**Route – маршрут**

(Routing - маршрутизация, middleware - посредники)

**- Маршрут** это объект, связывающий заданные шаблонный путь и допустимый НТТР-метод

с определенным действием определенного контроллера.

**- Маршрутизация** - процесс выяснения, какое действие какого контроллера следует

выполнить, на основании пути и НТТР-метода, извлеченных из клиентского запроса.

**- Настройки маршрутизатора:**

**App\Providers\RouteServiceProvider** - настройки маршрутизатора

**Hомe** - константа, задающая путь для перенаправления после авторизации, по

умолчанию '/home' - раздела пользователя);

**App\Http\Kernel** - настройки посредников

$**middleware** = [\App\Http\Middleware\TrustProxies::class, ... ];

массив посредников, через которые проходят все маршруты.

$**middlewareGroups** = ['**web**' => [\App\Http\Middleware\EncryptCookies::class, ...],'**api**' => [...]];

групповые массивы посредников, через которые проходят все маршруты этих групп.

$**routeMiddleware** = ['**auth**' => \App\Http\Middleware\Authenticate::class, ... ];

имхо, присвоение псевдонима посреднику, по которому можно обратится к посреднику.

**- Списки маршрутов:**

**api.php** - API маршруты, ведущие к действию контроллеров, выдающих данные в

формате JSON.

**web.php** - маршруты, ведущие к действию контроллеров, выдающих веб-страницы.

**- Созданием объектов маршрутов** Illuminate\Routing\Route занимается подсистема

маршрутизатора управляемая посредством фасада Illuminate\Support\Facades\Route.

**-Методы фасада Route:** get(), post(), put(), patch(), delete(), options(), match(), any(), view(), redirect(), permanentRedirect()

В маршруте, созданном методом get (), помимо НТТР-метода GET, записан еще и метод HEAD.

**- Виды записи маршрутов**(1.4)**:**

**с контроллером-классом**

**use** App\Http\Controllers\NameController;

Route::get('/', [NameController::class, 'methodName']);

**с контроллером-функцией**

Route::get('/', function(){

return view('viewName');

});

**с контроллером одного действия**

Route::get('/', OneActionController::class);

В контроллере одного действия обязательно должен быть магический метод

public function \_\_invoke(){}

**- Создание простых маршрутов**(8.3)**:**

Так же смотри "Виды записи маршрутов".

match() - несколько допустимых методов;

Route::match(['PUT', 'PATCH'], '/update', [NameController::class, 'updateMethodName']);

any() - примет любой метод

Route::any('/', [NameController::class, 'methodName']);

**- Специализированные маршруты**(8.3.1)**:**

view() - принимает методы GET или HEAD, возвращает view без участия контроллера

Route::view('/', 'viewName');

redirect() - принимает любой метод и перенаправит на указанный путь

Route::redirect('/', '/redirectingRoute');

Route::permanentRedirect('/', '/redirectingRoute');

**- Резервный маршрут**(8.3.2)**:**

**Вызов этого мотода должен быть в самом коце списка маршрутов!!!**

fallback() - если вместо стандартной "404 страница не найдена" нужно создать свою

Route::fallback([NameController::class, 'methodName']);

**- Именованные маршруты**(8.4)**:**

чтобы дать маршруту уникальное имя, нужно у объекта Route вызвать метод name()

Route::get('/', [NameController::class, 'methodName'])->name('uniqueRouteName');

На html странице можно автоматически генерировать соответствующие именованным маршрутам интернет-адреса, вызывая функцию route () с указанием имени маршрута

<а href="{{ route('uniqueRouteName') }}">Some Text</а>

**- URL-параметры и параметризованные маршруты**(8.5)**:**

В **интернет-адрес**, ведущий на какое-либо действие контроллера, можно поместить данные, необходимые этому действию для работы(/urlOptions - путь к странице):

http:/localhost:8000/urlOptions/firstOption/secondOption/thirdOption

**маршрут:**

Route::get('/urlOptions/{firstOption}/{secondOption}/{thirdOption?}', [HomeController::class, 'urlOptions']);

**метод контроллера:**

public function urlOptions($firstOption, $anotherNameForSecondOption, $thirdOption = ''){

$context = ['urlOptions' => [$firstOption, $anotherNameForSecondOption, $thirdOption]];

return view('urlOptionsPage', $context);

}

**на странице:**

{{ $urlOptions[0] }}, {{ $urlOptions[1] }}, {{ $urlOptions[2] }}

Первые два параметра обязательны к заполнению. Третий не обязателен: в маршруте он заканчивается ?, в методе приравнивается к пустой строке (равно двум одинарным ковычкам). В методе параметрам можно дать другие имена, сохраняется только очередность.

**Важно:** под маршрут с URL-параметроми описанный выше подойдет любой адрес вида http:/localhost:8000/Something/Something /Something/ Something который может быть адресован другому маршруту. Поэтому маршрут с URL-параметроми как можно ниже в списке маршрутов.

