### Laravel Cheat Sheet

**Artisan**

php artisan --help OR -h

php artisan --quiet OR -q

php artisan --version OR -V

php artisan --no-interaction OR -n

php artisan --ansi

php artisan --no-ansi

php artisan --env

// -v|vv|vvv Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug

php artisan --verbose

php artisan changes

php artisan clear-compiled

php artisan downgrade

php artisan dump-autoload

php artisan env

php artisan help

php artisan list

php artisan migrate

php artisan optimize

php artisan routes

php artisan serve

php artisan tinker

php artisan up

php artisan workbench

php artisan asset:publish [--bench[="vendor/package"]] [--path[="..."]] [package]

php artisan auth:reminders

php artisan cache:clear

php artisan command:make name [--command[="..."]] [--path[="..."]] [--namespace[="..."]]

php artisan config:publish

php artisan controller:make [--bench="vendor/package"]

php artisan db:seed [--class[="..."]] [--database[="..."]]

php artisan key:generate

php artisan migrate [--bench="vendor/package"] [--database[="..."]] [--path[="..."]] [--package[="..."]] [--pretend] [--seed]

php artisan migrate:install [--database[="..."]]

php artisan migrate:make name [--bench="vendor/package"] [--create] [--package[="..."]] [--path[="..."]] [--table[="..."]]

php artisan migrate:refresh [--database[="..."]] [--seed]

php artisan migrate:reset [--database[="..."]] [--pretend]

php artisan migrate:rollback [--database[="..."]] [--pretend]

php artisan queue:listen [--queue[="..."]] [--delay[="..."]] [--memory[="..."]] [--timeout[="..."]] [connection]

php artisan queue:subscribe [--type[="..."]] queue url

php artisan queue:work [--queue[="..."]] [--delay[="..."]] [--memory[="..."]] [--sleep] [connection]

php artisan session:table

php artisan view:publish [--path[="..."]] package

php artisan generate:model

php artisan generate:controller

php artisan generate:view

php artisan generate:seed

php artisan generate:migration

php artisan generate:resource (Generate a model, Generate index, show, create, and edit views, Generate a controller, Generate a migration with schema, Generate a table seeder, Migrate the database)

**Composer**

composer create-project laravel/laravel folder\_name

composer install

composer update

composer dump-autoload [--optimize]

composer self-update

**Routing**

Route::get('foo', function(){});

Route::get('foo', 'ControllerName@function');

Route::controller('foo', 'FooController');

**Triggering Errors**

App::abort(404);

App::missing(function($exception){});

throw new NotFoundHttpException;

**Route Parameters**

Route::get('foo/{bar}', function($bar){});

Route::get('foo/{bar?}', function($bar = 'bar'){});

**HTTP Verbs**

Route::any ('foo', function(){});

Route::post('foo', function(){});

Route::put('foo', function(){});

Route::patch('foo', function(){});

Route::delete('foo', function(){});

// RESTful actions

Route::resource('foo', 'FooController');

**Secure Routes**

Route::get('foo', array('https', function(){}));

**Route Constraints**

Route::get('foo/{bar}', function($bar){})

->where('bar', '[0-9]+');

Route::get('foo/{bar}/{baz}', function($bar, $baz){})

->where(array('bar' => '[0-9]+', 'baz' => '[A-Za-z]'))

**Filters**

// Declare an auth filter

Route::filter('auth', function(){});

// Register a class as a filter

Route::filter('foo', 'FooFilter');

// Routes in this group are guarded by the 'auth' filter

Route::get('foo', array('before' => 'auth', function(){}));

Route::group(array('before' => 'auth'), function(){});

// Pattern filter

Route::when('foo/\*', 'foo');

// HTTP verb pattern

Route::when('foo/\*', 'foo', array('post'));

**Named Routes**

Route::currentRouteName();

Route::get('foo/bar', array('as' => 'foobar', function(){}));

**Route Prefixing**

// This route group will carry the prefix 'foo'

Route::group(array('prefix' => 'foo'), function(){})

**Sub-Domain Routing**

// {sub} will be passed to the closure

Route::group(array('domain' => '{sub}.example.com'), function(){});

**URLs**

URL::full();

URL::current();

URL::previous();

