Eric The Coder
Posted on 16 Feb 2021 • Updated on 18 Sept 2021

My beloved Laravel cheat sheet

#laravel #php #beginners

Follow me!: Follow @EricTheCoder_

Laravel cheat sheet

Project commands

```
// New project
$ laravel new projectName

// Launch server/project
$ php artisan serve

// commands list
$ php artisan list

// command help
$ php artisan help migrate

// Laravel console
$ php artisan tinker

// Route list
$ php artisan route:list
```

Commons commands

```
// Database migration
$ php artisan migrate

// Data seed
$ php artisan db:seed
```

```
// Create table migration
$ php artisan make:migration create_products_table


// Create from model with options:
// -m (migration), -c (controller), -r (resource controllers), -f (factory), -s (seed)
$ php artisan make:model Product -mcf

// Create a controller
$ php artisan make:controller ProductsController

// Update table migration
$ php artisan make:migration add_date_to_blogposts_table

// Rollback latest migration
php artisan migrate:rollback

// Rollback all migrations
php artisan migrate:reset

// Rollback all and re-migrate
php artisan migrate:refresh

// Rollback all, re-migrate and seed
php artisan migrate:refresh --seed
```

Create and update data tables (with migrations)

```
// Create data table
$ php artisan make:migration create_products_table

// Create table(migration exemple)
Schema::create('products', function (Blueprint $table) {
    // auto-increment primary key
    $table->id();
    // created_at and updated_at
    $table->timestamps();
    // Unique constraint
    $table->string('modelNo')->unique();
    // Not required
    $table->text('description')->nullable();
    // Default value
    $table->boolean('isActive')->default(true);
    // Index
    $table->index(['account_id', 'created_at']);
    // Foreign Key relation
    $table->foreignId('user_id')->constrained('users')->onDelete('cascade');
});

// Update table (migration exemple)
$ php artisan make:migration add_comment_to_products_table

// up()
Schema::table('users', function (Blueprint $table) {
    $table->text('comment');
});

// down()
Schema::table('users', function (Blueprint $table) {
    $table->dropColumn('comment');
});
```

Models

```
// Model mass assignment list exclude attributes
```

```php
protected $guarded = []; // empty == All

// Or list included attributes
protected $fillable = ['name', 'email', 'password',];


// One to Many relationship (posts with many comments)
public function comments()
{
    return $this->hasMany(Comment:class);
}

// One to Many relationship (comments with one post)
public function post()
{
    return $this->belongTo(Post::class);
}

// One to one relationship (Author with one profile)
public function profile()
{
    return $this->hasOne(Profile::class);
}

// One to one relationship (Profile with one Author)
public function author()
{
    return $this->belongTo(Author::class);
}

// Many to Many relationship
// 3 tables (posts, tags and post_tag)
// post_tag (post_id, tag_id)

// in Tag model...
public function posts()
    {
        return $this->belongsToMany(Post::class);
    }

// in Post model...
public function tags()
    {
        return $this->belongsToMany(Tag::class);
    }
```

Factory

```php
// ex: database/factories/ProductFactory.php
public function definition() {
    return [
        'name' => $this->faker->text(20),
        'price' => $this->faker->numberBetween(10, 10000),
    ];
}
// all fakers options : https://github.com/fzaninotto/Faker
```

Seed

```php
// ex: database/seeders/DatabaseSeeder.php
public function run() {
    Product::factory(10)->create();
}
```

Running Seeders

```
$ php artisan db:seed
// or with migration
$ php artisan migrate --seed
```

Eloquent ORM

```
// New
$flight = new Flight;
$flight->name = $request->name;
$flight->save();

// Update
$flight = Flight::find(1);
$flight->name = 'New Flight Name';
$flight->save();

// Create
$user = User::create(['first_name' => 'Taylor','last_name' => 'Otwell']);

// Update All:
Flight::where('active', 1)->update(['delayed' => 1]);

// Delete
$current_user = User::Find(1)
$current_user.delete();

// Delete by id:
User::destroy(1);

// Delete all
$deletedRows = Flight::where('active', 0)->delete();

// Get All
$items = Item::all().

// Find one by primary key
$flight = Flight::find(1);

// display 404 if not found
$model = Flight::findOrFail(1);

// Get last entry
$items = Item::latest()->get()

// Chain
$flights = App\Flight::where('active', 1)->orderBy('name', 'desc')->take(10)->get();

// Where
Todo::where('id', $id)->firstOrFail()

// Like
Todos::where('name', 'like', '%' . $my . '%')->get()

// Or where
Todos::where('name', 'mike')->orWhere('title', '=', 'Admin')->get();

// Count
$count = Flight::where('active', 1)->count();

// Sum
$sum = Flight::where('active', 1)->sum('price');

// Contain?
if ($project->$users->contains('mike'))
```

