Table of Contents

xxii.
xxiii.

7.
    i.
    ii.
    iii. Eloquent ORM
    iv.
    v.
    vi. Redis
8. Artisan
    i.
    ii.

Laravel5

Laravel5 `laravel-china`
laravel-china.org

issue pull request

laravel-china.org PHPHub PHP & Laravel

Laravel @appleboy , , PHPHub @NauxLiu

@lifesign
@NauxLiu
@summerblue
@appleboy
@zhangxuanming
@zhuzhichao
@frankyang4

Laravel 5.0

Laravel 5.0 Laravel PSR-4

`app/models` `app` `App` `app:name` Artisan

controller middleware requestsLaravel 5.0 `app/Http` HTTP

`app/Providers` Laravel 4.x `app/start`

`resources`

Contracts

Laravel `illuminate/contracts` Laravel Facades

contracts

`route:cache` Artisan 100

Middleware

Laravel 4 filters Laravel 5 HTTP CSRF

type-hint

```
public function createPost(Request $request, PostRepository $posts)
{
    //
}
```

`resources/views/auth` users `auth/login`

`auth/register` `App\Services\Auth\Registrar`

```
class PodcastWasPurchased {

    public $podcast;

    public function __construct(Podcast $podcast)
    {
        $this->podcast = $podcast;
    }

}
```

```
Event::fire(new PodcastWasPurchased($podcast));
```

```
class ReportPodcastPurchase {

    public function handle(PodcastWasPurchased $event)
    {
        //
    }

}
```

Commands Queueing

Laravel 4 Laravel 5 `app/Commands`

```
class PurchasePodcast extends Command implements SelfHandling, ShouldBeQueued {

    use SerializesModels;

    protected $user, $podcast;

    /**
     * Create a new command instance.
     *
     * @return void
     */
    public function __construct(User $user, Podcast $podcast)
    {
        $this->user = $user;
        $this->podcast = $podcast;
```

```
    }

    /**
     * Execute the command.
     *
     * @return void
     */
    public function handle()
    {
        // Handle the logic to purchase the podcast...

        event(new PodcastWasPurchased($this->user, $this->podcast));
    }

}
```

Laravel `DispatchesCommands` trait

```
$this->dispatch(new PurchasePodcastCommand($user, $podcast));
```

command bus

`database` Laravel

Laravel Scheduler

Cron SSH Cron

, Laravel Cron

```
$schedule->command('artisan:command')->dailyAt('15:00');
```

TinkerPsysh

`php artisan tinker` Justin Hileman Psysh PHP REPL Laravel 4 Boris Psysh Windows

```
php artisan tinker
```

DotEnv

Laravel 5 Vance Lucas DotEnv Laravel 5

Laravel Elixir

Jeffrey Way Laravel Elixir assets Grunt Gulp Elixir Gulp LessSass CoffeeScript

Elixir

Laravel Socialite

Laravel Socialite Laravel 5.0 OAuth Socialite FacebookTwitterGoogle GitHub

```
public function redirectForAuth()
{
    return Socialize::with('twitter')->redirect();
}

public function getUserFromProvider()
{
    $user = Socialize::with('twitter')->user();
}
```

OAuth

Flysystem

Laravel FlysystemAmazon S3 Rackspace API Amazon S3

```
Storage::put('file.txt', 'contents');
```

Laravel

Form Requests

Laravel 5.0 **form requests** `Illuminate\Foundation\Http\FormRequest` request `FormRequest`

```
<?php namespace App\Http\Requests;

class RegisterRequest extends FormRequest {

    public function rules()
    {
        return [
            'email' => 'required|email|unique:users',
            'password' => 'required|confirmed|min:8',
        ];
    }

    public function authorize()
    {
        return true;
```

```
    }

}
```

```
public function register(RegisterRequest $request)
{
    var_dump($request->input());
}
```

Laravel `FormRequest` HTTP form request HTTP session JSON `FormRequest`

Laravel 5 `ValidatesRequests` trait trait `validate` `FormRequests`

```
public function createPost(Request $request)
{
    $this->validate($request, [
        'title' => 'required|max:255',
        'body' => 'required',
    ]);
}
```

HTTP session AJAX Laravel JSON

Generators

Artisan generator `php artisan list`

`config:cache`

Symfony VarDumper

`dd` Symfony VarDumper

```
dd([1, 2, 3]);
```

Laravel 4.2

4.2 `php artisan changes` Github

> : 4.2 bug 4.1 Laravel 4.1

PHP 5.4

Laravel 4.2 PHP 5.4 PHP PHP traits Laravel PHP 5.4 PHP 5.3

Laravel Forge

Larvel Forge PHP LinodeDigitalOceanRackspace Amazon EC2 nginx SSH Cron job NewRelic & Papertrail Laravel queue worker Forge Laravel

Laravel 4.2 `app/config/database.php` Forge

Laravel Forge Forge

Laravel Homestead

Laravel Homestead Laravel PHP Vagrant Homestead Nginx 1.6PHP 5.5.12 MySQLPostresRedisMemcachedBeanstalkNodeGulpGrunt BowerHomestead `Homestead.yaml` Laravel

Laravel 4.2 `app/config/local/database.php` Homestead Laravel

Homestead

Laravel

Laravel Stripe Laravel Stripe API

Queue Workers

Artisan `queue:work` `--daemon` worker worker CPU

Queue Workers queue

Mail API Drivers

Laravel 4.2 `Mail` Mailgun Mandrill API SMTP Guzzle 4 HTTP

Traits

PHP 5.4 `traits`

: Eloquent documentation.

(auth) & Remindable Traits

PHP 5.4 traits `User`

""

`simplePaginate` Eloquent

`--force`

Laravel 4.1

4.1 `php artisan changes` Github

SSH

`SSH` SSH SSH

`php artisan tail` SSH `tail`

Boris In Tinker

Boris REPL `php artisan thinker` `readline` `pcntl` PHP 4.0

Eloquent

Eloquent `hasManyThrough` Eloquent

`whereHas`

Query Builder Eloquent

`queue:listen`

`queue:listen` `--tries`

Laravel 4.1 API 4.0 100% Symfony Routing

Session

Session Session Symfony Session

Doctrine DBAL

`renameColumn` `composer.json` `doctrine/dbal` Laravel

- 4.2 5.0
- 4.1 4.2
- 4.1.x 4.1.29
- 4.1.25 4.1.26
- 4.0 4.1

4.2 5.0

Laravel `5.0` `4.2` Eloquent Artisan Asset

Laravel 5

Composer

Composer 5.0 (SDKs)

Laravel 5 Laravel 5 Composer `composer update`

Laravel 4 Eloquent Laravel 5

`.env.example` `.env` `5.0` `.env.php` `APP_ENV` `APP_KEY` () session

`.env.php` `.env` () `.env.example` ()

> : Laravel 5 `.env`

Laravel 5.0 `app/config/{environmentName}/` `.env` `env('key', 'default value')` `config/database.php`

`config/` `env()`

`.env` key `.env.example` `.env`

`routes.php` `app/Http/routes.php`

`app/Http/Controllers` `app/Http/Controllers` `composer.json` `classmap` `app/Http/Controllers/Controller.php`

`app/Providers/RouteServiceProvider.php` `namespace` `null`

`app/filters.php` `app/Providers/RouteServiceProvider.php` `boot()` `app/Providers/RouteServiceProvider.php` `use Illuminate\Support\Facades\Route;` `Route` Facade

Laravel 4.0 `auth` `csrf` (`['before' => 'auth']`) (`['middleware' => 'auth'].`)

Laravel 5 `before` `after`

CSRF

CSRF `App\Http\Kernel` `middleware`

```
'App\Http\Middleware\VerifyCsrfToken',
```

`$routeMiddleware` :

```
'csrf' => 'App\Http\Middleware\VerifyCsrfToken',
```

`['middleware' => 'csrf']` /

Eloquent

`app/Models` Eloquent `composer.json` `classmap`

`SoftDeletingTrait` `Illuminate\Database\Eloquent\SoftDeletes` .

Eloquent

Eloquent `remember` `Cache::remember`

Laravel 5 `User`

`use`

```
use Illuminate\Auth\UserInterface;
use Illuminate\Auth\Reminders\RemindableInterface;
```

`use`

```
use Illuminate\Auth\Authenticatable;
use Illuminate\Auth\Passwords\CanResetPassword;
use Illuminate\Contracts\Auth\Authenticatable as AuthenticatableContract;
use Illuminate\Contracts\Auth\CanResetPassword as CanResetPasswordContract;
```

UserInterface RemindableInterface

```
implements AuthenticatableContract, CanResetPasswordContract
```

traits

```
use Authenticatable, CanResetPassword;
```

traits use `Illuminate\Auth\Reminders\RemindableTrait` `Illuminate\Auth\UserTrait`

Cashier

Laravel Cashier trait trait `Laravel\Cashier\Billable` `BillableTrait` `Laravel\Cashier\Contracts\Billable` `Larave\Cashier\BillableInterface`

Artisan

`app/commands` `app/Console/Commands` `app/Console/Commands` `composer.json` `classmap`

Artisan `start/artisan.php` `app/Console/Kernel.php` `command`

users Laravel 5

`app/database/migrations` `database/migrations` `app/database/seeds` `database/seeds`

IoC

`start/global.php` IoC `app/Providers/AppServiceProvider.php` `register` `App` facade

`app/views` `resources/views`

Blade

Laravel 5.0 `{{ }}` `{{{ }}}` `{!! !!}` `{!! !!}`

Blade `AppServiceProvider@register`

```
\Blade::setRawTags('{{', '}}');
\Blade::setContentTags('{{{', '}}}');
\Blade::setEscapedContentTags('{{{', '}}}');
```

XSS `{{--`

`app/lang` `resources/lang`

`4.2` `public` `5.0` `index.php`

`app/tests` `tests`

`.scrutinizer.yml` , `bower.json`

SassLess CoffeeScript `resources/assets`

HTML

HTML `class 'Form' not found` `class 'Html' not found` Form HTML Laravel 5.0 Laravel Collective

`composer.json` `require` `"laravelcollective/html": "~5.0"`

HTML facades `config/app.php` 'providers'

```
'Collective\Html\HtmlServiceProvider',
```

'aliases'

```
'Form' => 'Collective\Html\FormFacade',
```

```
'Html' => 'Collective\Html\HtmlFacade',
```

`Illuminate\Cache\CacheManager` Facade Laravel `Illuminate\Contracts\Cache\Repository`

`$paginator->links()` `$paginator->render()`

Beanstalk

Laravel 5.0 `"pda/pheanstalk": "~3.0"` `"pda/pheanstalk": "~2.1"`

Remote

Remote

4.1 4.2

PHP 5.4+

Laravel 4.2 PHP 5.4.0

`cipher` `app/config/app.php` `MCRYPT_RIJNDAEL_256`

```
'cipher' => MCRYPT_RIJNDAEL_256
```

Laravel

> : Laravel 4.2 `MCRYPT_RIJNDAEL_128` (AES) `MCRYPT_RIJNDAEL_256` Laravel <= 4.1 cookies/values

`softDeletes` `SoftDeletingTrait`

```
use Illuminate\Database\Eloquent\SoftDeletingTrait;

class User extends Eloquent {
    use SoftDeletingTrait;
```

```
}
```

`deleted_at` `dates`

```
class User extends Eloquent {
    use SoftDeletingTrait;

    protected $dates = ['deleted_at'];
}
```

API

> : `SoftDeletingTrait`

/ /

`Illuminate\View\Environment` `Illuminate\Pagination\Environment` `Illuminate\View\Factory` `Illuminate\Pagination\Factory`

Additional Parameter On Pagination Presenter

`Illuminate\Pagination\Presenter` `getPageLinkWrapper` `rel`

```
abstract public function getPageLinkWrapper($url, $page, $rel = null);
```

Iron.Io Queue

Iron.io queue `encrypt` queue

```
'encrypt' => true
```

4.1.x 4.1.29

Laravel 4.1.29 column quoting `fillable` mass assignment `fillable` mass assignment `guarded` `4.1.29` mass assign

Laravel 4.1.29 `composer update`

4.1.25 4.1.26

Laravel 4.1.26 cookies cookies cookie

`users` () `remember_token` token token cookie cookie

VARCHAR(100)TEXT `remember_token` `users`

Eloquent `User`

```
public function getRememberToken()
{
    return $this->remember_token;
}

public function setRememberToken($value)
{
    $this->remember_token = $value;
}

public function getRememberTokenName()
{
    return 'remember_token';
}
```

> : sessions

`Illuminate\Auth\UserProviderInterface`

```
public function retrieveByToken($identifier, $token);

public function updateRememberToken(UserInterface $user, $token);
```

`Illuminate\Auth\UserInterface`

4.0 4.1

Composer

Laravel 4.1 `composer.json` `laravel/framework` `4.1.*`

`public/index.php` repository

`artisan` repository

`app/config/app.php` `aliases` `providers` providers / aliases

repository `app/config/remote.php`

`app/config/session.php` `expire_on_close` `false`

`app/config/queue.php` `failed`

```
'failed' => array(
    'database' => 'mysql', 'table' => 'failed_jobs',
),
```

`app/config/view.php` `pagination` `pagination::slider-3`

Controllers

`app/controllers/BaseController.php` `use` `use Illuminate\Routing\Controllers\Controller;` `use Illuminate\Routing\Controller;`

Artisan `php artisan auth:reminders-controller`

`app/lang/en/reminders.php`

hostname Mac & Ubuntu `hostname`

VirtualHost

Laravel `app/storage/logs/laravel.log` `app/start/global.php`

`bootstrap/start.php` `$app->redirectIfTrailingSlash()` `.htaccess`

Apache `.htaccess`

`Route::getCurrentRoute()` `Route::current()`

Composer

`composer update` class load `update` `--no-scripts` `composer update --no-scripts`

`Event::firing()`.

