How to Create a REST API with Laravel Lumen

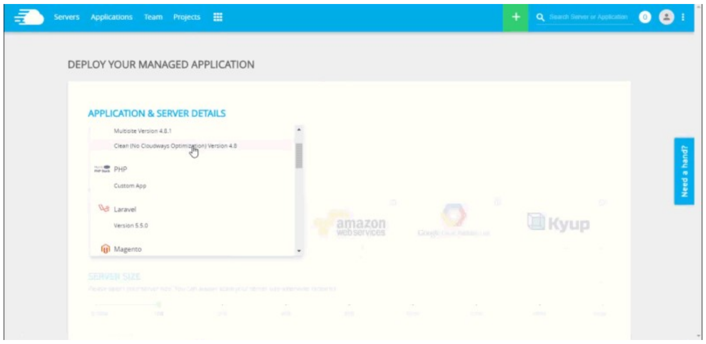
Updated on November 3, 2016 | 4 Min Read | API LARAVEL

Lumen is a micro-framework built on top of Laravel. The framework is ideal for small apps and services that are optimized for speed. The most important application of the framework is to build REST APIs.

Why build a REST API in Lumen

- Lumen is blazing fast.
- It can handle more requests per second than Laravel.
- It uses nikic/FastRoute instead of Symfony, thereby increasing performance.

To build a REST API with Lumen, signup to launch a server on Cloudways Laravel hosting. On Cloudways platform, you get Laravel in a 1-click install with an optimized stack and pre-installed Composer.



You might also like: Quickly Host Laravel on DigitalOcean in Few Clicks

Note: If you want to use the Laravel Stack

*Inside the Laravel **public_html** directory, run the following commands to create a Lumen app inside the Laravel project :*

```
> composer require "laravel/lumen"  //installing Lumen > vendor/bin/lumen new mylumen_car_app  //creating Lumen app
```

```
> composer require "laravel/lumen"  //installing Lumen
> vendor/bin/lumen new mylumen_car_app  //creating Lumen app
```

```
> composer require "laravel/lumen" //installing Lumen > vendor/bin/lumen new mylumen_car_app //creating Lumen app
```

From here you have 3 options to update the webroot:

Symlink your Lumen public directory to the Laravel public directory.

*OR update the webroot using the Cloudways **Application Settings** page.*

*OR move Lumen's **index.php** inside the Laravel public directory and*

*update the **bootstapp/app.php** path.*

Next, inside the app's **public_html** folder, run the following command to create a Lumen project.

```
> composer create-project laravel/lumen car_api
```

```
> composer create-project laravel/lumen car_api
```

```
> composer create-project laravel/lumen car_api
```

Change the webroot by adding **/public** (this is the public folder for all Laravel apps) in the Cloudways **Application Settings** page so that the new webroot is now **public_html/public**.

Load up the application URL in the browser and you will see the following page.

Lumen (5.3.0) (Laravel Components 5.3.*)

MySQL connection

I will use MySQL for this tutorial. Update the DB credentials in the **.env** file using the MySQL ACCESS available on the Cloudways **Application Page**. The following fields should be updated:

```
DB_DATABASE=<db_name> DB_USERNAME=<db_username> DB_PASSWORD=<db_password>
```

```
DB_DATABASE=<db_name>
DB_USERNAME=<db_username>
DB_PASSWORD=<db_password>
```

```
DB_DATABASE=<db_name> DB_USERNAME=<db_username> DB_PASSWORD=<db_password>
```

Next, uncomment the following lines in **bootstrap/app.php**

```
$app->withFacades(); $app->withEloquent();
```

```
$app->withFacades();
$app->withEloquent();
```

```
$app->withFacades(); $app->withEloquent();
```

The Facade class is a static interface to classes available in the application's service containers. This class is required to access certain core components of Lumen. Eloquent is the ORM that is used communicate with the MySQL database.

Migration

It is now time to create the database schema.

Create a table for **Cars** with four fields including the auto- increment **id**. The other three fields are **make**, **model**, and **year**. To create the table, run:

```
> php artisan make:migration create_table_cars --create=cars
```

```
> php artisan make:migration create_table_cars --create=cars
```

```
> php artisan make:migration create_table_cars --create=cars
```

This will create a **<date>_create_table_cars.php** file inside the **database/migrations/** folder. I will now edit this file and define the table.

Add the following code inside **up** function:

```
Schema::create('cars', function (Blueprint $table) { $table->increments('id'); $table->string('make'); $table->string('model'); $table->string('year'); });
```

```
Schema::create('cars', function (Blueprint $table) {
            $table->increments('id');
            $table->string('make');
            $table->string('model');
            $table->string('year');
        });
```

```
Schema::create('cars', function (Blueprint $table) { $table->increments('id'); $table->string('make'); $table->string('model'); $table->string('year'); });
```

The **up** function will be triggered when the table is actually created during the migration. The **down** function (no changes required in this function) will delete the table if the need arises.

