Getting Started With Lumen - Setup, Authorization, Cors, Cookies

Sumaiya Tasmeem

15 May, 2022

Lumen setup

1. Installation of Lumen:

```
composer create-project laravel/lumen project-name 7.*
```

2. Serving the Lumen application

```
php -S localhost:8000 -t public
```

3. To use Laravel code generator commands in Lumen, add following package:

```
composer require flipbox/lumen-generator
```

Open bootstrap/app.php file and add following code for configuration:

```
$app->register(Flipbox\LumenGenerator\LumenGeneratorServiceProvider:
:class);
```

Find more details about lumen generators at
https://github.com/flipboxstudio/lumen-generator.

JWT Authentication with Lumen 7.0.x API

1. Install required packages -

```
composer require chuckrincon/lumen-config-discover
```

```
composer require tymon/jwt-auth:dev-develop
```

2. Create a new folder in the root directory named config to add all the configuration files. Run the following command in root directory -

```
mkdir config
```

3. Open the file bootstrap/app.php –

Uncomment the following codes:

```
$app->withFacades();

$app->withEloquent();

$app->routeMiddleware([

    'auth' => App\Http\Middleware\Authenticate::class,

]);

$app->register(App\Providers\AuthServiceProvider::class);
```

Add the following code in bootstrap/app.php file:

```
$app->register(Tymon\JWTAuth\Providers\LumenServiceProvider::class);

$app->register(Chuckrincon\LumenConfigDiscover\DiscoverServiceProvider::cla
ss);
```

4. Generate JWT secret by typing the following command:

```
php artisan jwt:secret
```

5. Create the file config/auth.php and add the following to set up the auth config:

```
<?php
return [
    'defaults' => [
        'guard' => 'api',
        'passwords' => 'users',
    ],
    'guards' => [
        'api' => [
            'driver' => 'jwt',
            'provider' => 'users',
        ],
    ],
    'providers' => [
        'users' => [
            'driver' => 'eloquent',
            'model' => \App\Models\User::class
        ]
    ]
];
```

6. Run the following command to create users table and password reset table:

```
php artisan make:migration create_users_table
```

```
php artisan make:migration create_password_resets_table
```

Place the following code in the up() method for the create_users_table migration:

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

Place the following code in the up() method for the create_password_resets_table migration:

```
public function up()
{
    Schema::create('password_resets', function (Blueprint $table) {
        $table->string('email')->index();
        $table->string('token');
        $table->timestamp('created_at')->nullable();
    });
}
```

Finally run the migrate commands to create the tables.

```
php artisan migrate
```

7. Set up the User model under app/Models/User.php and add following codes:

Add the following use statement:

```
use Tymon\JWTAuth\Contracts\JWTSubject;
```

Add JWTSubject to the class implements clause:

```
class User extends Model implements AuthenticatableContract,
AuthorizableContract, JWTSubject
```

Add following methods for JWT handling:

```
/**
* Retrieve the identifier for the JWT key.
*
* @return mixed
*/
public function getJWTIdentifier()
{
   return $this->getKey();
}
/**
* Return a key value array, containing any custom claims to be added to the
JWT.
*
* @return array
*/
public function getJWTCustomClaims()
{
   return [];
}
```

8. Create the app/Http/Controllers/AuthController.php file and place the following code:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\User;
use Illuminate\Support\Facades\Auth;

class AuthController extends Controller
{
   public function __construct()
   {
       $this->middleware('auth', ['except' => ['login', 'register',]]);
   }

   /*
    * Attempt to register a new user to the API.
    *
    * @param Request $request
    * @return Response
    */
   public function register(Request $request)
   {
       // Are the proper fields present?
       $this->validate($request, [
           'name' => 'required|string|max:255',
```

```
            'email' => 'required|string|email|max:255|unique:users',
            'password' => 'required|string|min:6'
        ]);
        try {
            $user = new User;
            $user->name = $request->input('name');
            $user->email = $request->input('email');
            $plainPassword = $request->input('password');
            $user->password = app('hash')->make($plainPassword);
            $user->save();
            return response()->json(['user' => $user, 'message' =>
'CREATED'], 201);
        } catch (\Exception $e) {
            return response()->json(['message' => 'User Registration
Failed!'], 409);
        }
    }

    /*
     * Attempt to authenticate the user and retrieve a JWT.
     * Note: The API is stateless. This method _only_ returns a JWT. There
is not an
     * indicator that a user is logged in otherwise (no sessions).
     *
     * @param Request $request
     * @return Response
     */
    public function login(Request $request)
    {
        // Are the proper fields present?
        $this->validate($request, [
            'email' => 'required|string',
            'password' => 'required|string',
        ]);
        $credentials = $request->only(['email', 'password']);
        if (!$token = Auth::attempt($credentials)) {
            // Login has failed
            return response()->json(['message' => 'Unauthorized'], 401);
        }
        return $this->respondWithToken($token);
    }