-
-
- ?
-
-

Laravel GitHub Pull Request Pull Request

Laravel Github Laravel :

- Laravel Framework
- Laravel Application
- Laravel Documentation
- Laravel Cashier
- Laravel Envoy
- Laravel Homestead
- Laravel Homestead Build Scripts
- Laravel Website
- Laravel Art

(Freenode) #laravel-dev IRC . Laravel Taylor Otwell 8am-5am (UTC-06:00 or America/Chicago)

#laravel-dev IRC !

?

`master`

Laravel

Laravel `master`

(Freenode) #laravel-dev IRC Taylor Otwell

Laravel Taylor Otwell taylorotwell@gmail.com

Laravel PSR-4 PSR-1 :

- `<?php`
- `{`
- Allman
-

Laravel

Composer

Laravel Composer Laravel Composer

Laravel

Laravel

Composer Laravel

```
composer global require "laravel/installer=~1.1"
```

`~/.composer/vendor/bin` `PATH` `laravel`

`laravel new` `Laravel` `laravel new blog` `blog` Laravel `composer install`

```
laravel new blog
```

Composer Create-Project

Composer `create-project` Laravel

```
composer create-project laravel/laravel --prefer-dist
```

Laravel `fresh`

php artisan fresh

Laravel

- PHP >= 5.4
- Mcrypt PHP
- OpenSSL PHP

- Mbstring PHP

PHP 5.5 PHP JSON Ubuntu `apt-get install php5-json`

Laravel comoser Laravel `key:generate`

32 `.env` **sessions**

Laravel `config/app.php`

Laravel

> `app.debug` `true`

Laravel `storage`

Apache

Laravel `public/.htaccess` `index.php` Apache `mod_rewrite`

Laravel `.htaccess` Apache

```
Options +FollowSymLinks
RewriteEngine On

RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^ index.php [L]
```

Nginx

Nginx

```
location / {
    try_files $uri $uri/ /index.php?$query_string;
}
```

Homestead

-
-
-
-
-
-
-

Laravel `config`

Laravel `app` `App` Composer PSR-4 Artisan `app:name`

Horsefly

```
php artisan app:name Horsefly
```

`App`

Laravel `config/app.php`

Laravel

`app.debug` `true`

Laravel `storage`

`Config` facade

```
$value = Config::get('app.timezone');

Config::set('app.timezone', 'America/Chicago');
```

`config`

```
$value = config('app.timezone');
```

---

cache driver

Laravel DotEnv Vance Lucas PHP Laravel `.env.example` Composer Laravel `.env`

`$_ENV` PHP `env` Laravel

`.env`

`.env.example`

`Application` `environment`

```
$environment = $app->environment();
```

`environment`

```
if ($app->environment('local'))
{
    // The environment is local
}

if ($app->environment('local', 'staging'))
{
    // The environment is either local OR staging...
}
```

`Illuminate\Contracts\Foundation\Application` contract `$this->app`

`App` facade `app`

```
$environment = app()->environment();

$environment = App::environment();
```

Artisan `config:cache`

`config:cache`

`HttpException` 503

Artisan `down`

```
php artisan down
```

Artisan `up`

```
php artisan up
```

`resources/views/errors/503.blade.php`

Apache

Laravel `public/.htaccess` `index.php` Apache `mod_rewrite`

Laravel `.htaccess` Apache

```
Options +FollowSymLinks
RewriteEngine On

RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^ index.php [L]
```

Nginx

Nginx

```
location / {
    try_files $uri $uri/ /index.php?$query_string;
}
```

Homestead

Laravel Homestead

-
-
-
-
-
- Blackfire Profiler

Laravel PHP Vagrant

Laravel Homestead Vagrant PHPHHVMVagrant

Homestead WindowsMac Linux Nginx PHP 5.6MySQLPostgresRedis
Memcached Laravel

> Windows VT-x BIOS

Homestead Vagrant 1.6

- Ubuntu 14.04
- PHP 5.6
- HHVM
- Nginx
- MySQL
- Postgres
- Node (With Bower, Grunt, and Gulp)
- Redis
- Memcached
- Beanstalkd
- Laravel Envoy
- Fabric + HipChat Extension
- Blackfire Profiler

VirtualBox Vagrant

Homestead VirtualBox Vagrant.

Vagrant

VirtualBox Vagrant 'laravel/homestead' Vagrant :

```
vagrant box add laravel/homestead
```

, Vagrant URL

```
vagrant box add laravel/homestead https://atlas.hashicorp.com/laravel/boxes/homestead
```

Homestead

Git PHP

PHP Homestead "home" `Homestead` Homestead Laravel PHP:

```
git clone https://github.com/laravel/homestead.git Homestead
```

Homestead CLI `bash init.sh` `Homestead.yaml` :

```
bash init.sh
```

`Homestead.yaml` `~/.homestead`

Composer + PHP

Vagrant Composer `global` Homestead CLI

```
composer global require "laravel/homestead=~2.0"
```

`homestead` `~/.composer/vendor/bin` `homestead`

Homestead CLI `init` `Homestead.yaml` :

```
homestead init
```

`Homestead.yaml` `~/.homestead` Mac Linux `homestead edit` `Homestead.yaml`
:

```
homestead edit
```

SSH

`Homestead.yaml` SSH Homestead

SSH Mac Linux SSH :

```
ssh-keygen -t rsa -C "you@homestead"
```

Windows Git Git `Git Bash` PuTTY PuTTYgen

SSH `Homestead.yaml` `authorize`

`Homestead.yaml` `folders` Homestead Homestead

NFS `folders`

```
folders:
    - map: ~/Code
      to: /home/vagrant/Code
      type: "nfs"
```

Nginx

Nginx `sites` homestead `Homestead.yaml` Homestead
Homestead Laravel

`hhvm` `true` HHVM:

```
sites:
    - map: homestead.app
      to: /home/vagrant/Code/Laravel/public
      hhvm: true
```

Bash Aliases

Bash aliases Homestead `~/.homestead` `aliases`

Vagrant

`Homestead.yaml` Homestead `homestead up`

Vagrant Nginx `vagrant destroy --force`

Nginx `hosts` `hosts` Homestead Mac Linux `/etc/hosts`
Windows `C:\Windows\System32\drivers\etc\hosts`

```
192.168.10.10  homestead.app
```

IP `Homestead.yaml` `hosts`

```
http://homestead.app
```

SSH

SSH Homestead Homestead `vagrant ssh`

SSH Homestead "":

```
alias vm="ssh vagrant@127.0.0.1 -p 2222"
```

"vm" SSH Homestead

`Homestead` MySQL Postgres Laravel `local`

Navicat Sequel Pro MySQL Postgres `127.0.0.1` 33060 (MySQL) 54320 (Postgres) `homestead` / `secret`

> Laravel Laravel 3306 5432

Homestead Laravel Nginx Homestead Laravel `Homestead.yaml` `homestead provision` `vagrant provision`

Homestead `serve` `serve` SSH Homestead

```
serve domain.app /home/vagrant/Code/path/to/public/directory
```

> `serve` `hosts`

Homestead

- **SSH:** 2222 → Forwards To 22
- **HTTP:** 8000 → Forwards To 80
- **MySQL:** 33060 → Forwards To 3306
- **Postgres:** 54320 → Forwards To 5432

Vagrant box

```
ports:
    - send: 93000
      to: 9300
    - send: 7777
      to: 777
      protocol: udp
```

Blackfire Profiler

Blackfire Profiler SensioLabs RAM, CPU time, disk I/O. Homestead

blackfire Homestead box `Homestead.yaml` Server ID token

```
blackfire:
    - id: your-id
      token: your-token
```

Blackfire `homestead provision` `vagrant provision` Blackfire Blackfire

HTTP

`app/Http/routes.php` `App\Providers\RouteServiceProvider` Laravel URI `(Closure)`

GET

```
Route::get('/', function()
{
    return 'Hello World';
});
```

```
Route::post('foo/bar', function()
{
    return 'Hello World';
});

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

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

```
Route::match(['get', 'post'], '/', function()
{
    return 'Hello World';
});
```

HTTP

```
Route::any('foo', function()
{
    return 'Hello World';
});
```

URL `url`

```
$url = url('foo');
```

CSRF

Laravel CSRF ()

Laravel session `token` token

CSRF Token

```
<input type="hidden" name="_token" value="<?php echo csrf_token(); ?>">
```

Blade

```
<input type="hidden" name="_token" value="{{ csrf_token() }}">
```

POSTPUTDELETE CSRF token `VerifyCsrfToken` HTTP session token token

X-CSRF-TOKEN

CSRF token POST `X-XSRF-TOKEN` token meta , jQuery

```
<meta name="csrf-token" content="{{ csrf_token() }}" />

$.ajaxSetup({
    headers: {
        'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
    }
});
```

AJAX CSRF token

```
$.ajax({
  url: "/foo/bar",
})
```

X-XSRF-TOKEN

Laravel cookie `XSRF-TOKEN` CSRF token cookie `X-XSRF-TOKEN` Javascript Angular

> `X-CSRF-TOKEN` `X-XSRF-TOKEN` Laravel cookies `csrf_token()` token
> `X-CSRF-TOKEN`

HTML `PUT` `DELETE` `PUT` `DELETE` HTML `_method`

`_method` HTTP

```
<form action="/foo/bar" method="POST">
    <input type="hidden" name="_method" value="PUT">
    <input type="hidden" name="_token" value="<?php echo csrf_token(); ?>">
</form>
```

URI

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

```
Route::get('user/{name?}', function($name = null)
{
    return $name;
});
```

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

```
Route::get('user/{name}', function($name)
{
    //
})
->where('name', '[A-Za-z]+');

Route::get('user/{id}', function($id)
{
```

```
    //
})
->where('id', '[0-9]+');
```

```
Route::get('user/{id}/{name}', function($id, $name)
{
    //
})
->where(['id' => '[0-9]+', 'name' => '[a-z]+'])
```

`pattern` `RouteServiceProvider` `boot`

```
$router->pattern('id', '[0-9]+');
```

```
Route::get('user/{id}', function($id)
{
    //  {id}
});
```

`input`

```
if ($route->input('id') == 1)
{
    //
}
```

`Illuminate\Http\Request` `Request` facade `Illuminate\Http\Request`

```
use Illuminate\Http\Request;

Route::get('user/{id}', function(Request $request, $id)
{
    if ($request->route('id'))
    {
        //
    }
});
```

URL `as`

```
Route::get('user/profile', ['as' => 'profile', function()
{
    //
}]);
```

```
Route::get('user/profile', [
    'as' => 'profile', 'uses' => 'UserController@showProfile'
]);
```

URL

```
$url = route('profile');

$redirect = redirect()->route('profile');
```

`currentRouteName`

```
$name = Route::currentRouteName();
```

```
Route::group(['middleware' => 'auth'], function()
{
    Route::get('/', function()
    {
        // Has Auth Filter
    });

    Route::get('user/profile', function()
    {
        // Has Auth Filter
    });
});
```

`group` `namespace`

```
Route::group(['namespace' => 'Admin'], function()
{
    //
});
```

> `RouteServiceProvider` `routes.php`

Laravel

```
Route::group(['domain' => '{account}.myapp.com'], function()
{

    Route::get('user/{id}', function($account, $id)
    {
        //
    });

});
```

`prefix`

```
Route::group(['prefix' => 'admin'], function()
{

    Route::get('user', function()
    {
        //
    });

});
```

Laravel User ID ID User

`model` `RouteServiceProvider::boot`

```
public function boot(Router $router)
{
    parent::boot($router);

    $router->model('user', 'App\User');
}
```

`{user}`

```
Route::get('profile/{user}', function(App\User $user)
{
    //
});
```

`{user}` `App\User` `User` `profile/1` ID 1 `User`

> 404

`model`

```
Route::model('user', 'User', function()
{
    throw new NotFoundHttpException;
});
```

`Router::bind` `bind` URI

```
Route::bind('user', function($value)
{
    return User::where('name', $value)->first();
});
```

404

404 `abort`

```
abort(404);
```

`abort` `Symfony\Component\HttpFoundation\Exception\HttpException`

`Symfony\Component\HttpKernel\Exception\NotFoundHttpException`

404

HTTP

-
-
-
-

HTTP HTTP Laravel

CORS Laravel CSRF `app/Http/Middleware`

`make:middleware` Artisan

```
php artisan make:middleware OldMiddleware
```

`app/Http/Middleware` `OldMiddleware` 200 home URI

