

Finishing touch

Before we end this section there's a couple of things that we need to do.

1. Refactor `categories` route
2. Moving entire routes from `web.php` into `api.php`
3. Remove unused files

1. Refactor `categories` route

Let's move the logic inside `categories` route declaration from `web.php` into a dedicated controller. Let's open up our terminal and type:

```
php artisan make:controller CategoryController -i
```

Note that I added `-i` option in the command above. It's a shorthand option of `--invokable` which tell *artisan* to generate an invokable controller class. If we open that file it would be like this:

```
class CategoryController extends Controller
{
    /**
     * Handle the incoming request.
     *
     * @param \Illuminate\Http\Request $request
     * @return \Illuminate\Http\Response
     */
    public function __invoke(Request $request)
    {
        //
    }
}
```

Now our `categories` route declaration will look as follow:

```
Route::get('/categories', 'CategoryController');
```

Note that we don't need to specify the action/method name in the second argument of `get` method since we've defined the `__invoke` method.

Our `CategoryController` will look like this:

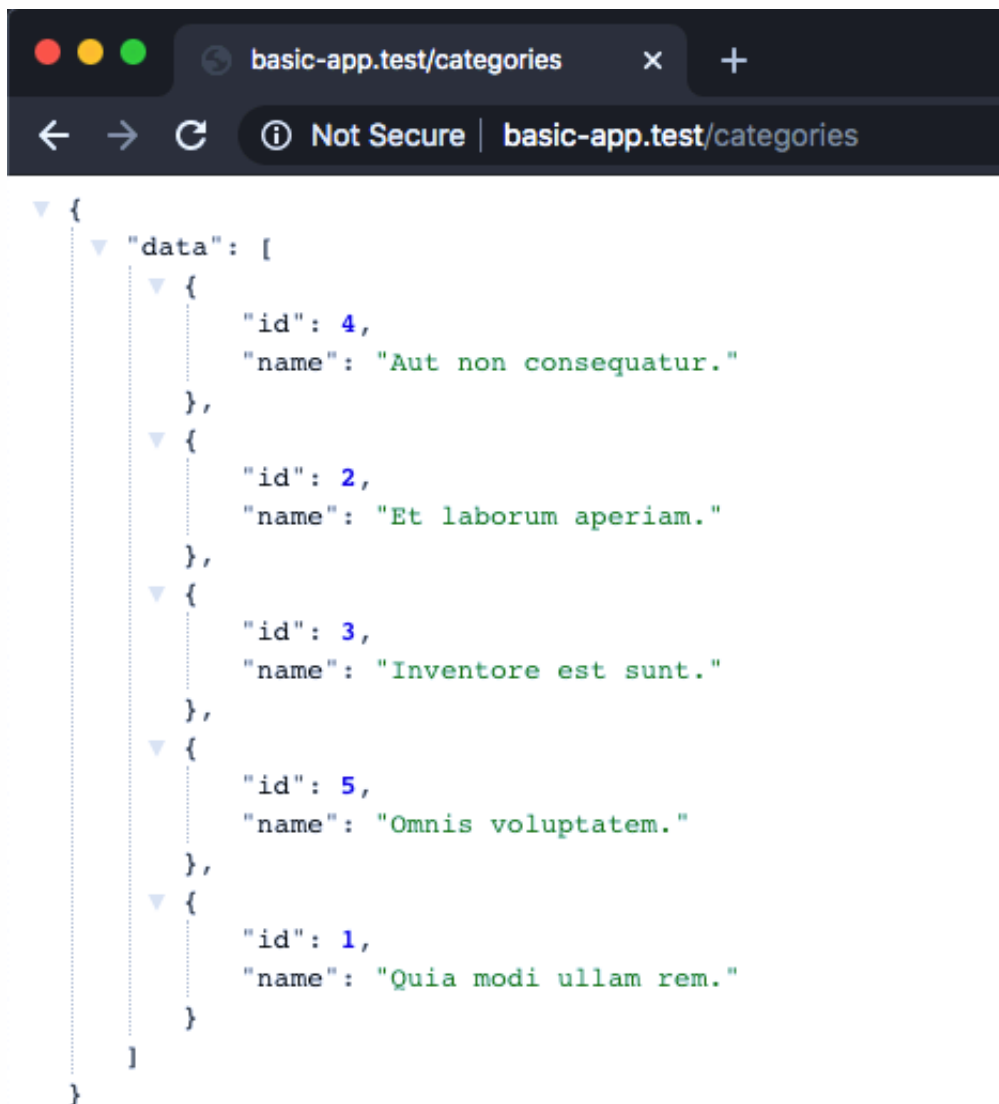
```

use App\Category;
use Illuminate\Http\Request;
use App\Http\Resources\CategoryResource;

class CategoryController extends Controller
{
    public function __invoke(Request $request)
    {
        $categories = Category::orderBy('name')->get();
        return CategoryResource::collection($categories);
    }
}

