Публикация на тему

Маршрутизация Laravel

Маршрутизация запросов в php-фрэймворке Laravel

Автор
Михалькевич Александр Викторович

Публикация

Наименование Маршрутизация Laravel
Автор А.В.Михалькевич
Специальность Маршрутизация запросов в php-фрэймворке Laravel,
Анотация
Anotation in English
Ключевые слова
Количество символов 9486

Содержание

Введение

1 Первоначальная маршрутизация

Запрос из адресной строки попадает в так называемый обработчик маршрутов, или маршрутизатор, или роутер (routes). Маршрутизатор определяет, какой контроллер необходимо вызывать.

Маршруты определяются в файле routes/web.php

Простейший get-маршрут. Листинг 1.1

```
Route::get('/', function () {
return 'Hello World';
});
```

Таким образом маршрутизатор Laravel перехватывает запрос на главную страницу, и вызывает callback-функцию, которая выводит на экран текст.

Для перехвата POST-данных можно воспользоваться методом Route::post

Простейший post-маршрут. Листинг 1.2

```
Route::post('foo/bar', function () {
return 'Hello World';
});
```

Метод Route::any перехватывает и POSTи GET данные.

Маршрут любого http-запроса. Листинг 1.3

```
Route::any('foo', function () {
return 'Hello World';
});
```

Для перехвата маршрутов только по протоколу HTTPS, вторым входящим параметром можно передать не функцию, а массив, первым элементом которого является тип протокола, а вторым – функция.

Маршрут любого https-запроса. Листинг 1.4

```
Route::get('foo', array('https', function() {
return 'Must be over HTTPS';
}));
```

2 Параметры маршрутов

Добавление к маршруту обязательного параметра id:

Добавление к маршруту обязательного параметра id. Листинг 2.1

```
Route::get('user/{id}', function ($id) {
return 'User '.$id;
});
```

Добавление к маршруту необязательного параметра name.

Добавление к маршруту необязательного параметра name. Листинг 2.2

```
Route::get('user/{name?}', function ($name = null) {
return $name;
});
```

Вместо $name = null можно использовать любое значение по умолчанию.

Добавление к маршруту необязательного параметра name. Листинг 2.3

```
Route::get('user/{name?}', function ($name ='Jhon') {
return $name;
});
```

Использование регулярных выражений в маршрутах.

Маршруты с соответствием пути регулярному выражению. Листинг 2.4

```
Route::get('user/{name}', function ($name) {
})->where('name', '[A-Za-z]+');
Route::get('user/{id}', function ($id) {
})->where('id', '[0-9]+');
```

Вместо последовательного вызова метода where, можно передать массив ограничений.

Использование массива в регулярных выррaжениях. Листинг 2.5

```
Route::get('user/{id}/{name}', function ($name) {
})->where(['name'=>'[A-Za-z]+', 'id'=>'[0-9]+']);
```

Если какие-то регулярные выражения нужно связать со всеми параметрами, то можно использовать метод pattern:

Использование шаблона регулярного выражения. Листинг 2.6

```
public function boot(Router $router)
{
$router->pattern('id', '[0-9]+');

parent::boot($router);
}
```

Таким образом, будет осуществляться проверка всех параметров с именем id.

3 Именнованные маршруты

Задать имя маршруту можно следующим способом:

Назначение имени текущего исполняемого маршрута. Листинг 3.1

```
Route::get('user/profile', array('as' => 'profile', function () {
//
}));
```

Также можно задать контроллер и его экшн, который будет выполняется по данному маршруту.

Назначение контроллера и экшна. Листинг 3.2

```
Route::get('user/profile', array('as' => 'profile',
'uses' => 'UserController@showProfile'));
```

Теперь можно использовать имя маршрута при генерации URL либо при перенаправлении.

Генерация URL. Листинг 3.3

```
$url = URL::route('profile');
$redirect = Redirect::route('profile');
$url = URL::to('foo');
```

Получить имя текущего выполняемого маршрута можно методом currentRouteName():

Получить имя текущего исполняемого маршрута. Листинг 3.4

```
$name = Route::currentRouteName();
```

Заключение

Список использованных источников

Приложения