

Роут/роутинг (маршрут/маршрутизация)

Теория

Каждая операция, производимая сайтом (вывод страницы, сохранение введенных данных в базе и пр.), выполняется при получении им от веб-обозревателя клиентского запроса по определенному пути, выполненного с применением определенного HTTP-метода (GET, POST, PATCH и др.).

Путь — это часть интернет-адреса, находящаяся между адресом хоста и набором GET-параметров и идентифицирующая запрашиваемую страницу (например, интернет-адрес <http://localhost:8000/items/34?from=index> содержит путь `items/34`).

Следовательно, чтобы какое-либо действие контроллера выполнилось при получении запроса по определенному пути, выполненного определенным HTTP-методом, его следует связать с этими путем и методом, создав маршрут.

Маршрут Laravel — это объект особого класса, содержащий следующие сведения:

- шаблонный путь - задает нужный формат путей;
- допустимый HTTP-метод - которым должен быть выполнен клиентский запрос;
- действие контроллера - выполняется при совпадении шаблонного пути и допустимого метода с путем и методом, извлеченными из запроса (т. е. если маршрут является совпавшим).

В качестве примера рассмотрим следующие маршруты (записаны в формате «шаблонный путь - допустимый метод - выполняемая операция»):

- `/` (прямой слеш - «корень» сайта) - GET - вывод перечня объявлений;
- `/<ключ объявления>/` - GET - вывод объявления с заданным ключом;
- `/add/` - GET - вывод страницы для добавления объявления;
- `/` - POST - сохранение добавленного объявления в базе.

Созданные маршруты записываются в один из двух списков:

- список *веб-маршрутов* - содержит список маршрутов, ведущих на действия контроллеров, которые выдают обычные веб-страницы. Хранится в модуле `routes/web.php`;
- список *API-маршрутов* - содержит список маршрутов, ведущих на действия контроллеров, которые выдают данные в формате JSON. Хранится в модуле `routes/api.php`.

Просмотр одного из списков маршрутов, в зависимости от типа полученного запроса, в поисках совпавшего выполняет подсистема фреймворка, называемая маршрутизатором. Если ни один маршрут не совпал, выводится страница с сообщением об ошибке 404 (запрашиваемый путь не существует).

Практика

1. Чтобы обработать какой-то запрос, приложению нужно прописать маршруты, которые располагаются в файле `routes/web.php`. Поменяем код в файле.

```
kebagina_laravel > kebagina > routes > web.php > ...
15
16 Route::get('/', function () {
17     return 'Я начинаю изучать фреймворк';
18 });
19
```

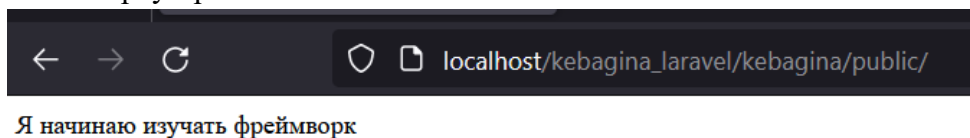
Класс `Route` — это фасад маршрутизатора (*фасадом* называется класс, служащий своего рода «пультом управления» одной из подсистем фреймворка).

Статический метод `get()`, вызванный у этого фасада, указывает маршрутизатору создать новый объект маршрута, связывающий допустимый HTTP-метод GET (одноименный методу фасада), шаблонный путь из первого параметра (у нас `- /`, «корень» сайта) и контроллер-функцию, заданную вторым параметром.

Статический метод `get()` класса `Route` принимает 2 аргумента:

- 1) URL который введён в браузер
- 2) Контент, который может быть передан двумя способами:
 - a. функцию, которая должна вернуть какие-то данные, либо представление (view)
 - b. строку в формате `controller@action` с указанием контроллера и метода, который должен быть вызван