```
<?php namespace App\Http\Middleware;

class OldMiddleware {

    /**
     * Run the request filter.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @return mixed
     */
    public function handle($request, Closure $next)
    {
        if ($request->input('age') < 200)
        {
            return redirect('home');
        }

        return $next($request);
    }

}
```

 `200` HTTP `$request` `$next` () HTTP

Before / After

```php
<?php namespace App\Http\Middleware;

class BeforeMiddleware implements Middleware {

    public function handle($request, Closure $next)
    {
        // Perform action

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

```php
<?php namespace App\Http\Middleware;

class AfterMiddleware implements Middleware {

    public function handle($request, Closure $next)
    {
        $response = $next($request);

        // Perform action

        return $response;
    }
}
```

HTTP `app/Http/Kernel.php` `$middleware`

`app/Http/Kernel.php` `$routeMiddleware` Laravel HTTP kernel `middleware`

```php
Route::get('admin/profile', ['middleware' => 'auth', function()
{
    //
}]);
```

HTTP Laravel session session

```php
use Illuminate\Contracts\Routing\TerminableMiddleware;

class StartSession implements TerminableMiddleware {
```

```
    public function handle($request, $next)
    {
        return $next($request);
    }

    public function terminate($request, $response)
    {
        // Store the session data...
    }

}
```

`handle` `TerminableMiddleware` `terminate` terminable HTTP kernel

HTTP

-
-
-
-
- RESTful
-
-

`routes.php` HTTP `app/Http/Controllers`

```
<?php namespace App\Http\Controllers;

use App\Http\Controllers\Controller;

class UserController extends Controller {

    /**
     *
     *
     * @param  int  $id
     * @return Response
     */
    public function showProfile($id)
    {
        return view('user.profile', ['user' => User::findOrFail($id)]);
    }

}
```

```
Route::get('user/{id}', 'UserController@showProfile');
```

`App\Http\Controllers` `RouteServiceProvider` `routes.php`

`App\Http\Controllers` PHP `App\Http\Controllers`
`App\Http\Controllers\Photos\AdminController`

```
Route::get('foo', 'Photos\AdminController@method');
```

```
Route::get('foo', ['uses' => 'FooController@method', 'as' => 'name']);
```

URL

URL `action`

```
$url = action('App\Http\Controllers\FooController@method');
```

URL URL

```
URL::setRootControllerNamespace('App\Http\Controllers');

$url = action('FooController@method');
```

`currentRouteAction`

```
$action = Route::currentRouteAction();
```

```
Route::get('profile', [
    'middleware' => 'auth',
    'uses' => 'UserController@showProfile'
]);
```

```
class UserController extends Controller {

    /**
     *  UserController
     */
    public function __construct()
    {
        $this->middleware('auth');

        $this->middleware('log', ['only' => ['fooAction', 'barAction']]);

        $this->middleware('subscribed', ['except' => ['fooAction', 'barAction']]);
```

```
    }

}
```

Laravel `Route::controller`

```
Route::controller('users', 'UserController');
```

`Controller` base URI HTTP

```
class UserController extends BaseController {

    public function getIndex()
    {
        //
    }

    public function postProfile()
    {
        //
    }

    public function anyLogin()
    {
        //
    }

}
```

`index` URI `users`

URI `UserController` `users/admin-profile` URI

```
public function getAdminProfile() {}
```

“” `controller`

```
Route::controller('users', 'UserController', [
    'anyLogin' => 'user.login',
]);
```

RESTful

RESTful HTTP `make:controller` Artisan

```
php artisan make:controller PhotoController
```

```
Route::resource('photo', 'PhotoController');
```

RESTful URI

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

```
Route::resource('photo', 'PhotoController',
                ['only' => ['index', 'show']]);

Route::resource('photo', 'PhotoController',
                ['except' => ['create', 'store', 'update', 'destroy']]);
```

`names`

```
Route::resource('photo', 'PhotoController',
                ['names' => ['create' => 'photo.build']]);
```

```
Route::resource('photos.comments', 'PhotoCommentController');
```

`photos/{photos}/comments/{comments}` URL

```
class PhotoCommentController extends Controller {

    /**
```

```
     *
     *
     * @param  int  $photoId
     * @param  int  $commentId
     * @return Response
     */
    public function show($photoId, $commentId)
    {
        //
    }

}
```

`Route::resource`

```
Route::get('photos/popular', 'PhotoController@method');

Route::resource('photos', 'PhotoController');
```

Laravel Laravel

```
<?php namespace App\Http\Controllers;

use Illuminate\Routing\Controller;
use App\Repositories\UserRepository;

class UserController extends Controller {

    /**
     *
     */
    protected $users;

    /**
     *
     *
     * @param  UserRepository  $users
     * @return void
     */
    public function __construct(UserRepository $users)
    {
        $this->users = $users;
    }

}
```

Laravel contract

`Request`

```
<?php namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Routing\Controller;

class UserController extends Controller {

    /**
     *
     *
     * @param  Request  $request
     * @return Response
     */
    public function store(Request $request)
    {
        $name = $request->input('name');

        //
    }

}
```

```
<?php namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Routing\Controller;

class UserController extends Controller {

    /**
     *
     *
     * @param  Request  $request
     * @param  int  $id
     * @return Response
     */
    public function update(Request $request, $id)
    {
        //
    }

}
```

Laravel 100 `route:cache` Artisan

```
php artisan route:cache
```

`app/Http/routes.php` `route:cache`

`route:clear`

```
php artisan route:clear
```

HTTP

-
-
-
- Cookies
-
-

Facade

`Request` facade

```
$name = Request::input('name');
```

`Request` facade `use Request;`

HTTP

```
<?php namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Routing\Controller;

class UserController extends Controller {

    /**
     * Store a new user.
     *
     * @param  Request  $request
     * @return Response
     */
    public function store(Request $request)
    {
        $name = $request->input('name');

        //
    }

}
```

```
<?php namespace App\Http\Controllers;

use Illuminate\Http\Request;
```

```
use Illuminate\Routing\Controller;

class UserController extends Controller {

    /**
     * Store a new user.
     *
     * @param  Request  $request
     * @param  int  $id
     * @return Response
     */
    public function update(Request $request, $id)
    {
        //
    }

}
```

`Illuminate\Http\Request` HTTP

```
$name = Request::input('name');
```

```
$name = Request::input('name', 'Sally');
```

```
if (Request::has('name'))
{
    //
}
```

```
$input = Request::all();
```

```
$input = Request::only('username', 'password');

$input = Request::except('credit_card');
```

```
$input = Request::input('products.0.name');
```

Laravel

Session

`flash` session

```
Request::flash();
```

Session

```
Request::flashOnly('username', 'email');

Request::flashExcept('password');
```

Session

```
return redirect('form')->withInput();

return redirect('form')->withInput(Request::except('password'));
```

Session `Request` `old`

```
$username = Request::old('username');
```

Blade `old`

```
{{ old('username') }}
```

Cookies

Laravel cookie cookie

Cookie

```
$value = Request::cookie('name');
```

Cookie

`cookie` `Symfony\Component\HttpFoundation\Cookie` `Response` `withCookie` cookie

```
$response = new Illuminate\Http\Response('Hello World');

$response->withCookie(cookie('name', 'value', $minutes));
```

Cookie*

```
$response->withCookie(cookie()->forever('name', 'value'));
```

```
$file = Request::file('photo');
```

```
if (Request::hasFile('photo'))
{
    //
}
```

`file` `Symfony\Component\HttpFoundation\File\UploadedFile` `UploadedFile` PHP `SplFileInfo`

```
if (Request::file('photo')->isValid())
{
    //
}
```

```
Request::file('photo')->move($destinationPath);

Request::file('photo')->move($destinationPath, $fileName);
```

`UploadedFile` API

`Request` HTTP `Symfony\Component\HttpFoundation\Request`

URI

```
$uri = Request::path();
```

```
$method = Request::method();

if (Request::isMethod('post'))
{
    //
}
```

```
if (Request::is('admin/*'))
{
    //
}
```

URL

```
$url = Request::url();
```

HTTP

-
-
-
-

Laravel

```
Route::get('/', function()
{
    return 'Hello World';
});
```

`Illuminate\Http\Response` `Response` HTTP `Response` `Symfony\Component\HttpFoundation\Response` HTTP

```
use Illuminate\Http\Response;

return (new Response($content, $status))
              ->header('Content-Type', $value);
```

`response`

```
return response($content, $status)
              ->header('Content-Type', $value);
```

> `Response` API Symfony API .

`Response` `view`

```
return response()->view('hello')->header('Content-Type', $type);
```

Cookies

```
return response($content)->withCookie(cookie('name', 'value'));
```

`Response`

```
return response()->view('hello')->header('Content-Type', $type)
                 ->withCookie(cookie('name', 'value'));
```

`Illuminate\Http\RedirectResponse` URL

`RedirectResponse` `redirect`

```
return redirect('user/login');
```

Flash Data

URL Session `RedirectResponse` Session

```
return redirect('user/login')->with('message', 'Login Failed');
```

URL

`back`

```
return redirect()->back();

return redirect()->back()->withInput();
```

`redirect` `Illuminate\Routing\Redirector` `RedirectResponse` `route`

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

`route`

```
//  URI profile/{id}

return redirect()->route('profile', [1]);
```

Eloquent IDID

```
return redirect()->route('profile', [$user]);
```

```
//  URI profile/{user}

return redirect()->route('profile', ['user' => 1]);
```

`RedirectResponse`

```
return redirect()->action('App\Http\Controllers\HomeController@index');
```

> `URL::setRootControllerNamespace` `action()`

```
return redirect()->action('App\Http\Controllers\UserController@profile', [1]);
```

```
return redirect()->action('App\Http\Controllers\UserController@profile', ['user' => 1]);
```

`response` `response` `Illuminate\Contracts\Routing\ResponseFactory` Contract Contract

JSON

`json` `Content-Type` `application/json`

```
return response()->json(['name' => 'Abigail', 'state' => 'CA']);
```

JSONP

```
return response()->json(['name' => 'Abigail', 'state' => 'CA'])
                 ->setCallback($request->input('callback'));
```

```
return response()->download($pathToFile);

return response()->download($pathToFile, $name, $headers);

return response()->download($pathToFile)->deleteFileAfterSend(true);
```

> Symfony HttpFoundation ASCII

`Illuminate\Contracts\Routing\ResponseFactory` `macro`

`boot` :

```
<?php namespace App\Providers;

use Response;
use Illuminate\Support\ServiceProvider;

class ResponseMacroServiceProvider extends ServiceProvider {

    /**
     * Perform post-registration booting of services.
     *
     * @return void
     */
    public function boot()
    {
        Response::macro('caps', function($value) use ($response)
        {
            return $response->make(strtoupper($value));
        });
    }

}
```

`macro` `ResponseFactory` `response`

```
return response()->caps('foo');
```

(View)

-
-

HTML `resources/views`

```
<!--  resources/views/greeting.php -->

<html>
    <body>
        <h1>Hello, <?php echo $name; ?></h1>
    </body>
</html>
```

```
Route::get('/', function()
{
    return view('greeting', ['name' => 'James']);
});
```

`view` `resources/views` `view`

`resources/views` `resources/views/admin/profile.php`

```
return view('admin.profile', $data);
```

```
//
$view = view('greeting')->with('name', 'Victoria');

//
$view = view('greeting')->withName('Victoria');
```

`$name` `Victoria`

`view`

```
$view = view('greetings', $data);
```

`view` `Illuminate\Contracts\View\Factory` (contract) (view composer)

`view`

```
view()->share('data', [1, 2, 3]);
```

`view` Facade

```
View::share('data', [1, 2, 3]);
```

`boot` `share` `AppServiceProvider`

> `view` `Illuminate\Contracts\View\Factory` (contract) (implementation)

`exists`

```
if (view()->exists('emails.customer'))
{
    //
}
```

```
return view()->file($pathToFile, $data);
```

`View` Facade `Illuminate\Contracts\View\Factory`

```
<?php namespace App\Providers;

use View;
use Illuminate\Support\ServiceProvider;

class ComposerServiceProvider extends ServiceProvider {

    /**
     * Register bindings in the container.
     *
```

```
     * @return void
     */
    public function boot()
    {
        //
        View::composer('profile', 'App\Http\ViewComposers\ProfileComposer');

        //
        View::composer('dashboard', function()
        {

        });
    }

    /**
     * Register
     *
     * @return void
     */
    public function register()
    {
        //
    }

}
```

> Laravel `App\Http\ViewComposers`

`config/app.php` `providers`

`profile` `ProfileComposer@compose`

```
<?php namespace App\Http\ViewComposers;

use Illuminate\Contracts\View\View;
use Illuminate\Users\Repository as UserRepository;

class ProfileComposer {

    /**
     * The user repository implementation.
     *
     * @var UserRepository
     */
    protected $users;

    /**
     * Create a new profile composer.
     *
     * @param  UserRepository  $users
     * @return void
     */
    public function __construct(UserRepository $users)
    {
        // service container
        $this->users = $users;
    }

    /**
     * Bind data to the view.
```

```
     *
     * @param  View  $view
     * @return void
     */
    public function compose(View $view)
    {
        $view->with('count', $this->users->count());
    }

}
```