URL::to('foo/bar', $parameters, $secure);

URL::action('FooController@method', $parameters, $absolute);

URL::route('foo', $parameters, $absolute);

URL::secure('foo/bar', $parameters);

URL::asset('css/foo.css', $secure);

URL::secureAsset('css/foo.css');

URL::isValidUrl('http://example.com');

URL::getRequest();

URL::setRequest($request);

URL::getGenerator();

URL::setGenerator($generator);

**Events**

Event::fire('foo.bar', array($bar));

Event::listen('foo.bar', function($bar){});

Event::listen('foo.\*', function($bar){});

Event::listen('foo.bar', 'FooHandler', 10);

Event::listen('foo.bar', 'BarHandler', 5);

Event::listen('foor.bar', function($event){ return false; });

Event::queue('foo', array($bar));

Event::flusher('foo', function($bar){});

Event::flush('foo');

Event::subscribe(new FooEventHandler);

**DB**

$results = DB::select('select \* from users where id = ?', array('value'));

DB::insert('insert into users (id, name) values (?, ?)', array(1, 'Dayle'));

DB::update('update users set votes = 100 where name = ?', array('John'));

DB::delete('delete from users');

DB::statement('drop table users');

DB::listen(function($sql, $bindings, $time){ code\_here() });

DB::transaction(function(){ code\_here() });

**Eloquent**

Model::create(array('key' => 'value'));

Model::destroy(1);

Model::all();

Model::find(1);

// Trigger an exception

Model::findOrFail(1);

Model::where('foo', '=', 'bar')->get();

Model::where('foo', '=', 'bar')->first();

// Exception

Model::where('foo', '=', 'bar')->firstOrFail();

Model::where('foo', '=', 'bar')->count();

Model::where('foo', '=', 'bar')->delete();

Model::whereRaw('foo = bar and cars = 2', array(20))->get();

Model::on('connection-name')->find(1);

Model::with('relation')->get();

Model::all()->take(10);

Model::all()->skip(10);

**Soft Delete**

Model::withTrashed()->where('cars', 2)->get();

Model::withTrashed()->where('cars', 2)->restore();

Model::where('cars', 2)->forceDelete();

Model::onlyTrashed()->where('cars', 2)->get();

**Events**

Model::creating(function($model){});

Model::created(function($model){});

Model::updating(function($model){});

Model::updated(function($model){});

Model::saving(function($model){});

Model::saved(function($model){});

Model::deleting(function($model){});

Model::deleted(function($model){});

Model::observe(new FooObserver);

**Mail**

Mail::send('email.view', $data, function($message){});

Mail::send(array('html.view', 'text.view'), $data, $callback);

Mail::queue('email.view', $data, function($message){});

Mail::queueOn('queue-name', 'email.view', $data, $callback);

Mail::later(5, 'email.view', $data, function($message){});

// Write all email to logs instead of sending

Mail::pretend();

**Queues**

Queue::push('SendMail', array('message' => $message));

Queue::push('SendEmail@send', array('message' => $message));

Queue::push(function($job) use $id {});

php artisan queue:listen

php artisan queue:listen connection

php artisan queue:listen --timeout=60

php artisan queue:work

**Localization**

App::setLocale('en');

Lang::get('messages.welcome');

Lang::get('messages.welcome', array('foo' => 'Bar'));

Lang::has('messages.welcome');

Lang::choice('messages.apples', 10);

**Files**

File::exists('path');

File::get('path');

File::getRemote('path');

File::getRequire('path');

File::requireOnce('path');

File::put('path', 'contents');

File::append('path', 'data');

File::delete('path');

File::move('path', 'target');

File::copy('path', 'target');

File::extension('path');

File::type('path');

File::size('path');

File::lastModified('path');

File::isDirectory('directory');

File::isWritable('path');

File::isFile('file');

// Find path names matching a given pattern.

File::glob($patterns, $flag);

// Get an array of all files in a directory.

File::files('directory');

// Get all of the files from the given directory (recursive).

File::allFiles('directory');

// Get all of the directories within a given directory.