Routes

```php
// Basic route closure
Route::get('/greeting', function () {
    return 'Hello World';
});

// Route direct view shortcut
Route::view('/welcome', 'welcome');

// Route to controller class
use App\Http\Controllers\UserController;
Route::get('/user', [UserController::class, 'index']);

// Route only for specific HTTP verbs
Route::match(['get', 'post'], '/', function () {
    //
});

// Route for all verbs
Route::any('/', function () {
    //
});

// Route Redirect
Route::redirect('/clients', '/customers');


// Route Parameters
Route::get('/user/{id}', function ($id) {
    return 'User '.$id;
});

// Optional Parameter
Route::get('/user/{name?}', function ($name = 'John') {
    return $name;
});

// Named Route
Route::get(
    '/user/profile',
    [UserProfileController::class, 'show']
)->name('profile');

// Resource
Route::resource('photos', PhotoController::class);

GET /photos index   photos.index
GET /photos/create  create  photos.create
POST    /photos store   photos.store
GET /photos/{photo} show    photos.show
GET /photos/{photo}/edit    edit    photos.edit
PUT/PATCH   /photos/{photo} update  photos.update
DELETE  /photos/{photo} destroy photos.destroy

// Nested resource
Route::resource('photos.comments', PhotoCommentController::class);

// Partial resource
Route::resource('photos', PhotoController::class)->only([
    'index', 'show'
]);

Route::resource('photos', PhotoController::class)->except([
    'create', 'store', 'update', 'destroy'
]);

// URL with route name
```

```php
$url = route('profile', ['id' => 1]);

// Generating Redirects...
return redirect()->route('profile');

// Route Prefix group
Route::prefix('admin')->group(function () {
    Route::get('/users', function () {
        // Matches The "/admin/users" URL
    });
});

// Route model binding
use App\Models\User;
Route::get('/users/{user}', function (User $user) {
    return $user->email;
});

// Route model binding (other than id)
use App\Models\User;
Route::get('/posts/{post:slug}', function (Post $post) {
    return view('post', ['post' => $post]);
});

// Route fallback
Route::fallback(function () {
    //
});
```

Cache

```php
// Route Caching
php artisan route:cache

// Retrieve & Store (key, duration, content)
$users = Cache::remember('users', now()->addMinutes(5), function () {
    return DB::table('users')->get();
});
```

Controllers

```php
// Set validation rules
protected $rules = [
    'title' => 'required|unique:posts|max:255',
    'name' => 'required|min:6',
    'email' => 'required|email',
    'publish_at' => 'nullable|date',
];

// Validate
$validatedData = $request->validate($rules)

// Show 404 error page
abort(404, 'Sorry, Post not found')

// Controller CRUD exemple
Class ProductsController
{

   public function index()
   {
       $products = Product::all();

       // app/resources/views/products/index.blade.php
       return view('products.index', ['products', $products]);
   }
```

```php
    public function create()
    {
        return view('products.create');
    }

    public function store()
    {
        Product::create(request()->validate([
            'name' => 'required',
            'price' => 'required',
            'note' => 'nullable'
        ]));

        return redirect(route('products.index'));
    }

    //method with model injection
    public function show(Product $product)
    {
        return view('products.show', ['product', $product]);
    }

    public function edit(Product $product)
    {
        return view('products.edit', ['product', $product]);
    }

    public function update(Product $product)
    {
        Product::update(request()->validate([
            'name' => 'required',
            'price' => 'required',
            'note' => 'nullable'
        ]));

        return redirect(route($product->path()));
    }

    public function delete(Product $product)
    {
         $product->delete();
         return redirect("/contacts");
    }
}

// Query Params www.demo.html?name=mike
request()->name //mike

// Form data (or default value)
request()->input('email', 'no@email.com')
```

Template

```php
<!-- Route name -->
<a href="{{ route('routeName.show', $id) }}">

<!-- Template inheritance -->
@yield('content')  <!-- layout.blade.php -->
@extends('layout')
@section('content') … @endsection

<!-- Template include -->
@include('view.name', ['name' => 'John'])

<!-- Template variable -->
{{ var_name }}
```

```
<!-- raw safe html variable -->
{ !! var_name !! }


<!-- Interation -->
@foreach ($items as $item)
   {{ $item.name }}
   @if($loop->last)
       $loop->index
   @endif
@endforeach

<!-- Conditional -->
@if ($post->id === 1)
    'Post one'
@elseif ($post->id === 2)
    'Post two!'
@else
    'Other'
@endif

<!-- Form -->
<form method="POST" action="{{ route('posts.store') }}">

@method('PUT')
@csrf

<!-- Request path match -->
{{ request()->is('posts*') ? 'current page' : 'not current page' }}

<!-- Route exist? -->
@if (Route::has('login'))

<!-- Auth blade variable -->
@auth
@endauth
@guest

<!-- Current user -->
{{ Auth::user()->name }}

<!-- Validations errors -->
@if ($errors->any())
    <div class="alert alert-danger">
        <ul>
            @foreach ($errors->all() as $error)
                <li>{{ $error }}</li>
            @endforeach
        </ul>
    </div>
@endif

<!-- Check a specific attributes -->
<input id="title" type="text" class="@error('title') is-invalid @enderror">

<!-- Repopulate form with data from previous request -->
{{ old('name') }}
```