`compose` `Illuminate\Contracts\View\View` `with` `view`

> (service container)

`View` `composer` `*` `composer`

```
View::composer('*', function()
{
    //
});
```

```
View::composer(['profile', 'dashboard'], 'App\Http\ViewComposers\MyViewComposer');
```

`composers`

```
View::composers([
    'App\Http\ViewComposers\AdminComposer' => ['admin.index', 'admin.profile'],
    'App\Http\ViewComposers\UserComposer' => 'user',
    'App\Http\ViewComposers\ProductComposer' => 'product'
]);
```

`creator`

```
View::creator('profile', 'App\Http\ViewCreators\ProfileCreator');
```

-
-
-
-

Laravel Laravel

Laravel `config/app.php` `providers`

Laravel

`Illuminate\Support\ServiceProvider` `register` `register` `register`

Artisan `make:provider`

```
php artisan make:provider RiakServiceProvider
```

```
<?php namespace App\Providers;

use Riak\Connection;
use Illuminate\Support\ServiceProvider;

class RiakServiceProvider extends ServiceProvider {

    /**
     *
     *
     * @return void
     */
    public function register()
    {
        $this->app->singleton('Riak\Contracts\Connection', function($app)
        {
            return new Connection($app['config']['riak']);
        });
    }

}
```

`register` `Riak\Contracts\Connection`

`App\Providers` Laravel Composer

`boot`

```
<?php namespace App\Providers;

use Event;
use Illuminate\Support\ServiceProvider;

class EventServiceProvider extends ServiceProvider {

    /**
     *
     *
     * @return void
     */
    public function boot()
    {
        Event::listen('SomeEvent', 'SomeEventHandler');
    }

    /**
     *
     *
     * @return void
     */
    public function register()
    {
        //
    }

}
```

`boot`

```
use Illuminate\Contracts\Events\Dispatcher;

public function boot(Dispatcher $events)
{
    $events->listen('SomeEvent', 'SomeEventHandler');
}
```

`config/app.php` `providers` Laravel Laravel

```
'providers' => [
    //

    'App\Providers\AppServiceProvider',
],
```

`defer` `true` `provides` `provides`

```
<?php namespace App\Providers;

use Riak\Connection;
use Illuminate\Support\ServiceProvider;

class RiakServiceProvider extends ServiceProvider {

    /**
     *
     *
     * @var bool
     */
    protected $defer = true;

    /**
     *
     *
     * @return void
     */
    public function register()
    {
        $this->app->singleton('Riak\Contracts\Connection', function($app)
        {
            return new Connection($app['config']['riak']);
        });
    }

    /**
     *
     *
     * @return array
     */
    public function provides()
    {
        return ['Riak\Contracts\Connection'];
    }

}
```

Laravel Laravel

-
-
-
-
-
-
-

Laravel setter

```
<?php namespace App\Handlers\Commands;

use App\User;
use App\Commands\PurchasePodcast;
use Illuminate\Contracts\Mail\Mailer;

class PurchasePodcastHandler {

    /**
     *
     */
    protected $mailer;

    /**
     *
     *
     * @param  Mailer  $mailer
     * @return void
     */
    public function __construct(Mailer $mailer)
    {
        $this->mailer = $mailer;
    }

    /**
     *
     *
     * @param  PurchasePodcastCommand  $command
     * @return void
     */
    public function handle(PurchasePodcastCommand $command)
    {
        //
    }

}
```

`PurchasePodcast`

Laravel Laravel

(context)(factory) `Illuminate\Contracts\Container\Container` `App` facade

`$this->app`

`$this->app`

key () :

```
$this->app->bind('FooBar', function($app)
{
    return new FooBar($app['SomethingElse']);
});
```

```
$this->app->singleton('FooBar', function($app)
{
    return new FooBar($app['SomethingElse']);
});
```

`instance`

```
$fooBar = new FooBar(new SomethingElse);

$this->app->instance('FooBar', $fooBar);
```

`make`

```
$fooBar = $this->app->make('FooBar');
```

PHP `ArrayAccess`

```
$fooBar = $this->app['FooBar'];
```

type-hint

```
<?php namespace App\Http\Controllers;

use Illuminate\Routing\Controller;
use App\Users\Repository as UserRepository;

class UserController extends Controller {

    /**
     * The user repository instance.
     */
    protected $users;

    /**
     * Create a new controller instance.
     *
     * @param  UserRepository  $users
     * @return void
     */
    public function __construct(UserRepository $users)
    {
        $this->users = $users;
    }

    /**
     * Show the user with the given ID.
     *
     * @param  int  $id
     * @return Response
     */
    public function show($id)
    {
        //
    }

}
```

Pusher Pusher PHP SDK Pusher

```
<?php namespace App\Handlers\Commands;

use App\Commands\CreateOrder;
use Pusher\Client as PusherClient;

class CreateOrderHandler {

    /**
     * Pusher SDK
     */
```

```
    protected $pusher;

    /**
     *
     *
     * @param  PusherClient  $pusher
     * @return void
     */
    public function __construct(PusherClient $pusher)
    {
        $this->pusher = $pusher;
    }

    /**
     *
     *
     * @param  CreateOrder  $command
     * @return void
     */
    public function execute(CreateOrder $command)
    {
        //
    }

}
```

Pusher SDK Pusher SDK `CreateOrderHandler`

`CreateOrderHandler` `EventPusher` `PusherEventPusher`

```
<?php namespace App\Contracts;

interface EventPusher {

    /**
     * Push a new event to all clients.
     *
     * @param  string  $event
     * @param  array  $data
     * @return void
     */
    public function push($event, array $data);

}
```

`PusherEventPusher`

```
$this->app->bind('App\Contracts\EventPusher', 'App\Services\PusherEventPusher');
```

`EventPusher` `PusherEventPusher` `EventPusher`

```
    /**
     * Create a new order handler instance.
```

```
     *
     * @param  EventPusher  $pusher
     * @return void
     */
    public function __construct(EventPusher $pusher)
    {
        $this->pusher = $pusher;
    }
```

PubNub Pusher Laravel

```
$this->app->when('App\Handlers\Commands\CreateOrderHandler')
          ->needs('App\Contracts\EventPusher')
          ->give('App\Services\PubNubEventPusher');
```

`Report` `Report` `tag`

```
$this->app->bind('SpeedReport', function()
{
    //
});

$this->app->bind('MemoryReport', function()
{
    //
});

$this->app->tag(['SpeedReport', 'MemoryReport'], 'reports');
```

`tagged`

```
$this->app->bind('ReportAggregator', function($app)
{
    return new ReportAggregator($app->tagged('reports'));
});
```

Laravel

```
<?php namespace App\Http\Controllers;

use Illuminate\Routing\Controller;
use App\Repositories\OrderRepository;

class OrdersController extends Controller {
```

```
    /**
     * The order repository instance.
     */
    protected $orders;

    /**
     * Create a controller instance.
     *
     * @param  OrderRepository  $orders
     * @return void
     */
    public function __construct(OrderRepository $orders)
    {
        $this->orders = $orders;
    }

    /**
     * Show all of the orders.
     *
     * @return Response
     */
    public function index()
    {
        $orders = $this->orders->all();

        return view('orders', ['orders' => $orders]);
    }

}
```

`OrderRepository` `OrderRepository`

Laravel

`resolving`

```
$this->app->resolving(function($object, $app)
{
    //
});

$this->app->resolving(function(FooBar $fooBar, $app)
{
    //  `FooBar`
});
```

Contracts

Laravel Contracts interfaces `Queue` contract `Mailer` contract e-mail

Laravel contract Laravel `Queue` SwiftMailer `Mailer`

Laravel contracts Github repository contracts

Contracts

contracts

```
<?php namespace App\Orders;

class Repository {

    /**
     * The cache.
     */
    protected $cache;

    /**
     * Create a new repository instance.
     *
     * @param  \SomePackage\Cache\Memcached  $cache
     * @return void
     */
    public function __construct(\SomePackage\Cache\Memcached $cache)
    {
        $this->cache = $cache;
    }

    /**
     * Retrieve an Order by ID.
     *
     * @param  int  $id
     * @return Order
     */
    public function find($id)
    {
```

```
        if ($this->cache->has($id))
        {
            //
        }
    }

}
```

package vendor API

Memcached Redis repository repository

```
<?php namespace App\Orders;

use Illuminate\Contracts\Cache\Repository as Cache;

class Repository {

    /**
     * Create a new repository instance.
     *
     * @param  Cache  $cache
     * @return void
     */
    public function __construct(Cache $cache)
    {
        $this->cache = $cache;
    }

}
```

Laravel contracts contract contracts

Laravel **contracts**

Contract

Laravel Contracts "facade"

Contract	Laravel 4.x Facade
Illuminate\Contracts\Auth\Guard	Auth
Illuminate\Contracts\Auth\PasswordBroker	Password
Illuminate\Contracts\Bus\Dispatcher	Bus
Illuminate\Contracts\Cache\Repository	Cache
Illuminate\Contracts\Cache\Factory	Cache::driver()
Illuminate\Contracts\Config\Repository	Config

Illuminate\Contracts\Container\Container	App
Illuminate\Contracts\Cookie\Factory	Cookie
Illuminate\Contracts\Cookie\QueueingFactory	Cookie::queue()
Illuminate\Contracts\Encryption\Encrypter	Crypt
Illuminate\Contracts\Events\Dispatcher	Event
Illuminate\Contracts\Filesystem\Cloud	
Illuminate\Contracts\Filesystem\Factory	File
Illuminate\Contracts\Filesystem\Filesystem	File
Illuminate\Contracts\Foundation\Application	App
Illuminate\Contracts\Hashing\Hasher	Hash
Illuminate\Contracts\Logging\Log	Log
Illuminate\Contracts\Mail\MailQueue	Mail::queue()
Illuminate\Contracts\Mail\Mailer	Mail
Illuminate\Contracts\Queue\Factory	Queue::driver()
Illuminate\Contracts\Queue\Queue	Queue
Illuminate\Contracts\Redis\Database	Redis
Illuminate\Contracts\Routing\Registrar	Route
Illuminate\Contracts\Routing\ResponseFactory	Response
Illuminate\Contracts\Routing\UrlGenerator	URL
Illuminate\Contracts\Support\Arrayable	
Illuminate\Contracts\Support\Jsonable	
Illuminate\Contracts\Support\Renderable	
Illuminate\Contracts\Validation\Factory	Validator::make()
Illuminate\Contracts\Validation\Validator	
Illuminate\Contracts\View\Factory	View::make()
Illuminate\Contracts\View\View	

Contracts

contract Laravel service container contract

```
<?php namespace App\Handlers\Events;

use App\User;
use App\Events\NewUserRegistered;
use Illuminate\Contracts\Redis\Database;

class CacheUserInformation {

    /**
     * Redis
     */
    protected $redis;

    /**
```

```
     *
     *
     * @param  Database  $redis
     * @return void
     */
    public function __construct(Database $redis)
    {
        $this->redis = $redis;
    }

    /**
     *
     *
     * @param  NewUserRegistered  $event
     * @return void
     */
    public function handle(NewUserRegistered $event)
    {
        //
    }

}
```

Facades

-
-
-
- Facades
- Facades
- Facade

Facades Laravel facadesLaravel facades IoC

facades

> facades Laravel IoC .

