

bootstrapdojo.com

REST API Laravel 5.4 with Token Authentication

by Perry Rico

4-5 minutes

08 Mar REST API Laravel 5.4 with Token Authentication

Posted at 16:06h in [Blog](#) [0 Comments](#)

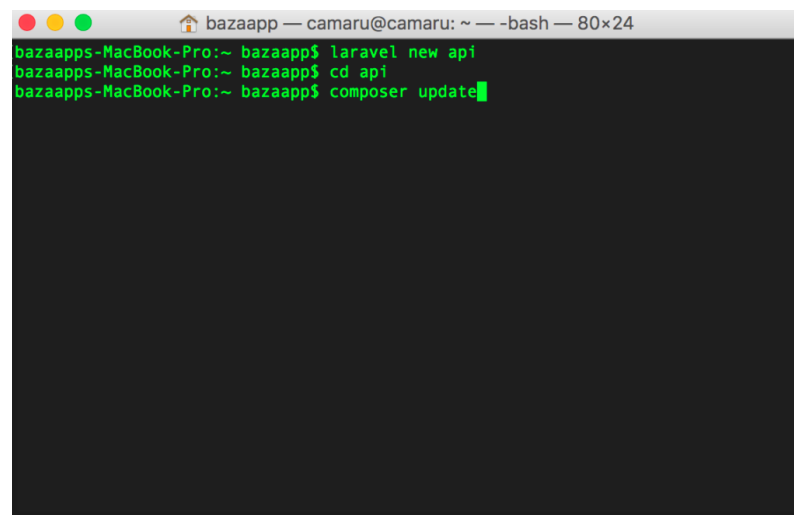
For this tutorial, we are going to do step by step creation of Laravel REST API with token authentication. The API will be a simple blog post API with users relation.

1. Setup Database and Project

1.1 Create a laravel project

We are going to name it as “api” project. Run the following commands in terminal.

```
# laravel new api
# cd api
# composer update
```

A screenshot of a terminal window on a Mac. The window title is 'bazaapp — camaru@camaru: ~ — -bash — 80x24'. The terminal shows three commands being executed: 'laravel new api', 'cd api', and 'composer update'. The output of these commands is not visible, only the prompts and the commands themselves. The terminal background is dark, and the text is green.

```
bazaapps-MacBook-Pro:~ bazaapp$ laravel new api
bazaapps-MacBook-Pro:~ bazaapp$ cd api
bazaapps-MacBook-Pro:~ bazaapp$ composer update
```

1.2 Create database in mysql and setup laravel environment

First, let us create a database. Run the following commands in terminal

```
# mysql -u <user> -p
```

```
create database api
exit
```



```
bazaapp — camaru@camaru: ~ — ssh camaru@192.168.0.101 — 80x24
camaru@camaru:~$ mysql -u truffles -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 133
Server version: 5.7.17-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database api
```

Next would be editing the .env file in your laravel project

```
DB_CONNECTION=mysql
DB_HOST=replace with your IP
DB_PORT=3306
DB_DATABASE=api
DB_USERNAME=replace with your username
DB_PASSWORD=replace with your password
```

2. Create the necessary database tables

2.1 Update Users table

Let us first update the users migration table. My filename is 2014_10_12_000000_create_users_table.php please check your own. Change the entire function with the code below.

```
public function up()
{
    Schema::create('users', function (Blueprint
$table) {
        $table->increments('id');
        $table->string('name');
        $table->string('email')->unique();
        $table->string('password');
        $table->string('api_token', 60)->unique();
        $table->rememberToken();
        $table->timestamps();
    });
}
```

2.2 Create the necessary model and relation to user

Run command below in terminal.

```
# php artisan make:model Post -m
```

Update **Post.php** model by adding a foreign key relation to Users

```
public function user() {  
  
    return $this->belongsTo(User::class);  
}
```

Update **Users.php** model and add parent key relation

```
public function posts() {  
    return $this->hasMany(Post::class);  
}
```

Update the **post migration file**, in my case the file is named 2017_03_08_135051_create_posts_table.php. We need to add two custom fields in our post table for demo purposes.

```
public function up()  
{  
    Schema::create('posts', function (Blueprint  
$table) {  
        $table->increments('id');  
        $table->string('title');  
        $table->text('body');  
        $table->integer('user_id');  
        $table->timestamps();  
    });  
}
```

Then update the database by running the command below.

```
# php artisan migrate:refresh
```

3. Create the Controller for API

For this example, we will name our controller as APIController. To create the controller run the command below in terminal.

```
# php artisan make:controller APIController  
--resource
```

3.1 Create the api that show all post of a given user

Update index function in APIController.php

```
*/  
public function index()  
{  
    $posts = Post::all();
```

```
return response()->json($posts);
}
```

Do not forget to include the Post model in your API controller

```
use App\Post;
```

4. Seed the database using tinker

Note that seed generator will be discussed in separate topic and is out of scope in this current tutorial.

```
# php artisan tinker
```

4.1 Create a test user.

```
$user = new App\User();
$user->name = 'perry';
$user->email = 'perry@bootstrapdojo.com';
$user->password = bcrypt('123456');
$user->api_token = str_random(60);
$user->save();
```

4.2 Create a sample blog post

```
$post->title='test 02';
$post->body='test 02 body';
$post->user_id=1;
$post->save();
Close tinker by running the command
exit
```

5. Create API route

Note that we will secure this route later

Edit api.php

```
Route::group(['middleware' => 'auth:api'],
function () {
    Route::resource('post', 'APIController');
});
```

6. Enable Authentication

Run the command in terminal

```
# php artisan make:auth
```

7. Test your API

```
# php artisan serve
```

Run the API in browser. Make sure you pass your api_token. See sample URL call below:

127.0.0.1:8000/api

/post?api_token=Pwbo7CmLtqAP5TvIHfYZPqkmetJFTi9UZIfxE1GLEBFFJgzZ8D