Так-же URL-параметры могут находится в центре адреса, тогда маршрут будет выглядеть так:

Route::get('/home/{Option}/second/third', [HomeController::class, 'methodName']);

**- GET-параметры:**

GET-параметры видно в интернет адресе после знака ? И разделены они &

http://127.0.0.1:8000/GET-options**?**first=one**&**second=two

Обратится к ним на странице можно так:

{{ $\_GET['first'] }}, {{ $\_GET['second'] }}

**- Шаблоны URL-параметров**(8.5.1)**:**

**У одного маршрута**

Route::get('/patternURLoption/{id}', [...])->where('id', '[0-9]');

Route::get('/patternURLoption/{first}/{second}', [...]) -> where(['first' => '[0-9]', 'second' => '[10-20]');

**У всех маршрутов** - у провайдера RouteServiceProvider необходимо переопределить

родительский метод boot (см. стр. 190).

**- Внедрение зависимостей** (смотри Controller)**:** URL-рараметры могут использоватся во

внедрении зависимостей реализующихся в коннтроллере.

**- Значение URL-параметров по умолчанию**(8.5.3)**:**

Запрос проходит через middleware, где не указанным параметрам присваеваются значения

по умолчанию. Можно назначить middleware группе маршрутов.

**- Дополнительные параметры маршрутов**(8.6)**:**

Дополнительные параметры задаются вызовом у объекта маршрута следующих методов

Route::get()->**middleware**();

Route::get()->**withoutmiddleware**();

Route::get()->**prefix**();

Route::get()->**domain**();

**- Группы маршрутов**(8.7)**:**

Группа маршрутов служит для объединения произвольного количества маршрутов с целью задать у них одинаковые параметры - префикс шаблонного пути или связанных посредников.

**Объединение анонимной функцией в списке маршрутов**

Route::middleware('auth')->group(function(){

Route::get('/home',[HomeController::class,'index']);

Route::get('/admin',[AdminController::class,'index']);

));

**Список сформированный в дупугом файле**

Route::middleware ('auth')->group(base\_path('routes/admin.php'));

Группе маршрутов можно отдельно задать Route::namespace('App\\Http\\Controllers')->g... и

префикс Route::name('private')->g..

**- Маршруты на ресурсные контроллеры**(8.8)**:**

Разные разделы сайта имеющие функционал CRUD реализуются похожими/одинаковыми маршрутами и методами контроллеров. Один маршрут с методом Route::**resource**('nameResource', nameController::class) имеет "под капотом" 7 именованных путей на которые могут указывать ссылки

+----------+----------------------+-----------------+--------+

| Method | URI | Name | Action |

+-----------+-------------------------+------------------+---------+

| GET|HEAD | / | | Closure |

| GET|HEAD | articles | articles.index | index |

| POST | articles | articles.store | store |

| GET|HEAD | articles/create | articles.create | create |

| GET|HEAD | articles/**{**article**}** | articles.show | show |

| PUT|PATCH | articles/**{**article**}** | articles.update | update |

| DELETE | articles/**{**article**}** | articles.destroy | destroy |

| GET|HEAD | articles/**{**article**}**/edit | articles.edit | edit |

+-----------+-------------------------+------------------+---------+

Ресурсный контроллер должен иметь 7 методов с именами указанными в каждом "пути" маршрута; создать такой контроллер можно автоматически командой php artisan make:controller NameController --resource.

Ресурсные маршруты можно групперовать методом **resources()**.

Еще методы apiResource(), apiResources()

**- Маршруты на подчиненные или вложенные ресурсные контроллеры**(8.8.1)**:**

Используются, если структура сайта содержит в разделе подразделы имеющие функционал

CRUD.

**- Дополнительные параметры маршрутов на ресурсные контроллеры**()**:**

**Route**::resource('nameResource', nameController::class)->**only**(['index', 'show']);

Создает маршрут только на перечисленные действия.

**Route**::resource('nameResource', nameController::class)->**except**(['destroy']);

Исключает из стандартного списка маршрутов перечисленные действия.

**Route**::resource('nameResource', nameController::class)->**names**(['destroy' => 'BBs.erase']);

Задает новые имена для маршрутов. Ключи элементов заданного ассоциативного массива должны соответствовать действиям контроллера, а значения элементов укажут имена для

соответствующих маршрутов.

**Route**::resource('rubrics.bbs', RuЬricBbController::class)

->**parameters**(['rubrics' => 'rubric\_key', 'bbs' => 'bb\_key']);

задает новые имена для URL-параметров. Ключи элементов заданного ассоциативного массива должны соответствовать именам ресурсов, а значения элементов укажут имена

для соответствующих URL-параметров.

**- Как Laravel обрабатывает списки маршрутов**(8.9)

Обработка и добавление нового файла списка маршрутов.