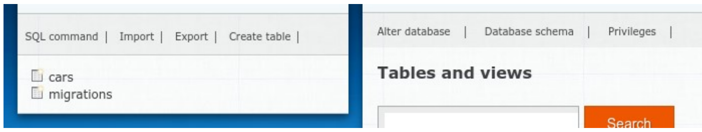
Now run the migration using:

```
> php artisan migrate Migration table created successfully. Migrated: 2016_09_08_142212_create_table_cars
```

```
> php artisan migrate
Migration table created successfully.
Migrated: 2016_09_08_142212_create_table_cars
```

```
> php artisan migrate Migration table created successfully. Migrated: 2016_09_08_142212_create_table_cars
```

At this point, the table has been created and can be viewed in MySQL manager.



The Model

Next step is the creation of the model. Create the **app/Car.php** file and add the following code:

```
<?php namespace App; use Illuminate\Database\Eloquent\Model; class Car extends Model { protected $fillable = ['make', 'model', 'year']; } ?>
```

```
<?php namespace App;

use Illuminate\Database\Eloquent\Model;

class Car extends Model
{
   protected $fillable = ['make', 'model', 'year'];
}
?>
```

```
<?php namespace App; use Illuminate\Database\Eloquent\Model; class Car extends Model { protected $fillable = ['make', 'model', 'year']; } ?>
```

The Controller

Create a controller inside the **app/Http/Controllers/CarController.php**.

```
<?php namespace App\Http\Controllers; use App\Car; use App\Http\Controllers\Controller; use Illuminate\Http\Request; class CarController extends Controller{ public function createCar(Request $request){ $car = Car::create($request->all()); return response()->json($car); } public function updateCar(Request $request, $id){ $car = Car::find($id); $car->make = $request->input('make'); $car->model = $request->input('model'); $car->year = $request->input('year'); $car->save(); return response()->json($car); } public function deleteCar($id){ $car = Car::find($id); $car->delete(); return response()->json('Removed successfully.'); } public function index(){ $cars = Car::all() return response()->json($cars); } } ?>
```

```
<?php

namespace App\Http\Controllers;

use App\Car;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;

class CarController extends Controller{

  public function createCar(Request $request){

      $car = Car::create($request->all());

      return response()->json($car);

  }

  public function updateCar(Request $request, $id){

      $car  = Car::find($id);
      $car->make = $request->input('make');
      $car->model = $request->input('model');
      $car->year = $request->input('year');
      $car->save();

      return response()->json($car);
  }

  public function deleteCar($id){
      $car  = Car::find($id);
      $car->delete();

      return response()->json('Removed successfully.');
  }

  public function index(){

```

```
        $cars  = Car::all();

        return response()->json($cars);

    }
}
?>
```

```
<?php namespace App\Http\Controllers; use App\Car; use App\Http\Controllers\Controller; use Illuminate\Http\Request; class CarController extends Controller{ public function createCar(Request $request){ $car = Car::create($request->all()); return response()->json($car); } public function updateCar(Request $request, $id){ $car = Car::find($id); $car->make = $request->input('make'); $car->model = $request->input('model'); $car->year = $request->input('year'); $car->save(); return response()->json($car); } public function deleteCar($id){ $car = Car::find($id); $car->delete(); return response()->json('Removed successfully.'); } public function index(){ $cars = Car::all(); return response()->json($cars); } } ?>
```

The routes

Now all that remains is the addition of the routes. I will write routes for creating, updating, deleting and viewing cars.

Open up **app/Http/routes.php** and add the following routes.

```
$app->group(['prefix' => 'api/v1','namespace' => 'App\Http\Controllers'], function($app) { $app->post('car','CarController@createCar'); $app->put('car/{id}','CarController@updateCar'); $app->delete('car/{id}','CarController@deleteCar'); $app->get('car','CarController@index'); });
```

```
$app->group(['prefix' => 'api/v1','namespace' => 'App\Http\Controllers'], function($app)
{
  $app->post('car','CarController@createCar');

  $app->put('car/{id}','CarController@updateCar');

  $app->delete('car/{id}','CarController@deleteCar');

  $app->get('car','CarController@index');
});
```

```
$app->group(['prefix' => 'api/v1','namespace' => 'App\Http\Controllers'], function($app)  { $app->post('car','CarController@createCar'); $app->put('car/{id}','CarController@updateCar'); $app->delete('car/{id}','CarController@deleteCar'); $app->get('car','CarController@index'); });
```

Notice that I have grouped the routes together into a common **api/v1** route.

Testing It Out

Now, test it all out using **curl**.