Laravel facade `Facade` Laravel facades facades `Facade`

facade `getFacadeAccessor` `getFacadeAccessor` `Facade` `__callStatic()` facade

facade `Cache::get` Laravel IoC `get` Laravel Facades Laravel IoC

Laravel `get` `Cache`

```
$value = Cache::get('key');
```

 `Illuminate\Support\Facades\Cache` `get`

```
class Cache extends Facade {

    /**
     *
     *
     * @return string
     */
    protected static function getFacadeAccessor() { return 'cache'; }

}
```

Cache `Facade` `getFacadeAccessor()` IoC

 `Cache` facade Laravel IoC `cache` (`get`)

`Cache::get`

```
$value = $app->make('cache')->get('key');
```

Facades

facade facade facades

```
<?php namespace App\Http\Controllers;

use Cache;

class PhotosController extends Controller {

    /**
     *
     *
     * @return Response
     */
    public function index()
    {
        $photos = Cache::get('photos');

        //
    }

}
```

Facades

facade 3

- IoC
- facade
- facade

`PaymentGateway\Payment`

```
namespace PaymentGateway;

class Payment {

    public function process()
    {
        //
    }

}
```

IoC

```
App::bind('payment', function()
```

```
{
    return new \PaymentGateway\Payment;
});
```

`PaymentServiceProvider` `register` Laravel `config/app.php`

facade

```
use Illuminate\Support\Facades\Facade;

class Payment extends Facade {

    protected static function getFacadeAccessor() { return 'payment'; }

}
```

`config/app.php` facade `aliases` `Payment` `process`

```
Payment::process();
```

`aliases` PHP `\ServiceWrapper\ApiTimeoutException` `ApiTimeoutException` `\ServiceWrapper` `catch(ApiTimeoutException $e)` `use`

Facades

facades facades facades

Facade

facade facade API IoC

Facade	Class	IoC Binding
App	Illuminate\Foundation\Application	`app`
Artisan	Illuminate\Console\Application	`artisan`
Auth	Illuminate\Auth\AuthManager	`auth`
Auth ()	Illuminate\Auth\Guard	
Blade	Illuminate\View\Compilers\BladeCompiler	`blade.compiler`
Bus	Illuminate\Contracts\Bus\Dispatcher	
Cache	Illuminate\Cache\Repository	`cache`
Config	Illuminate\Config\Repository	`config`
Cookie	Illuminate\Cookie\CookieJar	`cookie`
Crypt	Illuminate\Encryption\Encrypter	`encrypter`
DB	Illuminate\Database\DatabaseManager	`db`

DB ()	Illuminate\Database\Connection	
Event	Illuminate\Events\Dispatcher	`events`
File	Illuminate\Filesystem\Filesystem	`files`
Hash	Illuminate\Contracts\Hashing\Hasher	`hash`
Input	Illuminate\Http\Request	`request`
Lang	Illuminate\Translation\Translator	`translator`
Log	Illuminate\Log\Writer	`log`
Mail	Illuminate\Mail\Mailer	`mailer`
Password	Illuminate\Auth\Passwords\PasswordBroker	`auth.password`
Queue	Illuminate\Queue\QueueManager	`queue`
Queue ()	Illuminate\Queue\QueueInterface	
Queue ()	Illuminate\Queue\Queue	
Redirect	Illuminate\Routing\Redirector	`redirect`
Redis	Illuminate\Redis\Database	`redis`
Request	Illuminate\Http\Request	`request`
Response	Illuminate\Contracts\Routing\ResponseFactory	
Route	Illuminate\Routing\Router	`router`
Schema	Illuminate\Database\Schema\Blueprint	
Session	Illuminate\Session\SessionManager	`session`
Session ()	Illuminate\Session\Store	
Storage	Illuminate\Contracts\Filesystem\Factory	`filesystem`
URL	Illuminate\Routing\UrlGenerator	`url`
Validator	Illuminate\Validation\Factory	`validator`
Validator ()	Illuminate\Validation\Validator	
View	Illuminate\View\Factory	`view`
View ()	Illuminate\View\View	

-
-
-

Laravel

`public/index.php` Laravel Apache / Ngix `index.php`

`index.php` Composer `bootstrap/app.php` Laravel Laravel /

HTTP /

HTTP HTTP `app/Http/Kernel.php`

HTTP `Illuminate\Foundation\Http\Kernel` `bootstrappers` bootstrappers

HTTP HTTP HTTP session CSRF

HTTP `handle` `Request` `Response` HTTP HTTP

`config/app.php` `providers` `register` `boot`

`Request`

Laravel

Laravel `app/Providers`

`AppServiceProviders`

-
-
- App
-

Laravel Laravel - Composer

Laravel

`app`

`bootstrap`

`config`

`database`

`public` (JavaScriptCSS)

`resources` (LESSSASSCoffeeScript)

`storage` Blade session

`tests`

`vendor` Composer

App

`app` `App` Composer PSR-4 **`app:name`** **Artisan** .

`app` `Console` `Http` `Providers` `Console` `Http` API HTTP CLI
`Console` Artisan `Http`

`Commands`

`Events` Artisan

`Handlers`

`Services` Laravel `Registrar` API

`Exceptions`

> `app` Artisan `php artisan list make`

`App` `app:name` Artisan SocialNet

```
php artisan app:name SocialNet
```

-
-
-
-
- HTTP
-
-

Laravel `config/auth.php`

Laravel `app` Eloquent `App\User`

60 `users` () `remember_token` 100 `string` null
session token `$table->rememberToken();` Laravel 5 migrations

Eloquent Laravel `database`

Laravel `AuthController` `PasswordController`

trait `resources/views/auth`

`App\Services\Registrar`

`Registrar` `validator` `Registrar` `create` `User` `Registrar`
`AuthenticatesAndRegistersUsers` trait `AuthController`

`AuthController` Laravel `attempt`

```
<?php namespace App\Http\Controllers;

use Auth;
use Illuminate\Routing\Controller;

class AuthController extends Controller {

    /**
     * Handle an authentication attempt.
     *
     * @return Response
     */
```

```
    public function authenticate()
    {
        if (Auth::attempt(['email' => $email, 'password' => $password]))
        {
            return redirect()->intended('dashboard');
        }
    }

}
```

`attempt` `password` `email` `password` session

`attempt` `true` `false`

> `email` username

`intended` URL URI

```
if (Auth::attempt(['email' => $email, 'password' => $password, 'active' => 1]))
{
    // The user is active, not suspended, and exists.
}
```

`check`

```
if (Auth::check())
{
    // The user is logged in...
}
```

`attempt` `users` `remember_token`

```
if (Auth::attempt(['email' => $email, 'password' => $password], $remember))
{
    // The user is being remembered...
}
```

`viaRemember` cookie

```
if (Auth::viaRemember())
{
    //
}
```

ID

ID `loginUsingId`

```
Auth::loginUsingId(1);
```

`validate`

```
if (Auth::validate($credentials))
{
    //
}
```

`once` session cookie

```
if (Auth::once($credentials))
{
    //
}
```

`login`

```
Auth::login($user);
```

`attempt`

```
Auth::logout();
```

Laravel

`attempt` `auth.attempt` `auth.login`

`Auth` facade

```php
<?php namespace App\Http\Controllers;

use Illuminate\Routing\Controller;

class ProfileController extends Controller {

    /**
     * Update the user's profile.
     *
     * @return Response
     */
    public function updateProfile()
    {
        if (Auth::user())
        {
            // Auth::user() returns an instance of the authenticated user...
        }
    }

}
```

`Illuminate\Http\Request`

```php
<?php namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Routing\Controller;

class ProfileController extends Controller {

    /**
     * Update the user's profile.
     *
     * @return Response
     */
    public function updateProfile(Request $request)
    {
        if ($request->user())
        {
            // $request->user() returns an instance of the authenticated user...
        }
    }

}
```

`Illuminate\Contracts\Auth\Authenticatable` contract

```php
<?php namespace App\Http\Controllers;

use Illuminate\Routing\Controller;
use Illuminate\Contracts\Auth\Authenticatable;

class ProfileController extends Controller {

    /**
     * Update the user's profile.
     *
```

```
     * @return Response
     */
    public function updateProfile(Authenticatable $user)
    {
        // $user is an instance of the authenticated user...
    }

}
```

Laravel `auth` `app\Http\Middleware\Authenticate.php`

```
// With A Route Closure...

Route::get('profile', ['middleware' => 'auth', function()
{
    // Only authenticated users may enter...
}]);

// With A Controller...

Route::get('profile', ['middleware' => 'auth', 'uses' => 'ProfileController@show']);
```

HTTP

HTTP `auth.basic`

HTTP

```
Route::get('profile', ['middleware' => 'auth.basic', function()
{
    // Only authenticated users may enter...
}]);
```

`basic` `email` username

HTTP

HTTP session cookie API `onceBasic`

```
public function handle($request, Closure $next)
{
    return Auth::onceBasic() ?: $next($request);
}
```

PHP FastCGIHTTP `.htaccess`

```
RewriteCond %{HTTP:Authorization} ^(.+)$
RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
```

web Laravel

`User` `Illuminate\Contracts\Auth\CanResetPassword` Laravel `User` `Illuminate\Auth\Passwords\CanResetPassword`

Reminder

Laravel `database/migrations`

```
php artisan migrate
```

Laravel `Auth\PasswordController` `resources/views/auth`

e-mail `PasswordController` `getReset` `/home` `PasswordController` `redirectTo`

```
protected $redirectTo = '/dashboard';
```

> tokens `config/auth.php` `reminder.expire`

Laravel Laravel Socialite OAuth **Socialite Facebook TwitterGoogle GitHub Bitbucket**

`composer.json`

```
"laravel/socialite": "~2.0"
```

`config/app.php` `Laravel\Socialite\SocialiteServiceProvider` facade

```
'Socialize' => 'Laravel\Socialite\Facades\Socialite',
```

OAuth `config/services.php` `facebook` `twitter` `google` `github`

```
'github' => [
    'client_id' => 'your-github-app-id',
    'client_secret' => 'your-github-app-secret',
    'redirect' => 'http://your-callback-url',
],
```

`Socialize` facade

```
public function redirectToProvider()
{
    return Socialize::with('github')->redirect();
}

public function handleProviderCallback()
{
    $user = Socialize::with('github')->user();

    // $user->token;
}
```

`redirect` OAuth `user` scopes

```
return Socialize::with('github')->scopes(['scope1', 'scope2'])->redirect();
```

```
$user = Socialize::with('github')->user();

// OAuth Two Providers
$token = $user->token;

// OAuth One Providers
$token = $user->token;
$tokenSecret = $user->tokenSecret;

// All Providers
$user->getId();
$user->getNickname();
$user->getName();
$user->getEmail();
$user->getAvatar();
```

Laravel Cashier

Laravel Cashier Stripe Cashier PDF

Composer

Cashier `composer.json`

```
"laravel/cashier": "~3.0"
```

`app` `Laravel\Cashier\CashierServiceProvider`

Cashier `cashier:table` Artisan users `php artisan cashier:table users` `migrate`

`Billable` trait

```
use Laravel\Cashier\Billable;
use Laravel\Cashier\Contracts\Billable as BillableContract;

class User extends Eloquent implements BillableContract {

    use Billable;
```

```
    protected $dates = ['trial_ends_at', 'subscription_ends_at'];

}
```

Stripe Key

`AppServiceProvider` Stripe key

```
User::setStripeKey('stripe-key');
```

Stripe

```
$user = User::find(1);

$user->subscription('monthly')->create($creditCardToken);
```

`withCoupon`

```
$user->subscription('monthly')
     ->withCoupon('code')
     ->create($creditCardToken);
```

`subscription` Stripe Stripe customer ID Stripe

Stripe

```
$user->trial_ends_at = Carbon::now()->addDays(14);

$user->save();
```

`create`

```
$user->subscription('monthly')->create($creditCardToken, [
    'email' => $email, 'description' => 'Our First Customer'
]);
```

Stripe Stripe

`charge`

```
$user->charge(100);
```

`charge` 100 1.00

`charge` Stripe

```
$user->charge(100, [
    'source' => $token,
    'receipt_email' => $user->email,
]);
```

`charge` `false`

```
if ( ! $user->charge(100))
{
    // The charge was denied...
}
```

, Stripe

`cardUpFront` `false`

```
protected $cardUpFront = false;
```

```
$user->trial_ends_at = Carbon::now()->addDays(14);

$user->save();
```

`swap`

```
$user->subscription('premium')->swap();
```

$10 `increment` `decrement`

```
$user = User::find(1);

$user->subscription()->increment();

// Add five to the subscription's current quantity...
$user->subscription()->increment(5);

$user->subscription()->decrement();

// Subtract five to the subscription's current quantity...
$user->subscription()->decrement(5);
```

```
$user->subscription()->cancel();
```

Cashier `subscription_ends_at` `subscribed` `false` `subscribed` `true`

`resume`

```
$user->subscription('monthly')->resume($creditCardToken);
```

`subscribed`

```
if ($user->subscribed())
{
    //
}
```

`subscribed` :

```
public function handle($request, Closure $next)
{
    if ($request->user() && ! $request->user()->subscribed())
    {
        return redirect('billing');
    }

    return $next($request);
```

```
}
```

`onTrial`

```
if ($user->onTrial())
{
    //
}
```

`cancelled`

```
if ($user->cancelled())
{
    //
}
```

`subscribed` `true`

```
if ($user->onGracePeriod())
{
    //
}
```

`everSubscribed`

```
if ($user->everSubscribed())
{
    //
}
```

`onPlan` ID

```
if ($user->onPlan('monthly'))
{
    //
}
```

Cashier Webhook

```
Route::post('stripe/webhook', 'Laravel\Cashier\WebhookController@handleWebhook');
```

`stripe/webhook` URI Stripe URI

Stripe Webhooks

Stripe webhook Webhook Cashier `handle` Stripe webhook `invoice.payment_succeeded` webhook `handleInvoicePaymentSucceeded`

```
class WebhookController extends Laravel\Cashier\WebhookController {

    public function handleInvoicePaymentSucceeded($payload)
    {
        // Handle The Event
    }

}
```

> Webhook Stripe API

`invoices`

```
$invoices = $user->invoices();
```

```
{{ $invoice->id }}

{{ $invoice->dateString() }}

{{ $invoice->dollars() }}
```

`downloadInvoice` PDF

```
return $user->downloadInvoice($invoice->id, [
    'vendor'  => 'Your Company',
    'product' => 'Your Product',
]);
```

-
-
-
-
-

Laravel API `config/cache.php` Laravel Memcached Redis

Laravel `` Memcached APC

Laravel Redis Composer `predis/predis` (~1.0)

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

Carbon

```
$expiresAt = Carbon::now()->addMinutes(10);

Cache::put('key', 'value', $expiresAt);
```

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

`add` `true` `false`

```
if (Cache::has('key'))
{
    //
}
```

```
$value = Cache::get('key');
```

```
$value = Cache::get('key', 'default');

$value = Cache::get('key', function() { return 'default'; });
```