File::directories('directory');

File::makeDirectory('path', $mode = 0777, $recursive = false);

File::copyDirectory('directory', 'destination', $options = null);

File::deleteDirectory('directory', $preserve = false);

File::cleanDirectory('directory');

**Input**

Input::get('key');

// Default if the key is missing

Input::get('key', 'default');

Input::has('key');

Input::all();

// Only retrieve 'foo' and 'bar' when getting input

Input::only('foo', 'bar');

// Disregard 'foo' when getting input

Input::except('foo');

**Session Input (flash)**

// Flash input to the session

Input::flash();

Input::flashOnly('foo', 'bar');

Input::flashExcept('foo', 'baz');

Input::old('key','default\_value');

**Files**

// Use a file that's been uploaded

Input::file('filename');

// Determine if a file was uploaded

Input::hasFile('filename');

// Access file properties

Input::file('name')->getRealPath();

Input::file('name')->getClientOriginalName();

Input::file('name')->getSize();

Input::file('name')->getMimeType();

// Move an uploaded file

Input::file('name')->move($destinationPath);

// Move an uploaded file

Input::file('name')->move($destinationPath, $fileName);

**Cache**

Cache::put('key', 'value', $minutes);

Cache::add('key', 'value', $minutes);

Cache::forever('key', 'value');

Cache::remember('key', $minutes, function(){ return 'value' });

Cache::rememberForever('key', function(){ return 'value' });

Cache::forget('key');

Cache::has('key');

Cache::get('key');

Cache::get('key', 'default');

Cache::get('key', function(){ return 'default'; });

Cache::increment('key');

Cache::increment('key', $amount);

Cache::decrement('key');

Cache::decrement('key', $amount);

Cache::section('group')->put('key', $value);

Cache::section('group')->get('key');

Cache::section('group')->flush();

**Cookies**

Cookie::get('key');

// Create a cookie that lasts for ever

Cookie::forever('key', 'value');

// Send a cookie with a response

$response = Response::make('Hello World');

$response->withCookie(Cookie::make('name', 'value', $minutes));

**Sessions**

Session::get('key');

Session::get('key', 'default');

Session::get('key', function(){ return 'default'; });

Session::put('key', 'value');

Session::all();

Session::has('key');

Session::forget('key');

Session::flush();

Session::regenerate();

Session::flash('key', 'value');

Session::reflash();

Session::keep(array('key1', 'key2'));

**Requests**

Request::path();

Request::is('foo/\*');

Request::url();

Request::segment(1);

Request::header('Content-Type');

Request::server('PATH\_INFO');

Request::ajax();

Request::secure();

**Responses**

return Response::make($contents);

return Response::make($contents, 200);

return Response::json(array('key' => 'value'));

return Response::json(array('key' => 'value'))

->setCallback(Input::get('callback'));

return Response::download($filepath);

return Response::download($filepath, $filename, $headers);

// Create a response and modify a header value

$response = Response::make($contents, 200);

$response->header('Content-Type', 'application/json');

return $response;

// Attach a cookie to a response

return Response::make($content)

->withCookie(Cookie::make('key', 'value'));

**Redirects**

return Redirect::to('foo/bar');

return Redirect::to('foo/bar')->with('key', 'value');

return Redirect::to('foo/bar')->withInput(Input::get());

return Redirect::to('foo/bar')->withInput(Input::except('password'));

return Redirect::to('foo/bar')->withErrors($validator);

return Redirect::back();

return Redirect::route('foobar');

return Redirect::route('foobar', array('value'));

return Redirect::route('foobar', array('key' => 'value'));

return Redirect::action('FooController@index');

return Redirect::action('FooController@baz', array('value'));

return Redirect::action('FooController@baz', array('key' => 'value'));

// If intended redirect is not defined, defaults to foo/bar.

return Redirect::intended('foo/bar');

**IoC**

App::bind('foo', function($app){ return new Foo; });

App::make('foo');

// If this class exists, it's returned

App::make('FooBar');

App::singleton('foo', function(){ return new Foo; });

App::instance('foo', new Foo);

App::bind('FooRepositoryInterface', 'BarRepository');

App::register('FooServiceProvider');

// Listen for object resolution

App::resolving(function($object){});

**Security**

**Passwords**

Hash::make('secretpassword');

Hash::check('secretpassword', $hashedPassword);

Hash::needsRehash($hashedPassword);

**Auth**

// Check if the user is logged in

Auth::check();