Creation

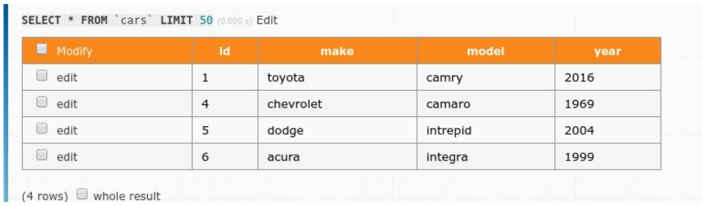
```
> curl -i -X POST -H "Content-Type:application/json" http://<unique-to-you>.cloudwaysapps.com/api/v1/car -d '{"make":"toyota", "model":"camry", "year":"2016"}' {"make":"toyota","model":"camry","year":"2016","id":7}
```

```
> curl -i -X POST -H "Content-Type:application/json" http://<unique-to-you>.cloudwaysapps.com/api/v1/car -d '{"make":"toyota", "model":"camry", "year":"2016"}'

{"make":"toyota","model":"camry","year":"2016","id":7}
```

```
> curl -i -X POST -H "Content-Type:application/json" http://<unique-to-you>.cloudwaysapps.com/api/v1/car -d '{"make":"toyota", "model":"camry", "year":"2016"}' {"make":"toyota","model":"camry","year":"2016","id":7}
```

After adding several more records, the DB would look like:



```
SELECT * FROM `cars` LIMIT 50 (0.000 s) Edit
```

☐ Modify	id	make	model	year
☐ edit	1	toyota	camry	2016
☐ edit	4	chevrolet	camaro	1969
☐ edit	5	dodge	intrepid	2004
☐ edit	6	acura	integra	1999

(4 rows) ☐ whole result

Updating

Next,I will update **id = 1** to **Toyota Supra 1999** (the previous valve was **Toyota Camry 2016**).

```
> curl -H "Content-Type:application/json" -X PUT http://<unique-to-you>.cloudwaysapps.com/api/v1/car/1 -d '{"make":"toyota", "model":"supra", "year":"1999"}' {"id":1,"make":"toyota","model":"supra","year":"1999"}
```

```
> curl -H "Content-Type:application/json" -X PUT http://<unique-to-you>.cloudwaysapps.com/api/v1/car/1 -d '{"make":"toyota", "model":"supra", "year":"1999"}'

{"id":1,"make":"toyota","model":"supra","year":"1999"}
```

```
> curl -H "Content-Type:application/json" -X PUT http://<unique-to-you>.cloudwaysapps.com/api/v1/car/1 -d '{"make":"toyota", "model":"supra", "year":"1999"}' {"id":1,"make":"toyota","model":"supra","year":"1999"}
```

Deletion

Now, I will delete **id = 1**.

```
> curl -X DELETE http://<unique-to-you>.cloudwaysapps.com/api/v1/car/1
```

```
> curl -X DELETE http://<unique-to-you>.cloudwaysapps.com/api/v1/car/1
```

```
> curl -X DELETE http://<unique-to-you>.cloudwaysapps.com/api/v1/car/1
```

"Removed successfully."

View

This is what the final data should look like:

```
> curl http://<unique-to-you>.cloudwaysapps.com/api/v1/car [{"id":4,"make":"chevrolet","model":"camaro","year":"1969"},{"id":5,"make":"dodge","model":"intrepid","year":"2004"},{"id":6,"make":"acura","model":"integra","year":"1999"}]
```

```
> curl http://<unique-to-you>.cloudwaysapps.com/api/v1/car

[{"id":4,"make":"chevrolet","model":"camaro","year":"1969"},
{"id":5,"make":"dodge","model":"intrepid","year":"2004"},
{"id":6,"make":"acura","model":"integra","year":"1999"}]
```

```
> curl http://<unique-to-you>.cloudwaysapps.com/api/v1/car [{"id":4,"make":"chevrolet","model":"camaro","year":"1969"},{"id":5,"make":"dodge","model":"intrepid","year":"2004"},{"id":6,"make":"acura","model":"integra","year":"1999"}]
```

As a final confirmation,I can verify the data using the Cloudways MySQL manager:

```
SELECT * FROM `cars` LIMIT 50 (0.000 s) Edit
```

☐ Modify	id	make	model	year
☐ edit	4	chevrolet	camaro	1969
☐ edit	5	dodge	intrepid	2004
☐ edit	6	acura	integra	1999

(3 rows) ☐ whole result

You might also like: Create ToDo App With Authentication Using Lumen

Conclusion

This article highlighted Lumen, a Laravel based micro framework that is optimized for REST API. I created the model, controller and the view for the CURL based app. The aim of this tutorial is to show how easy it is to create REST API with Lumen. If you have any problems following the code or would like to contribute to the discussion, please leave a comment below.

Share your opinion in the comment section.

COMMENT NOW

Fahad Saleh

Fahad Saleh is a DevOps Engineer at Cloudways

Get Connected on:

Twitter

Community Forum

PRODUCT & SOLUTION

- WordPress Hosting
- Magento Hosting
- PHP Cloud Hosting
- Laravel Hosting
- Drupal Hosting
- Joomla Hosting
- WooCommerce Hosting
- Cloudways Platform
- Cloudways API
- Breeze – Free WordPress Cache
- Add-ons
- CloudwaysCDN

COMPANY

- About us
- Testimonials
- Terms
- Media Kit
- Sitemap

SUPPORT

- Knowledge base
- Contact us
- Blog
- Community
- Feedback
- Free Website Migration

QUICK LINKS

- Features
- Pricing
- Partners
- Cloud Affiliate Program

CLOUDWAYS
52 Springvale, Pope Pius XII Street Mosta MST2653, Malta