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

`Cache::remember`

```
$value = Cache::remember('users', $minutes, function()
{
    return DB::table('users')->get();
});
```

`remember` `forever`

```
$value = Cache::rememberForever('users', function()
{
    return DB::table('users')->get();
});
```

`pull`

```
$value = Cache::pull('key');
```

```
Cache::forget('key');
```

`store`

```
$value = Cache::store('foo')->get('key');
```

```
Cache::increment('key');

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

```
Cache::decrement('key');

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

> forever `memcached`

`tags`

```
Cache::tags('people', 'authors')->put('John', $john, $minutes);

Cache::tags(array('people', 'artists'))->put('Anne', $anne, $minutes);
```

`remember`, `forever`, `rememberForever` `increment` `decrement`

```
$anne = Cache::tags('people', 'artists')->get('Anne');

$john = Cache::tags(array('people', 'authors'))->get('John');
```

`people` `authors` AnneJohn:

```
Cache::tags('people', 'authors')->flush();
```

`authors` JohnAnne

```
Cache::tags('authors')->flush();
```

`Schema`

```
Schema::create('cache', function($table)
{
    $table->string('key')->unique();
    $table->text('value');
    $table->integer('expiration');
});
```

-
-

`Illuminate\Support\Collection` `collect`

```
$collection = collect(['taylor', 'abigail', null])->map(function($name)
{
    return strtoupper($name);
})
->reject(function($name)
{
    return empty($name);
});
```

`Collection` `Collection` `Collection`

`collect` `Illuminate\Support\Collection` `Collection` `make`

```
$collection = collect([1, 2, 3]);

$collection = Collection::make([1, 2, 3]);
```

Eloquent `Collection` `Collection`

() API

Command Bus

-
-
-
-
-

Command bus podcast

podcasts

HTTP HTTP podcast

`PurchasePodcast`

`make:command` Artisan

```
php artisan make:command PurchasePodcast
```

`app/Commands` `handle` `handle`

```
class PurchasePodcast extends Command implements SelfHandling {

    protected $user, $podcast;

    /**
     * Create a new command instance.
     *
     * @return void
     */
    public function __construct(User $user, Podcast $podcast)
    {
        $this->user = $user;
        $this->podcast = $podcast;
    }

    /**
     * Execute the command.
     *
     * @return void
     */
    public function handle()
    {
        // Handle the logic to purchase the podcast...

        event(new PodcastWasPurchased($this->user, $this->podcast));
    }
```

```
}
```

`handle`

```
    /**
     * Execute the command.
     *
     * @return void
     */
    public function handle(BillingGateway $billing)
    {
        // Handle the logic to purchase the podcast...
    }
```

`handle` Laravel "command bus"

`DispatchesCommands` trait `dispatch`

```
public function purchasePodcast($podcastId)
{
    $this->dispatch(
        new PurchasePodcast(Auth::user(), Podcast::findOrFail($podcastId))
    );
}
```

Command bus IoC `handle`

`Illuminate\Foundation\Bus\DispatchesCommands` trait command bus `Illuminate\Contracts\Bus\Dispatcher` `Bus` facade

```
    Bus::dispatch(
        new PurchasePodcast(Auth::user(), Podcast::findOrFail($podcastId))
    );
```

HTTP Laravel `DispatchesCommands` trait `dispatchFrom`

```
$this->dispatchFrom('Command\Class\Name', $request);
```

HTTP (`ArrayAccess`) `firstName` command bus HTTP `firstName`

`dispatchFrom` HTTP

```
$this->dispatchFrom('Command\Class\Name', $request, [
    'firstName' => 'Taylor',
]);
```

Command bus Laravel command bus `--queued`

```
php artisan make:command PurchasePodcast --queued
```

`Illuminate\Contracts\Queue\ShouldBeQueued` `SerializesModels` trait command bus Eloquent

`Illuminate\Contracts\Queue\ShouldBeQueued` ""

bus

.

"" HTTP

bus `App\Providers\BusServiceProvider::boot` `pipeThrough`

```
$dispatcher->pipeThrough(['UseDatabaseTransactions', 'LogCommand']);
```

`handle`

```
class UseDatabaseTransactions {

    public function handle($command, $next)
    {
        return DB::transaction(function() use ($command, $next)
        {
            return $next($command);
        });
    }

}
```

IoC

`

```
$dispatcher->pipeThrough([function($command, $next)
{
    return DB::transaction(function() use ($command, $next)
    {
```

```
        return $next($command);
    }
}]);
```

-
-
- Session
-
-

Laravel `Manager` session `CacheManager` APC Memcached

`extend`

> Laravel `Manager` `CacheManager` `SessionManager` Laravel `Illuminate\Support\Manager`

Laravel `CacheManager` `extend` mongo

```
Cache::extend('mongo', function($app)
{
    return Cache::repository(new MongoStore);
});
```

`extend` `config/cache.php` `driver` `Illuminate\Cache\Repository` `$app` `Illuminate\Foundation\Application`

`Cache::extend` Laravel `App\Providers\AppServiceProvider` `boot` - `config/app.php`

`Illuminate\Contracts\Cache\Store` contract MongoDB

```
class MongoStore implements Illuminate\Contracts\Cache\Store {

    public function get($key) {}
    public function put($key, $value, $minutes) {}
    public function increment($key, $value = 1) {}
    public function decrement($key, $value = 1) {}
    public function forever($key, $value) {}
    public function forget($key) {}
    public function flush() {}

}
```

MongoDB

```
Cache::extend('mongo', function($app)
{
    return Cache::repository(new MongoStore);
});
```

Packagist `app` `Extensions` Larave

Session

session Laravel `extend`

```
Session::extend('mongo', function($app)
{
    // Return implementation of SessionHandlerInterface
});
```

Session

session `AppServiceProvider` `boot`

Session

`SessionHandlerInterface` MongoDB

```
class MongoHandler implements SessionHandlerInterface {

    public function open($savePath, $sessionName) {}
    public function close() {}
    public function read($sessionId) {}
    public function write($sessionId, $data) {}
    public function destroy($sessionId) {}
    public function gc($lifetime) {}

}
```

`StoreInterface`

- `open` session Laravel `file` session PHP ()
- `close` `open`
- `read` `$sessionId` session session Laravel
- `write` `$data` `$sessionId` MongoDB Dynamo
- `destroy` `$sessionId`
- `gc` `$lifetime` UNIX session Memcached Redis

`SessionHandlerInterface` Session

```
Session::extend('mongo', function($app)
{
    return new MongoHandler;
```

```
});
```

session `config/session.php` `mongo`

> session Packagist

---

session `extend`

```
Auth::extend('riak', function($app)
{
    //  Illuminate\Contracts\Auth\UserProvider
});
```

`UserProvider` `Illuminate\Contracts\Auth\Authenticatable` MySQL Riak Laravel

`UserProvider` contract

```
interface UserProvider {

    public function retrieveById($identifier);
    public function retrieveByToken($identifier, $token);
    public function updateRememberToken(Authenticatable $user, $token);
    public function retrieveByCredentials(array $credentials);
    public function validateCredentials(Authenticatable $user, array $credentials);

}
```

`retrieveById` MySQL ID ID `Authenticatable`

`retrieveByToken` `$identifier` `remember_token` `$token` `Authenticatable`

`updateRememberToken` `$token` `$user` `remember_token` token token null

`retrieveByCredentials` `Auth::attempt` `$credentials['username']` where `UserInterface`

`validateCredentials` `$user` `$credentials` `$user->getAuthPassword()` `Hash::make` `$credentials['password']`

`UserProvider` `Authenticatable` `retrieveById` `retrieveByCredentials`

```
interface Authenticatable {

    public function getAuthIdentifier();
    public function getAuthPassword();
    public function getRememberToken();
```

```
    public function setRememberToken($value);
    public function getRememberTokenName();

}
```

The `getAuthIdentifier` MySQL `getAuthPassword` ORM Laravel `User` `app`

`UserProvider` `Auth` facade

```
Auth::extend('riak', function($app)
{
    return new RiakUserProvider($app['riak.connection']);
});
```

`extend` `config/auth.php`

Laravel `config/app.php`

`HashServiceProvider` `hash` `Illuminate\Hashing\BcryptHasher` IoC

```
<?php namespace App\Providers;

class SnappyHashProvider extends \Illuminate\Hashing\HashServiceProvider {

    public function boot()
    {
        parent::boot();

        $this->app->bindShared('hash', function()
        {
            return new \Snappy\Hashing\ScryptHasher;
        });
    }

}
```

`HashServiceProvider` `ServiceProvider` `config/app.php` `HashServiceProvider`

Laravel

Laravel Elixir

-
-
-
- Gulp
-
-

Laravel Elixir API Laravel Gulp Elixir CSS JavaScrtip

Gulp Laravel Elixir

Node

Elixir Node.js

```
node -v
```

Laravel Homestead Vagrant Node

Gulp

Gulp NPM

```
npm install --global gulp
```

Laravel Elixir

Elixir Laravel `package.json` `composer.json` Node PHP

```
npm install
```

Elixir `gulpfile.js` Elixir

Less

```
elixir(function(mix) {
```

```
    mix.less("app.less");
});
```

Elixir Less `resources/assets/less`

Less

```
elixir(function(mix) {
    mix.less([
        'app.less',
        'something-else.less'
    ]);
});
```

Sass

```
elixir(function(mix) {
    mix.sass("app.scss");
});
```

Elixir Sass `resources/assets/sass`

CoffeeScript

```
elixir(function(mix) {
    mix.coffee();
});
```

Elixir CoffeeScript `resources/assets/coffee`

Less CoffeeScript

```
elixir(function(mix) {
    mix.less()
       .coffee();
});
```

PHPUnit

```
elixir(function(mix) {
    mix.phpUnit();
});
```

PHPSpec

```
elixir(function(mix) {
    mix.phpSpec();
});
```

```
elixir(function(mix) {
    mix.styles([
        "normalize.css",
        "main.css"
    ]);
});
```

`resources/css`

```
elixir(function(mix) {
    mix.styles([
        "normalize.css",
        "main.css"
    ], 'public/build/css/everything.css');
});
```

```
elixir(function(mix) {
    mix.styles([
        "normalize.css",
        "main.css"
    ], 'public/build/css/everything.css', 'public/css');
});
```

`styles` `scrtips`

```
elixir(function(mix) {
    mix.stylesIn("public/css");
});
```

```
elixir(function(mix) {
    mix.scripts([
        "jquery.js",
        "app.js"
    ]);
});
```

`resources/js`

```
elixir(function(mix) {
    mix.scriptsIn("public/js/some/directory");
});
```

```
elixir(function(mix) {
    mix.scripts(['jquery.js', 'main.js'], 'public/js/main.js')
       .scripts(['forum.js', 'threads.js'], 'public/js/forum.js');
});
```

```
elixir(function(mix) {
    mix.version("css/all.css");
});
```

`all-16d570a7.css`

`elixir()`

```
<link rel="stylesheet" href="{{ elixir("css/all.css") }}">
```

`elixir()`

`version`

```
elixir(function(mix) {
    mix.version(["css/all.css", "js/app.js"]);
});
```

```
<link rel="stylesheet" href="{{ elixir("css/all.css") }}">
<script src="{{ elixir("js/app.js") }}"></script>
```

```
elixir(function(mix) {
    mix.copy('vendor/foo/bar.css', 'public/css/bar.css');
});
```

```
elixir(function(mix) {
    mix.copy('vendor/package/views', 'resources/views');
});
```

Elixir

```
elixir(function(mix) {
    mix.less("app.less")
       .coffee()
       .phpUnit()
       .version("css/bootstrap.css");
});
```

Gulp

Elixir Gulp

```
gulp
```

```
gulp watch
```

javascript

```
gulp scripts
```

css

```
gulp styles
```

PHP

```
gulp tdd
```

```
gulp --production
```

Gulp Elixir

```
var gulp = require("gulp");
var shell = require("gulp-shell");
var elixir = require("laravel-elixir");
```

```
elixir.extend("message", function(message) {

    gulp.task("say", function() {
        gulp.src("").pipe(shell("say " + message));
    });

    return this.queueTask("say");

});
```

`extend` Elixir API Gulpfile Gulp

```
this.registerWatcher("message", "**/*.php");
```

`message`

Gulpfile Gulpfile

```
require("./custom-tasks")
```

```
elixir(function(mix) {
    mix.message("Tea, Earl Grey, Hot");
});
```

GulpPicard

-
-

Laravel Mcrypt PHP AES

```
$encrypted = Crypt::encrypt('secret');
```

> `config/app.php` `key` 16, 24, 32

```
$decrypted = Crypt::decrypt($encryptedValue);
```

```
Crypt::setMode('ctr');

Crypt::setCipher($cipher);
```

Envoy Task Runner

Introduction

Laravel Envoy provides a clean, minimal syntax for defining common tasks you run on your remote servers. Using a Blade style syntax, you can easily setup tasks for deployment, Artisan commands, and more.

> **Note:** Envoy requires PHP version 5.4 or greater, and only runs on Mac / Linux operating systems.

Installation

First, install Envoy using the Composer `global` command:

```
composer global require "laravel/envoy=~1.0"
```

Make sure to place the `~/.composer/vendor/bin` directory in your PATH so the `envoy` executable is found when you run the `envoy` command in your terminal.

