



LARAVEL

THE BASIC PART II

by dimitri.casier@howest.be

REQUEST

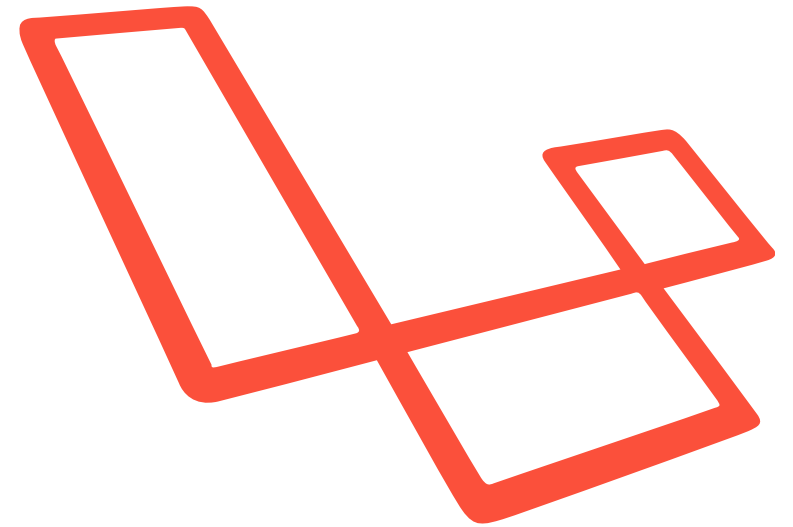
RESPONSE

VALIDATION

SESSION

LOGS

MIDDELWARE



REQUEST

```
Route::get('/request', function(\Illuminate\Http\Request $request){

    return [
        $request->path(),           // url without domain
        request()->path(),         // -- same but with helper
        $request->url(),            // -- url without querysting
        $request->fullUrl(),        // -- url with querysting
        $request->method(),         // -- method of request
        $request->isMethod( method: 'post'), // -- check method
        $request->all(),            // -- $_GET and $_POST
        $request->input( key: 'key', default: ''), // -- from $_GET and $_POST
        $request->get( key: 'key', default: ''), // -- from $_GET and $_POST
        $request->query( key: 'key', default: ''), // -- from $_GET
        $request->only(['email', 'password']), // -- also except
        $request->cookie( key: 'key', default: ''), // -- $_COOKIE
        $request->file( key: 'photo'), // -- uploaded file
        $request->file( key: 'photo')->path(), // -- path uploaded file(s)
        $request->file( key: 'photo')->extension(), // -- extension uploaded file(s)
    ];
});
```

RESPONSE

```
return response( content: 'content')
    ->header( key: 'Content-Type', values: 'a-type')
    ->header( key: 'X-Header-One', values: 'Header Value')
    ->header( key: 'X-Header-Two', values: 'Header Value')
    ->cookie('$name', '$value', '$minutes', '$path', '$domain', '$secure', '$httpOnly');
```

```
return redirect()->route( route: 'login');
```

```
return response()
    ->view( view: 'hello', data: '$data', status: 200)
    ->header( key: 'Content-Type', values: '$type');
```

```
return response()->json([
    'name' => 'Dimitri Casier',
    'age' => '21'
]);
```

```
return response()
    ->download( file: '$pathToFile', name: '$name', headers: '$headers')
    ->deleteFileAfterSend();
```

```
return response()->file( file: '$pathToFile', headers: '$headers');
```

VALIDATION

```
Route::post('/store', function(\Illuminate\Http\Request $request) {  
  
    $validator = Validator::make($request->all(), [  
        'title' => 'required|unique:posts|max:255',  
        'body' => 'required',  
    ]);  
  
    if ($validator->fails()) {  
        // return errors: $validator->errors()->all();  
    }  
  
    // Store the blog post...  
  
});
```

[HTTPS://LARAVEL.COM/DOCS/5.8/VALIDATION#AVAILABLE-VALIDATION-RULES](https://laravel.com/docs/5.8/validation#available-validation-rules)

SESSIONS

Configuration

The session configuration file is stored at `config/session.php`. Be sure to review the options available to you in this file. By default, Laravel is configured to use the `file` session driver, which will work well for many applications. In production applications, you may consider using the `memcached` or `redis` drivers for even faster session performance.

The session `driver` configuration option defines where session data will be stored for each request. Laravel ships with several great drivers out of the box:

- `file` - sessions are stored in `storage/framework/sessions`.
- `cookie` - sessions are stored in secure, encrypted cookies.
- `database` - sessions are stored in a relational database.
- `memcached` / `redis` - sessions are stored in one of these fast, cache based stores.
- `array` - sessions are stored in a PHP array and will not be persisted.