Auth::user();

Auth::attempt(array('email' => $email, 'password' => $password));

// Remember user login

Auth::attempt($credentials, true);

// Log in for a single request

Auth::once($credentials);

Auth::login(User::find(1));

Auth::loginUsingId(1);

Auth::logout();

Auth::validate($credentials);

Auth::basic('username');

Auth::onceBasic();

Password::remind($credentials, function($message, $user){});

**Encryption**

Crypt::encrypt('secretstring');

Crypt::decrypt($encryptedString);

Crypt::setMode('ctr');

Crypt::setCipher($cipher);

**Validation**

Validator::make(

array('key' => 'Foo'),

array('key' => 'required|in:Foo')

);

Validator::extend('foo', function($attribute, $value, $params){});

Validator::extend('foo', 'FooValidator@validate');

Validator::resolver(function($translator, $data, $rules, $msgs)

{

return new FooValidator($translator, $data, $rules, $msgs);

});

**Validation Rules**

accepted

active\_url

after:YYYY-MM-DD

before:YYYY-MM-DD

alpha

alpha\_dash

alpha\_num

between:1,10

confirmed

date

date\_format:YYYY-MM-DD

different:fieldname

email

exists:table,column

image

in:foo,bar,baz

not\_in:foo,bar,baz

integer

numeric

ip

max:value

min:value

mimes:jpeg,png

regex:[0-9]

required

required\_if:field,value

required\_with:foo,bar,baz

required\_without:foo,bar,baz

same:field

size:value

unique:table,column,except,idColumn

url

**Views**

View::make('path/to/view');

View::make('foo/bar')->with('key', 'value');

View::make('foo/bar', array('key' => 'value'));

View::exists('foo/bar');

// Share a value across all views

View::share('key', 'value');

// Nesting views

View::make('foo/bar')->nest('name', 'foo/baz', $data);

// Register a view composer

View::composer('viewname', function($view){});

//Register multiple views to a composer

View::composer(array('view1', 'view2'), function($view){});

// Register a composer class

View::composer('viewname', 'FooComposer');

View::creator('viewname', function($view){});

**Blade Templates**

@extends('layout.name')

@section('name') // Begin a section

@stop // End a section

@show // End a section and then show it

@yield('name') // Show a section name in a template

@include('view.name')

@include('view.name', array('key' => 'value'));

@lang('messages.name')

@choice('messages.name', 1);

@if

@else

@elseif

@endif

@unless

@endunless

@for

@endfor

@foreach

@endforeach

@while

@endwhile

{{ $var }} // Echo content

{{{ $var }}} // Echo escaped content

{{-- Blade Comment --}}

**Forms**

Form::open(array('url' => 'foo/bar', 'method' => 'PUT'));

Form::open(array('route' => 'foo.bar'));

Form::open(array('route' => array('foo.bar', $parameter)));

Form::open(array('action' => 'FooController@method'));

Form::open(array('action' => array('FooController@method', $parameter)));

Form::open(array('url' => 'foo/bar', 'files' => true));

Form::token();

Form::model($foo, array('route' => array('foo.bar', $foo->bar)));

Form::close;

**Form Elements**

Form::label('id', 'Description');

Form::label('id', 'Description', array('class' => 'foo'));

Form::text('name');

Form::text('name', $value);

Form::textarea('name');

Form::textarea('name', $value);

Form::textarea('name', $value, array('class' => 'name'));

Form::hidden('foo', $value);

Form::password('password');

Form::email('name', $value, array());

Form::file('name', array());

Form::checkbox('name', 'value');

Form::radio('name', 'value');

Form::select('name', array('key' => 'value'));

Form::select('name', array('key' => 'value'), 'key');

Form::submit('Submit!');

Form::input(‘type’, ‘name’, ‘value’) // type can be number…

**Form Macros**

Form::macro('fooField', function()

{ return '<input type="custom"/>'; });

Form::fooField();

**HTML Builder**

HTML::macro('name', function(){});

HTML::entities($value);

HTML::decode($value);

HTML::script($url, $attributes);

HTML::style($url, $attributes);

HTML::link($url, 'title', $attributes, $secure);

HTML::secureLink($url, 'title', $attributes);