Database direct access no model

```
use Illuminate\Support\Facades\DB;
$user = DB::table('users')->first();
$users = DB::select('select name, email from users');
DB::insert('insert into users (name, email, password) value(?, ?, ?)', ['Mike', 'mike@hey.com', 'pass123']);
DB::update('update users set name = ? where id = 1', ['eric']);
DB::delete('delete from users where id = 1');
```

Helpers

```php
// Display variable content and kill execution
dd($products)

// Create a Laravel collection from array.
$collection = collect($array);

// Sort by description ascending
$ordered_collection = $collection->orderBy('description');

// Reset numbering value
$ordered_collection = $ordered_collection->values()->all();


// Return project full Path
app\ : app_path();
resources\ : resource_path();
database\ :database_path();
```

Flash & Session

```php
// Flash (only next request)
$request->session()->flash('status', 'Task was successful!');

// Flash with redirect
return redirect('/home')->with('success' => 'email sent!');

// Set Session
$request->session()->put('key', 'value');

// Get session
$value = session('key');
If session: if ($request->session()->has('users'))

// Delete session
$request->session()->forget('key');

// Display flash in template
@if (session('message')) {{ session('message') }} @endif
```

HTTP Client

```php
// Use package
use Illuminate\Support\Facades\Http;

//Http get
$response = Http::get('www.thecat.com')
$data = $response->json()

// Http get with query
$res = Http::get('www.thecat.com', ['param1', 'param2'])

// Http post
$res = Http::post('http://test.com', ['name' => 'Steve','role' => 'Admin']);

// Bearer
$res = Http::withToken('123456789')->post('http://the.com', ['name' => 'Steve']);

// Headers
$res = Http::withHeaders(['type'=>'json'])->post('http://the.com', ['name' => 'Steve']);
```

Storage (helper class to store files locally or in the cloud)

```
// Public config: Local storage/app/public
Storage::disk('public')->exists('file.jpg'))
// S3 config: storage: Amazon cloud ex:
Storage::disk('s3')->exists('file.jpg'))

// Make the Public config available from the web
php artisan storage:link

// Get or put file in the storage folder
use Illuminate\Support\Facades\Storage;
Storage::disk('public')->put('example.txt', 'Contents');
$contents = Storage::disk('public')->get('file.jpg');

// Access files by generating a url
$url = Storage::url('file.jpg');
// or by direct path to public config
<img src={{ asset('storage/image1.jpg') }}/>

// Delete file
Storage::delete('file.jpg');

// Trigger download
Storage::disk('public')->download('export.csv');
```

Install a project from github

```
$ git clone {project http address} projectName
$ cd projectName
$ composer install
$ cp .env.example .env
$ php artisan key:generate
$ php artisan migrate
$ npm install
```

Heroku deployment

```
// Install Heroku on local machine (macOs)
$ brew tap heroku/brew && brew install heroku

// Login to heroku (create account if none)
$ heroku login

// Create Procile
$ touch Procfile

// Save in Profile
web: vendor/bin/heroku-php-apache2 public/
```

Rest API (create a Rest API endpoint)

API Routes (All api routes will be prefix with 'api/')

```
// routes/api.php
Route::get('products', [App\Http\Controllers\ProductsController::class, 'index']);
Route::get('products/{product}', [App\Http\Controllers\ProductsController::class, 'show']);
Route::post('products', [App\Http\Controllers\ProductsController::class, 'store']);
```

API Resource (Layer that sits between your models and the JSON responses)

```
$ php artisan make:resource ProductResource
```

API Resource definition file

```
// app/resource/ProductResource.php
public function toArray($request)
    {
        return [
            'id' => $this->id,
            'name' => $this->name,
            'price' => $this->price,
            'custom' => 'This is a custom field',
        ];
    }
```

API Controller (Best practice is to place your API controllers inside app/Http/Controllers/Api/v1/)

```
public function index() {
        //$products = Product::all();
        $products = Product::paginate(5);
        return ProductResource::collection($products);
    }

    public function show(Product $product) {
        return new ProductResource($product);
    }

    public function store(StoreProductRequest $request) {
        $product = Product::create($request->all());
        return new ProductResource($product);
    }
```

API Token authentication

First you need to create a Token for a specific user.

```
$user = User::first();
$user->createToken('dev token');
// plainTextToken: "1|v39On3Uvwl0yA4vex0f9SgOk3pVdLECDk4Edi4OJ"
```

You can then use this token is a request

```
GET api/products (Auth Bearer Token: plainTextToken)
```

Authorization rules

You can create a token with pre-define auth rules

```
$user->createToken('dev token', ['product-list']);

// in controllers
if !auth()->user()->tokenCan('product-list') {
    abort(403, "Unauthorized");
}
```

Top comments (9)

SiteBuilderStudio • 24 Feb 22

Thank you Eric!