The array driver is used during testing and prevents the data stored in the session from being persisted.

SESSIONS

Driver Prerequisites

Database

When using the `database` session driver, you will need to create a table to contain the session items. Below is an example `Schema` declaration for the table:

```
Schema::create('sessions', function ($table) {  
    $table->string('id')->unique();  
    $table->unsignedInteger('user_id')->nullable();  
    $table->string('ip_address', 45)->nullable();  
    $table->text('user_agent')->nullable();  
    $table->text('payload');  
    $table->integer('last_activity');  
});
```

You may use the `session:table` Artisan command to generate this migration:

```
php artisan session:table
```

```
php artisan migrate
```

SESSIONS

```
Route::get('/session', function() {  
    $value = session( key: 'key', default: 'default');  
    $values = session()->all();  
  
    session()->put('key', 'value');  
    session()->forget( keys: 'key');  
    session()->forget(['key1', 'key2']);  
    session()->flush();  
});
```


LOGS

```
Route::get('/logs', function() {  
    $message = 'HELP!';  
  
    Log::emergency($message);  
    Log::alert($message);  
    Log::critical($message);  
    Log::error($message);  
    Log::warning($message);  
    Log::notice($message);  
    Log::info($message);  
    Log::debug($message);  
});
```

MIDDLEWARE – CREATING

COMMAND:

php artisan make:middleware RoleMiddleware

Kernel.php:

```
class Kernel extends HttpKernel

    /**
     * The application's global HTTP middleware stack.
     *
     * These middleware are run during every request to your application.
     *
     * @var array
     */
    protected $middleware = [...];

    /**
     * The application's route middleware groups.
     *
     * @var array
     */
    protected $middlewareGroups = [...];

    /**
     * The application's route middleware.
     *
     * These middleware may be assigned to groups or used individually.
     *
     * @var array
     */
    protected $routeMiddleware = [
        'auth' => \App\Http\Middleware\Authenticate::class,
        'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
        'bindings' => \Illuminate\Routing\Middleware\SubstituteBindings::class,
        'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,
        'can' => \Illuminate\Auth\Middleware\Authorize::class,
        'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
        'signed' => \Illuminate\Routing\Middleware\ValidateSignature::class,
        'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
        'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,
        'jwt.auth' => \Tymon\JWTAuth\Http\Middleware\Authenticate::class,
        'jwt.refresh' => \Tymon\JWTAuth\Http\Middleware\RefreshToken::class,
        'role' => \App\Http\Middleware\RoleMiddleware::class,
    ];

    /**
     * The priority-sorted list of middleware. ...
     */
    protected $middlewarePriority = [...];
```

Install Dependencies
From package.json
Run "npm install" Don't ask again

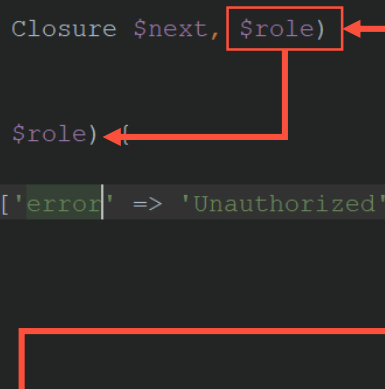
MIDDLEWARE – BEFORE

```
namespace App\Http\Middleware;

use Closure;

class RoleMiddleware
{
    /**
     * @param $request
     * @param Closure $next
     * @param $role
     * @return mixed
     */
    public function handle($request, Closure $next, $role)
    {
        // -- or Auth::user()
        if($request->user()->role != $role)
        {
            //abort(401);
            return response()->json(['error' => 'Unauthorized'], status: 401);
        }

        return $next($request);
    }
}
```



routes/api.php:

```
// -- eternal cards
Route::middleware(['jwt.auth', 'role:admin'])
    ->get('/eternal-cards', 'EternalCardApiController@index')
    ->name('api.eternal-cards.index');
```

MIDDLEWARE – AFTER (EXAMPLE)


```
namespace Tymon\JWTAuth\Http\Middleware;

use Closure;

class AuthenticateAndRenew extends BaseMiddleware
{
    /**
     * Handle an incoming request.
     *
     * @param \Illuminate\Http\Request $request
     * @param Closure $next
     *
     * @throws \Symfony\Component\HttpKernel\Exception\UnauthorizedHttpException
     *
     * @return mixed
     */
    public function handle($request, Closure $next)
    {
        $this->authenticate($request);

        $response = $next($request);

        // Send the refreshed token back to the client.
        return $this->setAuthenticationHeader($response);
    }
}
```

 Install Dependencies



PROJECT TIME!