Next, create an `Envoy.blade.php` file in the root of your project. Here's an example to get you started:

```
@servers(['web' => '192.168.1.1'])

@task('foo', ['on' => 'web'])
    ls -la
@endtask
```

As you can see, an array of `@servers` is defined at the top of the file. You can reference these servers in the `on` option of your task declarations. Within your `@task` declarations you should place the Bash code that will be run on your server when the task is executed.

The `init` command may be used to easily create a stub Envoy file:

```
envoy init user@192.168.1.1
```

Running Tasks

To run a task, use the `run` command of your Envoy installation:

```
envoy run foo
```

If needed, you may pass variables into the Envoy file using command line switches:

```
envoy run deploy --branch=master
```

You may use the options via the Blade syntax you are used to:

```
@servers(['web' => '192.168.1.1'])

@task('deploy', ['on' => 'web'])
    cd site
    git pull origin {{ $branch }}
    php artisan migrate
@endtask
```

Bootstrapping

You may use the `@setup` directive to declare variables and do general PHP work inside the Envoy file:

```
@setup
    $now = new DateTime();

    $environment = isset($env) ? $env : "testing";
@endsetup
```

You may also use `@include` to include any PHP files:

```
@include('vendor/autoload.php');
```

Multiple Servers

You may easily run a task across multiple servers. Simply list the servers in the task declaration:

```
@servers(['web-1' => '192.168.1.1', 'web-2' => '192.168.1.2'])

@task('deploy', ['on' => ['web-1', 'web-2']])
    cd site
    git pull origin {{ $branch }}
    php artisan migrate
@endtask
```

By default, the task will be executed on each server serially. Meaning, the task will finish running on the first server before proceeding to execute on the next server.

Parallel Execution

If you would like to run a task across multiple servers in parallel, simply add the `parallel` option to your task declaration:

```
@servers(['web-1' => '192.168.1.1', 'web-2' => '192.168.1.2'])

@task('deploy', ['on' => ['web-1', 'web-2'], 'parallel' => true])
    cd site
    git pull origin {{ $branch }}
    php artisan migrate
@endtask
```

Task Macros

Macros allow you to define a set of tasks to be run in sequence using a single command. For instance:

```
@servers(['web' => '192.168.1.1'])

@macro('deploy')
    foo
    bar
@endmacro

@task('foo')
    echo "HELLO"
@endtask

@task('bar')
    echo "WORLD"
@endtask
```

The `deploy` macro can now be run via a single, simple command:

```
envoy run deploy
```

Notifications

HipChat

After running a task, you may send a notification to your team's HipChat room using the simple `@hipchat` directive:

```
@servers(['web' => '192.168.1.1'])

@task('foo', ['on' => 'web'])
    ls -la
@endtask

@after
    @hipchat('token', 'room', 'Envoy')
```

```
@endafter
```

You can also specify a custom message to the hipchat room. Any variables declared in `@setup` or included with `@include` will be available for use in the message:

```
@after
    @hipchat('token', 'room', 'Envoy', "$task ran on [$environment]")
@endafter
```

This is an amazingly simple way to keep your team notified of the tasks being run on the server.

Slack

The following syntax may be used to send a notification to Slack:

```
@after
    @slack('team', 'token', 'channel')
@endafter
```

Updating Envoy

To update Envoy, simply use Composer:

```
composer global update
```

-
-
- HTTP
-

`Illuminate\Foundation\Bootstrap\ConfigureLogging` `config/app.php` `log`

Laravel Monolog Monolog

`config/app.php`

```
'log' => 'single'
```

Laravel `single` `daily` `syslog` `ConfigureLogging`

`config/app.php` `app.debug` `.env` `APP_DEBUG`

`APP_DEBUG` `true` **`false`**

`App\Exceptions\Handler` `report` `render`

`report` BugSnag `report` PHP

```
/**
 *
 *
 *  SentryBugsnag
 *
 * @param  \Exception  $e
 * @return void
 */
public function report(Exception $e)
{
    if ($e instanceof CustomException)
    {
        //
    }

    return parent::report($e);
}
```

`render` HTTP

`dontReport` 404

HTTP

HTTP (404)(401) 500

```
abort(404);
```

```
abort(403, 'Unauthorized action.');
```

404

404 `resources/views/errors/404.blade.php` 404

Laravel Monolog Laravel `storage/logs`

```
Log::info('This is some useful information.');

Log::warning('Something could be going wrong.');

Log::error('Something is really going wrong.');
```

RFC 5424 **debuginfonoticewarningerrorcritical alert**

```
Log::info('Log message', ['context' => 'Other helpful information']);
```

Monolog Laravel Monolog

```
$monolog = Log::getMonolog();
```

```
Log::listen(function($level, $message, $context)
{
```

```
    //
});
```

-
-
-

Laravel event `app/Events` `app/Handlers/Events`

Artisan

```
php artisan make:event PodcastWasPurchased
```

Laravel `EventServiceProvider` `listen` () ()
`PodcastWasPurchased`

```
/**
 *
 *
 * @var array
 */
protected $listen = [
    'App\Events\PodcastWasPurchased' => [
        'App\Handlers\Events\EmailPurchaseConfirmation',
    ],
];
```

Artisan `handler:event`

```
php artisan handler:event EmailPurchaseConfirmation --event=PodcastWasPurchased
```

`make:event` `handler:event` `EventServiceProvider` `event:generate`
`EventServiceProvider`

```
php artisan event:generate
```

`Event` facade

```
$response = Event::fire(new PodcastWasPurchased($podcast));
```

`fire`

`event`

```
event(new PodcastWasPurchased($podcast));
```

`EventServiceProvider` `boot`

```
Event::listen('App\Events\PodcastWasPurchased', function($event)
{
    // ...
});
```

`false`

```
Event::listen('App\Events\PodcastWasPurchased', function($event)
{
    // ...

    return false;
});
```

`--queued`

```
php artisan handler:event SendPurchaseConfirmation --event=PodcastWasPurchased --queued
```

`Illuminate\Contracts\Queue\ShouldBeQueued`

`delete` `release` `Illuminate\Queue\InteractsWithQueue` trait

```
public function handle(PodcastWasPurchased $event)
{
    if (true)
    {
        $this->release(30);
    }
}
```

`ShouldBeQueued`

`subscribe`

```
class UserEventHandler {

    /**
     *
     */
    public function onUserLogin($event)
    {
        //
    }

    /**
     *
     */
    public function onUserLogout($event)
    {
        //
    }

    /**
     *
     *
     * @param  Illuminate\Events\Dispatcher  $events
     * @return array
     */
    public function subscribe($events)
    {
        $events->listen('App\Events\UserLoggedIn', 'UserEventHandler@onUserLogin');

        $events->listen('App\Events\UserLoggedOut', 'UserEventHandler@onUserLogout');
    }

}
```

`Event`

```
$subscriber = new UserEventHandler;

Event::subscribe($subscriber);
```

`subscribe`

```
Event::subscribe('UserEventHandler');
```

/

-
-
-
-

Laravel Frank de Jonge Flysystem Laravel Flysystem Amazon S3 Rackspace Cloud Storage API

`config/filesystems.php`

S3 Rackspace Composer

- Amazon S3: `league/flysystem-aws-s3-v2 ~1.0`
- Rackspace: `league/flysystem-rackspace ~1.0`

`local` `root` `storage/app` `storage/app/file.txt`

```
Storage::disk('local')->put('file.txt', 'Contents');
```

`Storage` facade `Illuminate\Contracts\Filesystem\Factory` Laravel

```
$disk = Storage::disk('s3');

$disk = Storage::disk('local');
```

```
$exists = Storage::disk('s3')->exists('file.jpg');
```

```
if (Storage::exists('file.jpg'))
{
```

```
    //
}
```

```
$contents = Storage::get('file.jpg');
```

```
Storage::put('file.jpg', $contents);
```

```
Storage::prepend('file.log', 'Prepended Text');
```

```
Storage::append('file.log', 'Appended Text');
```

```
Storage::delete('file.jpg');

Storage::delete(['file1.jpg', 'file2.jpg']);
```

```
Storage::copy('old/file1.jpg', 'new/file1.jpg');
```

```
Storage::move('old/file1.jpg', 'new/file1.jpg');
```

```
$size = Storage::size('file1.jpg');
```

(UNIX)

```
$time = Storage::lastModified('file1.jpg');
```

```
$files = Storage::files($directory);

// Recursive...
$files = Storage::allFiles($directory);
```

```
$directories = Storage::directories($directory);

// Recursive...
$directories = Storage::allDirectories($directory);
```

```
Storage::makeDirectory($directory);
```

```
Storage::deleteDirectory($directory);
```

Laravel

`DropboxFilesystemServiceProvider` `boot` `Illuminate\Contracts\Filesystem\Factory` `extend` `Disk` facade `extend` `extend` `$app` `$config` `League\Flysystem\Filesystem`

> $config `config/filesystems.php`

Dropbox

```
<?php namespace App\Providers;

use Storage;
use League\Flysystem\Filesystem;
use Dropbox\Client as DropboxClient;
use League\Flysystem\Dropbox\DropboxAdapter;

class DropboxFilesystemServiceProvider {

    public function boot()
    {
        Storage::extend('dropbox', function($app, $config)
        {
            $client = new DropboxClient($config['accessToken'], $config['clientIdentifier

            return new Filesystem(new DropboxAdapter($client));
        });
```

```
    }

}
```

-
-

Laravel `Hash` Bcrypt Laravel Bcrypt Bcrypt Laravel `bcrypt`

A Bcrypt

```
$password = Hash::make('secret');
```

bcrypt function

```
$password = bcrypt('secret');
```

A

```
if (Hash::check('secret', $hashedPassword))
{
    // The passwords match...
}
```

A

```
if (Hash::needsRehash($hashed))
{
    $hashed = Hash::make('secret');
}
```

-
-
-
-
-
-

array_add

`array_add`

```
$array = array('foo' => 'bar');

$array = array_add($array, 'key', 'value');
```

array_divide

`array_divide`

```
$array = array('foo' => 'bar');

list($keys, $values) = array_divide($array);
```

array_dot

`array_dot`

```
$array = array('foo' => array('bar' => 'baz'));

$array = array_dot($array);

// array('foo.bar' => 'baz');
```

array_except

`array_except`

```
$array = array_except($array, array('keys', 'to', 'remove'));
```

array_fetch

`array_fetch`

```
$array = array(
    array('developer' => array('name' => 'Taylor')),
    array('developer' => array('name' => 'Dayle')),
);

$array = array_fetch($array, 'developer.name');

// array('Taylor', 'Dayle');
```

array_first

`array_first`

```
$array = array(100, 200, 300);

$value = array_first($array, function($key, $value)
{
    return $value >= 150;
});
```

```
$value = array_first($array, $callback, $default);
```

array_last

`array_last`

```
$array = array(350, 400, 500, 300, 200, 100);

$value = array_last($array, function($key, $value)
{
    return $value > 350;
});

// 500
```

```
$value = array_last($array, $callback, $default);
```

array_flatten

`array_flatten`

```
$array = array('name' => 'Joe', 'languages' => array('PHP', 'Ruby'));
```

```
$array = array_flatten($array);

// array('Joe', 'PHP', 'Ruby');
```

array_forget

`array_forget`

```
$array = array('names' => array('joe' => array('programmer')));

array_forget($array, 'names.joe');
```

array_get

`array_get`

```
$array = array('names' => array('joe' => array('programmer')));

$value = array_get($array, 'names.joe');

$value = array_get($array, 'names.john', 'default');
```