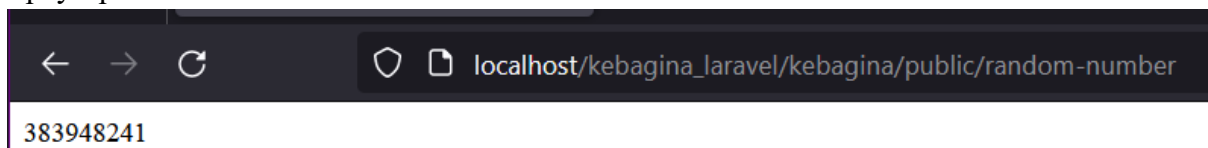
Итог в браузере:



2. Создадим второй маршрут: по запросу `/random-number` возвращаем случайное число

```
kebagina_laravel > kebagina > routes > web.php > ...
19
20 Route::get('/random-number', function () {
21     return rand();
22 });
23
```

Браузер:



В пунктах 1 и 2 используется маршрут в котором контент возвращается с помощью анонимной функции `function() { ... }`.

3. Теперь создадим маршрут для нашего контроллера `RandomController` (был создан в ч.1). в маршруте указывается имя контроллера и имя действия (в данном случае `generate`).

```
kebagina_laravel > kebagina > routes > web.php > ...
23
24 Route::get('/random-number2', [RandomController::class, 'generate']);
25
```


Такой подход позволяет разделять логику по контроллерам, а не писать весь функционал в одном файле. Файл с роутингом должен отвечать за адресацию, а не содержать весь код в одном месте.

Критически важно! Используемый контроллер необходимо подключить к файлу web.php. для этого выполняем приведённые ниже действия

Чтобы использовать такую запись выполните последовательно:

1. напишите код в указанном выше виде
2. затем поставьте курсор на окончание слова RandomController, нажмите на клавиши Ctrl + I (Ctrl + Alt + I) и затем выберите то, что необходимо (в данном случае контроллер)

```
[RandomController::class, 'generate']]);
```

The screenshot shows a code completion dropdown menu. The top item is 'RandomController' with a small icon to its left. Below it, a snippet of code is visible: 'use App\Http\Controllers\RandomContro...'.

3. Посмотрите на строки сверху – в них появилось подключение контроллера.

```
kebagina_laravel > kebagina > routes > web.php > ...
1  <?php
2
3  use App\Http\Controllers\RandomController;
4  use Illuminate\Support\Facades\Route;
```

Итоговый код в файле web.php

```
kebagina_laravel > kebagina > routes > web.php > ...
1  <?php
2
3  use App\Http\Controllers\RandomController;
4  use Illuminate\Support\Facades\Route;
5
6  /*
7  |-----
8  | Web Routes
9  |-----
10 |
11 | Here is where you can register web routes for your application. These
12 | routes are loaded by the RouteServiceProvider within a group which
13 | contains the "web" middleware group. Now create something great!
14 |
15 */
16
17 Route::get('/', function () {
18     return 'Я начинаю изучать фреймворк';
19 });
20
21 Route::get('/random-number', function () {
22     return rand();
23 });
24
25 Route::get('/random-number2', [RandomController::class, 'generate']);
26
```

Удобный способ отслеживания маршрутов

Для того чтобы наглядно посмотреть на маршрутизацию приложения можно воспользоваться еще одной командой в консоли:

```
php artisan route:list
```

На изображении ниже показан лист адресации, который у нас получился. Найдите те маршруты, которые мы с вами использовали.

В таком формате очень удобно следить за маршрутами приложения, т.к. сразу виден сам url, метод запроса, какой контроллер и какое действие вызывает данный роут, а также какие слои проверки (middleware) проходит маршрут.

```
kebagina@INT-A209-3 W:\domains\localhost\kebagina_laravel\kebagina
$ php artisan route:list

GET|HEAD / .....
POST _ignition/execute-solution ignition.executeSolution › Spatie\...
GET|HEAD _ignition/health-check ignition.healthCheck › Spatie\LaravelI...
POST _ignition/update-config ignition.updateConfig › Spatie\Larave...
GET|HEAD api/user .....
GET|HEAD random-number .....
GET|HEAD random-number2 ..... RandomController@generate
GET|HEAD sanctum/csrf-cookie sanctum.csrf-cookie › Laravel\Sanctum › C...

Showing [8] routes
```