Mr.Jingles • 19 Jul 21

Спасибо за памятку)

Facu • 17 Feb 21

So helpfull !!

Bobby Iliev • 17 Feb 21

This is perfect!

Tim Bogdanov • 13 Oct 21

thank you

Paul Duvall • 17 Feb 21

Very nice! I'll be bookmarking this one!

Mr.Jingles • 9 Sept 22

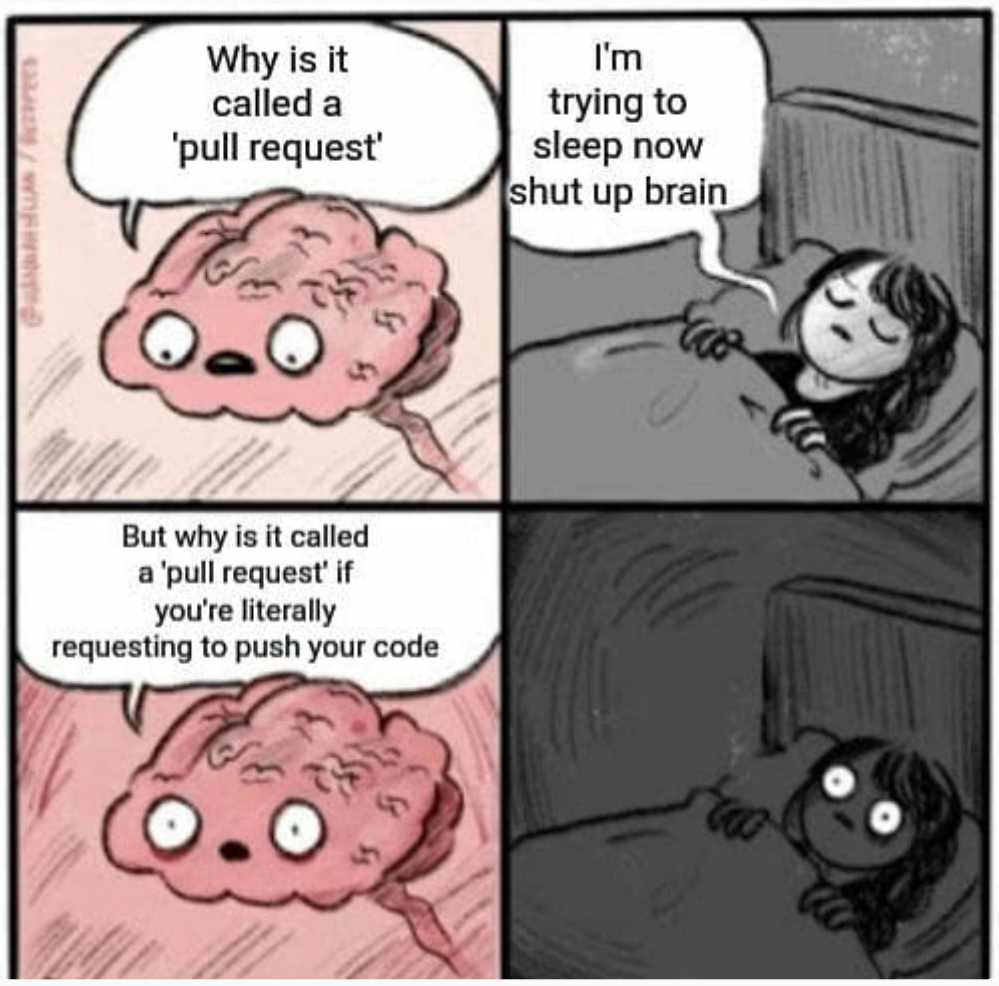
Премного благодарен!))

azeez • 17 Feb 21

Great stuff keep it up. savingit for daily code 😉

Dao Trong Duc • 15 Jul 21

thanks !



Stop by this week's meme thread!

Eric The Coder

Businessman and blogger #Javascript, #Python and #PHP. My favorite frameworks/librairies are #React, #Laravel, and #Django. I am also a fan of #TailwindCSS

LOCATION
Canada

JOINED
3 Sept 2020

More from Eric The Coder

Python : String Manipulations
#python #tutorial #beginners

Python : Crash Course
#python #tutorial #beginners

PHP crash course : require, include, files manipulation and enumerations
#php #backends #tutorial #beginners