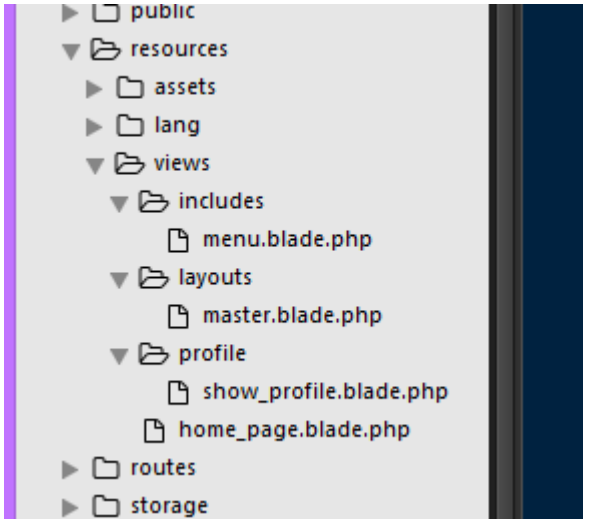
Организация на blade файловете

В организацията на папките обикновено присъства и папка includes.

В нея съхраняваме blade файлове, които включваме в други blade файлове чрез

@include('includes.menu')

/включваме menu.blade.php, намиращ се в папка includes/



Това са файлове, съдържащи информация, еднаква за група други файлове - меню, футър.