    /*
     * Log the user out (Invalidate the token). Requires a login to use as
the
     * JWT in the Authorization header is what is invalidated
     *
     * @return \Illuminate\Http\JsonResponse
     */
    public function logout()
    {
        auth()->logout();
```

```
        return response()->json(['message' => 'User successfully signed
out']);
    }

    /*
     * Refresh the current token.
     *
     * @return \Illuminate\Http\JsonResponse
     */
    public function refresh()
    {
        return $this->respondWithToken(auth()->refresh());
    }

    /*
     * Helper function to format the response with the token.
     *
     * @return \Illuminate\Http\JsonResponse
     */
    private function respondWithToken($token)
    {
        return response()->json([
            'token' => $token,
            'token_type' => 'bearer',
            'expires_in' => Auth::factory()->getTTL() * 60
        ], 200);
    }
}
```

9. Finally, add following code to routes/web.php file:

```
$router->post('/login', 'AuthController@login');
$router->post('/register', 'AuthController@register');
$router->group(
    [
        'middleware' => 'auth'

    ], function( $router ) {
    $router->post('/logout', 'AuthController@logout');
    $router->get('/refresh', 'AuthController@refresh');
    $router->post('/refresh', 'AuthController@refresh');

});
```

Lumen Cors:

1. Create app/Middleware/CorsMiddleware.php file and add the following code:

```
<?php

namespace App\Http\Middleware;

use Closure;

class CorsMiddleware
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @return mixed
     */
    public function handle($request, Closure $next)
    {
        $response = $next($request);

        $response->header('Access-Control-Allow-Methods', 'HEAD, GET, POST,
PUT, PATCH, DELETE');

        $response->header('Access-Control-Allow-Headers',
$request->header('Access-Control-Request-Headers'));

        $response->header('Access-Control-Allow-Origin', '*');

        return $response;
    }
}
```

2. Create app/Providers/CatchAllOptionsRequestsProvider.php file and add the following code:

```
<?php

namespace App\Providers;

use Illuminate\Support\ServiceProvider;

class CatchAllOptionsRequestsProvider extends ServiceProvider {

    public function register()
    {
        $request = app('request');

        if ($request->isMethod('OPTIONS'))
```

```
        {
            app()->options($request->path(), function() { return
response('', 200); });
        }
    }

}
```

3. Add following code to bootstrap/app.php file:

```
$app->register(App\Providers\CatchAllOptionsRequestsProvider::class);

$app->routeMiddleware([
    'cors' => App\Http\Middleware\CorsMiddleware::class
]);
```

4. Finally, add following code to routes/web.php file to enable cors to specific routes:

```
$router->group(
    [
        'middleware' => 'cors'
    ], function ($router) {

    $router->get('/', function () {
        return response()->json([
            "data" => "test",
            "status" => Response::HTTP_ACCEPTED
        ]);
    });
```

Final code in web.php will look like this:

```
$router->group(
    [
        'middleware' => 'auth', 'cors',
    ], function ($router) {

    $router->get('/', function () {
        return response()->json([
            "data" => "test",
            "status" => Response::HTTP_ACCEPTED
        ]);
    });
    $router->post('/logout', 'AuthController@logout');
    });
```

5. Lastly, for testing cors, create an index.html file and add the following codes in <script> tags.

```
<script
   src="https://code.jquery.com/jquery-3.6.0.min.js"
   integrity="sha256-/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4="
   crossorigin="anonymous">
</script>

<script>
   $.post(
       "http://localhost:8000/login",
       {email: "user_email", password: "user_password"},
       function (response) {
           console.log(response);
       }
   );

   $.get("http://localhost:8000/", function (el) {
       console.log(el);
   });
</script>
```

Lumen Cookies

1. Add following code in app/Http/Middleware/Authenticate.php to set the login token.

```
public function handle($request, Closure $next, $guard = null)
{
   if ($request->cookie('auth_token_key') !== null) {
       $request->headers->set('Authorization', 'bearer ' .
$request->cookie('auth_token_key'));
       return $next($request);
   } else {
       return response('Unauthorized.', 401);
   }
}
```

2. Next add the following code in app/Http/Controllers/AuthController.php to save the login token.

```
use Symfony\Component\HttpFoundation\Cookie;
```

Modify the following code under AuthController class:

```
public function login(Request $request)
   {
       // Are the proper fields present?
       $this->validate($request, [
           'email' => 'required|string',
           'password' => 'required|string',
       ]);
       $credentials = $request->only(['email', 'password']);
       if (!$token = Auth::attempt($credentials)) {
           // Login has failed
           return response()->json(['message' => 'Unauthorized'], 401);
       }
       return $this->respondWithToken($token)->withCookie(new
Cookie('auth_token_key', $token));
   }

public function logout()
   {
       auth()->logout();
       return response()->json(['message' => 'User successfully signed
out'])->withCookie(new Cookie('auth_token_key', null));
   }
```