HTML::linkAsset($url, 'title', $attributes, $secure);

HTML::linkSecureAsset($url, 'title', $attributes);

HTML::linkRoute($name, 'title', $parameters, $attributes);

HTML::linkAction($action, 'title', $parameters, $attributes);

HTML::mailto($email, 'title', $attributes);

HTML::email($email);

HTML::ol($list, $attributes);

HTML::ul($list, $attributes);

HTML::listing($type, $list, $attributes);

HTML::listingElement($key, $type, $value);

HTML::nestedListing($key, $type, $value);

HTML::attributes($attributes);

HTML::attributeElement($key, $value);

HTML::obfuscate($value);

**Strings**

// Transliterate a UTF-8 value to ASCII

Str::ascii($value)

Str::camel($value)

Str::contains($haystack, $needle)

Str::endsWith($haystack, $needles)

// Cap a string with a single instance of a given value.

Str::finish($value, $cap)

Str::is($pattern, $value)

Str::length($value)

Str::limit($value, $limit = 100, $end = '...')

Str::lower($value)

Str::words($value, $words = 100, $end = '...')

Str::plural($value, $count = 2)

// Generate a more truly "random" alpha-numeric string.

Str::random($length = 16)

// Generate a "random" alpha-numeric string.

Str::quickRandom($length = 16)

Str::upper($value)

Str::title($value)

Str::singular($value)

Str::slug($title, $separator = '-')

Str::snake($value, $delimiter = '\_')

Str::startsWith($haystack, $needles)

// Convert a value to studly caps case.

Str::studly($value)

Str::macro($name, $macro)

**Helpers**

**Arrays**

array\_add($array, 'key', 'value');

array\_build($array, function(){});

array\_divide($array);

array\_dot($array);

array\_except($array, array('key'));

array\_fetch($array, 'key');

array\_first($array, function($key, $value){}, $default);

// Strips keys from the array

array\_flatten($array);

array\_forget($array, 'foo');

// Dot notation

array\_forget($array, 'foo.bar');

array\_get($array, 'foo', 'default');

array\_get($array, 'foo.bar', 'default');

array\_only($array, array('key'));

// Return array of key => values

array\_pluck($array, 'key');

// Return and remove 'key' from array

array\_pull($array, 'key');

array\_set($array, 'key', 'value');

// Dot notation

array\_set($array, 'key.subkey', 'value');

array\_sort($array, function(){});

// First element of an array

head($array);

// Last element of an array

last($array);

**Paths**

app\_path();

public\_path();

// App root path

base\_path();

storage\_path();

**Strings**

camel\_case($value);

class\_basename($class);

// Escape a string

e('<html>');

starts\_with('Foo bar.', 'Foo');

ends\_with('Foo bar.', 'bar.');

snake\_case('fooBar');

str\_contains('Hello foo bar.', 'foo');

// Result: foo/bar/

str\_finish('foo/bar', '/');

str\_is('foo\*', 'foobar');

str\_plural('car');

str\_random(25);

str\_singular('cars');

// Result: FooBar

studly\_case('foo\_bar');

trans('foo.bar');

trans\_choice('foo.bar', $count);

**URLs and Links**

action('FooController@method', $parameters);

link\_to('foo/bar', $title, $attributes, $secure);

link\_to\_asset('img/foo.jpg', $title, $attributes, $secure);

link\_to\_route('route.name', $title, $parameters, $attributes);

link\_to\_action('FooController@method', $title, $params, $attrs);

// HTML Link

asset('img/photo.jpg', $title, $attributes);

// HTTPS link

secure\_asset('img/photo.jpg', $title, $attributes);

secure\_url('path', $parameters);

route($route, $parameters, $absolute = true);

url('path', $parameters = array(), $secure = null);

**Miscellaneous**

csrf\_token();

dd($value);

value(function(){ return 'bar'; });

with(new Foo)->chainedMethod();

[Google Docs Document Source](https://docs.google.com/document/d/1NeCLL4fOgAmm4pYHIcBeWgCSCem5pqnGcwD_j4sNZyc/edit?usp=sharing) :

https://docs.google.com/document/d/1NeCLL4fOgAmm4pYHIcBeWgCSCem5pqnGcwD4sNZyc/edit?usp=sharing\_j