```

If we save the change and visit the categories URI in our browser or in Postman, we'll get the exact same thing.



```

{
  "data": [
    {
      "id": 4,
      "name": "Aut non consequatur."
    },
    {
      "id": 2,
      "name": "Et laborum aperiam."
    },
    {
      "id": 3,
      "name": "Inventore est sunt."
    },
    {
      "id": 5,
      "name": "Omnis voluptatem."
    },
    {
      "id": 1,
      "name": "Quia modi ullam rem."
    }
  ]
}

```

2. Moving entire routes to api.php

As mentioned in the previous lessons that defining api routes would be better defined in `api.php`. So let's move entire routes except the root route from `web.php` into `api.php` file.

```

web.php x  ...  api.php x
routes > web.php
17 /
18 */
19
20 Route::get('/', function () {
21     return view('welcome');
22 });
23
api.php
16
17 Route::middleware('auth:api')->get('/user', function (Request $request) {
18     return $request->user();
19 });
20
21 Route::get('/products', 'ProductController@index');
22 Route::post('/products', 'ProductController@store');
23 Route::get('/products/{product}', 'ProductController@show');
24 Route::put('/products/{product}', 'ProductController@update');
25 Route::delete('/products/{product}', 'ProductController@destroy');
26
27 Route::get('/categories', 'CategoryController');

```

Now if we inspect our routes in our terminal through `php artisan route:list` command.

```

basic-app (finishing-touch) x php artisan route:list

```

Domain	Method	URI	Name	Action	Middleware
	GET HEAD	/		Closure	web
	GET HEAD	api/categories		App\Http\Controllers\CategoryController	api
	GET HEAD	api/products		App\Http\Controllers\ProductController@index	api
	POST	api/products		App\Http\Controllers\ProductController@store	api
	GET HEAD	api/products/{product}		App\Http\Controllers\ProductController@show	api
	PUT	api/products/{product}		App\Http\Controllers\ProductController@update	api
	DELETE	api/products/{product}		App\Http\Controllers\ProductController@destroy	api
	GET HEAD	api/user		Closure	api, auth:api

```

basic-app (finishing-touch) x

```

We'll see that all URI of our routes except the root route are prefixed with `/api`.

Also their middleware are now using `api` so that we don't need to worry about csrf token issue when sending POST, PUT or PATCH request.

For the products related routes we can simplified them by replacing them with `apiResource` like so:

```

21 // Route::get('/products', 'ProductController@index');
22 // Route::post('/products', 'ProductController@store');
23 // Route::get('/products/{product}', 'ProductController@show');
24 // Route::put('/products/{product}', 'ProductController@update');
25 // Route::delete('/products/{product}', 'ProductController@destroy');
26 Route::apiResource('/products', 'ProductController');

```

If you inspect your routes, you'll get the exact same thing. The only difference is that they now have names which I think not really useful for external API use.

```
→ basic-app (finishing-touch) x php artisan route:list
```

Domain	Method	URI	Name	Action	Middleware
	GET HEAD	/		Closure	web
	GET HEAD	api/categories		App\Http\Controllers\CategoryController	api
	GET HEAD	api/products	products.index	App\Http\Controllers\ProductController@index	api
	POST	api/products	products.store	App\Http\Controllers\ProductController@store	api
	GET HEAD	api/products/{product}	products.show	App\Http\Controllers\ProductController@show	api
	PUT PATCH	api/products/{product}	products.update	App\Http\Controllers\ProductController@update	api
	DELETE	api/products/{product}	products.destroy	App\Http\Controllers\ProductController@destroy	api
	GET HEAD	api/user		Closure	api auth:api

3. Remove unused files

Last, since we're building API for external use (we'll use this for our Products Management App), we no longer need any views inside products folder. So you might get rid of that folder.

CORS Stuff

Since we're using Laravel version 7 we almost don't need to setup CORS stuff in our app. In the prior version of Laravel we need to install [laravel-cors](#) package. But we don't need that because Laravel 7 has included that package by default.

You can see more information about CORS by hitting this link: [Mozilla CORS documentation](#).

If you need to make some change, you can modify `cors.php` file in `config` folder.

```
<?php
return [
    'paths' => ['api/*'],
    'allowed_methods' => ['*'],
    'allowed_origins' => ['*'],
    'allowed_origins_patterns' => [],
    'allowed_headers' => ['*'],
    'exposed_headers' => [],
    'max_age' => 0,
    'supports_credentials' => false,
];
```