Laravel Cheatsheet

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
```

Models

```


// Model mass assignment list exclude attributes
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);
    }

```php




## 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
~~~~

Seeder

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

$ 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'))
```

Cache

```
// 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

```
// 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]);
   }

   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')
```

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

```
// 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

```
// 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

```
// 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
```

Rest API (create a Rest API endpoint)

```
// 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 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

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

uthorization 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");
}
```