> : `array_get` `object_get`

array_only

`array_only`

```
$array = array('name' => 'Joe', 'age' => 27, 'votes' => 1);

$array = array_only($array, array('name', 'votes'));
```

array_pluck

`array_pluck`

```
$array = array(array('name' => 'Taylor'), array('name' => 'Dayle'));

$array = array_pluck($array, 'name');

// array('Taylor', 'Dayle');
```

array_pull

`array_pull`

```
$array = array('name' => 'Taylor', 'age' => 27);
```

```
$name = array_pull($array, 'name');
```

array_set

`array_set`

```
$array = array('names' => array('programmer' => 'Joe'));

array_set($array, 'names.editor', 'Taylor');
```

array_sort

`array_sort`

```
$array = array(
    array('name' => 'Jill'),
    array('name' => 'Barry'),
);

$array = array_values(array_sort($array, function($value)
{
    return $value['name'];
}));
```

array_where

```
$array = array(100, '200', 300, '400', 500);

$array = array_where($array, function($key, $value)
{
    return is_string($value);
});

// Array ( [1] => 200 [3] => 400 )
```

head

PHP 5.3.x

```
$first = head($this->returnsArray('foo'));
```

last

```
$last = last($this->returnsArray('foo'));
```

---

app_path

`app`

```
$path = app_path();
```

base_path

public_path

`public`

storage_path

`app/storage`

---

get

GET

```
get('/', function() { return 'Hello World'; });
```

post

POST

```
post('foo/bar', 'FooController@action');
```

put

PUT

```
put('foo/bar', 'FooController@action');
```

patch

PATCH

```
patch('foo/bar', 'FooController@action');
```

delete

DELETE

```
delete('foo/bar', 'FooController@action');
```

resource

RESTful

```
resource('foo', 'FooController');
```

---

camel_case

```
$camel = camel_case('foo_bar');

// fooBar
```

class_basename

```
$class = class_basename('Foo\Bar\Baz');

// Baz
```

e

`htmlentities` UTF-8

```
$entities = e('<html>foo</html>');
```

ends_with

```
$value = ends_with('This is my name', 'name');
```

snake_case

```
$snake = snake_case('fooBar');

// foo_bar
```

str_limit

```
str_limit($value, $limit = 100, $end = '...')
```

```
$value = str_limit('The PHP framework for web artisans.', 7);

// The PHP...
```

starts_with

```
$value = starts_with('This is my name', 'This');
```

str_contains

```
$value = str_contains('This is my name', 'my');
```

str_finish

```
$string = str_finish('this/string', '/');

// this/string/
```

str_is

```
$value = str_is('foo*', 'foobar');
```

str_plural

()

```
$plural = str_plural('car');
```

str_random

```
$string = str_random(40);
```

str_singular

()

```
$singular = str_singular('cars');
```

str_slug

slug

```
str_slug($title, $separator);
```

```
$title = str_slug("Laravel 5 Framework", "-");

// laravel-5-framework
```

studly_case

```
$value = studly_case('foo_bar');

// FooBar
```

trans

`Lang::get`

```
$value = trans('validation.required'):
```

trans_choice

`Lang::choice`

```
$value = trans_choice('foo.bar', $count);
```

---

action

```
$url = action('HomeController@getIndex', $params);
```

route

```
$url = route('routeName', $params);
```

asset

```
$url = asset('img/photo.jpg');
```

secure_asset

HTTPS HTML

```
echo secure_asset('foo/bar.zip', $title, $attributes = array());
```

secure_url

HTTPS

```
echo secure_url('foo/bar', $parameters = array());
```

url

```
echo url('foo/bar', $parameters = array(), $secure = null);
```

---

csrf_token

CSRF token

```
$token = csrf_token();
```

dd

```
dd($value);
```

elixir

Elixir

```
elixir($file);
```

env

```
env('APP_ENV', 'production')
```

event

```
event('my.event');
```

value

```
$value = value(function() { return 'bar'; });
```

view

Get a View instance for the given view path.

```
return view('auth.login');
```

with

PHP 5.3.x

```
$value = with(new Foo)->doWork();
```

-
-
-
-
-
-

Laravel `Lang` facade

`resources/lang`

```
/resources
    /lang
        /en
            messages.php
        /es
            messages.php
```

```
<?php

return array(
    'welcome' => 'Welcome to our application'
);
```

`config/app.php` `App::setLocale`

```
App::setLocale('es');
```

`config/app.php`

```
'fallback_locale' => 'en',
```

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

`get`

> `get`

`trans` `Lang::get`

```
echo trans('messages.welcome');
```

```
'welcome' => 'Welcome, :name',
```

`Lang::get`

```
echo Lang::get('messages.welcome', array('name' => 'Dayle'));
```

```
if (Lang::has('messages.welcome'))
{
    //
}
```

```
'apples' => 'There is one apple|There are many apples',
```

`Lang::choice`

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

(ru)

```
echo Lang::choice('товар|товара|товаров', $count, array(), 'ru');
```

Laravel Symfony

```
'apples' => '{0} There are none|[1,19] There are some|[20,Inf] There are many',
```

.

`resources/lang/packages/{locale}/{package}` `skyrim/hearthfire` `messages.php` `resources/lang/packages/en/hearthfire/messages.php`

-
-
-
-
-

Laravel SwiftMailer API `config/mail.php` SMTP
PHP `mail` `driver` `mail` `sendmail`

API

Laravel Mailgun Mandrill HTTP API API SMTP Guzzle 4 HTTP
`composer.josn` Guzzle 4

```
"guzzlehttp/guzzle": "~4.0"
```

Mailgun

Mailgun `config/mail.php` `driver` `mailgun` `config/service.php`

```
'mailgun' => array(
    'domain' => 'your-mailgun-domain',
    'secret' => 'your-mailgun-key',
),
```

Mandrill

Mandrill `config/mail.php` `driver` `mandrill` `config/service.php`

```
'mandrill' => array(
    'secret' => 'your-mandrill-key',
),
```

`config/mail.php` `driver` `log`

`Mail::send`

```
Mail::send('emails.welcome', array('key' => 'value'), function($message)
{
    $message->to('foo@example.com', 'John Smith')->subject('Welcome!');
});
```

`send` `$key` message

> `$message` `message`

HTML

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

`html` `text`

```
Mail::send(array('text' => 'view'), $data, $callback);
```

```
Mail::send('emails.welcome', $data, function($message)
{
    $message->from('us@example.com', 'Laravel');

    $message->to('foo@example.com')->cc('bar@example.com');

    $message->attach($pathToFile);
});
```

message MIME

```
$message->attach($pathToFile, array('as' => $display, 'mime' => $mime));
```

`raw`

```
Mail::raw('Text to e-mail', function($message)
{
    $message->from('us@example.com', 'Laravel');

    $message->to('foo@example.com')->cc('bar@example.com');
});
```

> `Mail::send` message SwiftMailer message

Laravel CID

```
<body>

    <img src="<?php echo $message->embed($pathToFile); ?>">
</body>
```

```
<body>

    <img src="<?php echo $message->embedData($data, $name); ?>">
</body>
```

`Mail` `$message`

Laravel queue A

```
Mail::queue('emails.welcome', $data, function($message)
{
    $message->to('foo@example.com', 'John Smith')->subject('Welcome!');
});
```

`later`

```
Mail::later(5, 'emails.welcome', $data, function($message)
{
    $message->to('foo@example.com', 'John Smith')->subject('Welcome!');
});
```

`queueOn` `laterOn`

```
Mail::queueOn('queue-name', 'emails.welcome', $data, function($message)
{
    $message->to('foo@example.com', 'John Smith')->subject('Welcome!');
});
```

`Mail::pretend` `config/mail.php` `pretend` `true` `pretend`

MailTrap

-
-
-
-
-
-
-

Laravel Carbon BDD testing Behat

LaravelCarbon Behat `composer.json`

Laravel Laravel Laravel

Laravel Packagist Composer PHP

.

:

```
return view('package::view.name');
```

Laravel “courier” `boot` :

```
public function boot()
{
    $this->loadViewsFrom(__DIR__.'/path/to/views', 'courier');
}
```

:

```
return view('courier::view.name');
```

`loadViewsFrom` Laravel `resources/views/vendor` `courier` Laravel `resources/views/vendor/courier` Laravel `loadViewsFrom`

`resources/views/vendor` `boot` `publishes` :

```
public function boot()
{
    $this->loadViewsFrom(__DIR__.'/path/to/views', 'courier');

    $this->publishes([
        __DIR__.'/path/to/views' => base_path('resources/views/vendor/courier'),
    ]);
}
```

Laravel `vendor:publish`

`--force` :

```
php artisan vendor:publish --force
```

> : `publishes`

:

```
return trans('package::file.line');
```

Laravel "courier" `boot` :

```
public function boot()
{
    $this->loadTranslationsFrom(__DIR__.'/path/to/translations', 'courier');
}
```

`translations` `en` `es` `ru`

:

```
return trans('courier::file.line');
```

`config`

`boot` `publishes` :

```
$this->publishes([
    __DIR__.'/path/to/config/courier.php' => config_path('courier.php'),
]);
```

`vendor:publish` :

```
$value = config('courier.option');
```

`register` `mergeConfigFrom`

```
$this->mergeConfigFrom(
    __DIR__.'/path/to/config/courier.php', 'courier'
);
```

JavaScriptCSS `boot` `publishes` “public”

```
$this->publishes([
    __DIR__.'/path/to/assets' => public_path('vendor/courier'),
], 'public');
```

`vendor:publish` `--force`

```
php artisan vendor:publish --tag=public --force
```

`composer.json` `post-update-cmd`

`tagging` :

```
// Publish a config file
$this->publishes([
    __DIR__.'/../config/package.php' => config_path('package.php')
], 'config');

// Publish your migrations
$this->publishes([
    __DIR__.'/../database/migrations/' => base_path('/database/migrations')
], 'migrations');
```

`tag`

```
php artisan vendor:publish --provider="Vendor\Providers\PackageServiceProvider" --tag="co
```

`boot` `include` :

```
public function boot()
{
    include __DIR__.'/../../routes.php';
}
```

: `composer.json` auto-load

-
-
-
- JSON

Laravel Laravel HTML Bootstrap CSS .

`paginate` Eloquent

```
$users = DB::table('users')->paginate(15);
```

> Laravel `groupBy` `groupBy` `Paginator::make`

Eloquent

Eloquent :

```
$allUsers = User::paginate(15);

$someUsers = User::where('votes', '>', 100)->paginate(15);
```

`paginate` `render`

```
<div class="container">
    <?php foreach ($users as $user): ?>
        <?php echo $user->name; ?>
    <?php endforeach; ?>
</div>

<?php echo $users->render(); ?>
```

Laravel Laravel

- `currentPage`
- `lastPage`
- `perPage`
- `hasMorePages`

- `url`
- `nextPageUrl`
- `total`
- `count`

`simplePaginate`

```
$someUsers = User::where('votes', '>', 100)->simplePaginate(15);
```

`Illuminate\Pagination\Paginator` `Illuminate\Pagination\LengthAwarePaginator`

URL

`setPath` URL

```
$users = User::paginate();

$users->setPath('custom/url');
```

URL http://example.com/custom/url?page=2

`appends`

```
<?php echo $users->appends(['sort' => 'votes'])->render(); ?>
```

```
http://example.com/something?page=2&sort=votes
```

URL `fragment`

```
<?php echo $users->fragment('foo')->render(); ?>
```

URL

```
http://example.com/something?page=2#foo
```

JSON

`Paginator` `Illuminate\Contracts\Support\JsonableInterface` `toJson`
`Paginator` JSONJSON `total` `current_page` `last_page` JSON `data`

-
-
-
-
-
-
-

Laravel API

`config/queue.php` BeanstalkdIronMQAmazon SQSRedis `null` () `null`

`database` `queue:table` Artisan

```
php artisan queue:table
```

- Amazon SQS: `aws/aws-sdk-php`
- Beanstalkd: `pda/pheanstalk ~3.0`
- IronMQ: `iron-io/iron_mq`
- Redis: `predis/predis ~1.0`

`App\Commands` Artisan

```
php artisan make:command SendEmail --queued
```

`Queue::push`

```
Queue::push(new SendEmail($message));
```

> : `Queue` Facade Command Bus `Queue` Facade command bus

`make:command` Artisan "self-handling" `handle` `handle`

```
public function handle(UserRepository $users)
{
    //
}
```

`make:command` `--handler`

```
php artisan make:command SendEmail --queued --handler
```

`App\Handlers\Commands`

```
Queue::pushOn('emails', new SendEmail($message));
```

`Queue::bulk`

```
Queue::bulk(array(new SendEmail($message), new AnotherCommand));
```

15 e-mail `Queue::later`

```
$date = Carbon::now()->addMinutes(15);

Queue::later($date, new SendEmail($message));
```

Carbon

> : Amazon SQS 900 15

Eloquent

Eloquent identifier Eloquent

`Illuminate\Queue\InteractsWithQueue` trait `release` `delete` `release`

```
public function handle(SendEmail $command)
{
    if (true)
    {
        $this->release(30);
    }
}
```

`queue:listen` `queue:work` Artisan `--tries`

`attempts`

```
if ($this->attempts() > 3)
{
    //
}
```

: `Illuminate\Queue\InteractsWithQueue` trait

```
Queue::push(function($job) use ($id)
{
    Account::delete($id);

    $job->delete();
});
```

: `use`

Iron.io push queues ,token Iron.io `https://yourapp.com/queue/receive?token=SecretToken` token

Laravel Artisan `queue:listen`

```
php artisan queue:listen
```

```
php artisan queue:listen connection
```

Supervisor

`listen`

```
php artisan queue:listen --queue=high,low
```

`high-connection` `low-connection`

```
php artisan queue:listen --timeout=60
```

```
php artisan queue:listen --sleep=5
```

`queue:work` Artisan

```
php artisan queue:work
```

`queue:work` `--daemon` `queue:listen` CPU

`--daemon`

```
php artisan queue:work connection --daemon

php artisan queue:work connection --daemon --sleep=3

php artisan queue:work connection --daemon --sleep=3 --tries=3
```

`queue:work` `queue:listen` `php artisan help queue:work`

`php artisan down` Laravel

`queue`

```
php artisan queue:restart
```

> : APCu APCu `apc.enable_cli=1` APCu

GD `imagedestroy`

`DB::reconnect`

Laravel 5 Iron.io Iron.io Iron `config/queue.php`

`queue:subscribe` Artisan URL

```
php artisan queue:subscribe queue_name http://foo.com/queue/receive
```

Iron URL URLs route `queue/receive` `Queue::marshal`

```
Route::post('queue/receive', function()
{
    return Queue::marshal();
});
```

`marshal` `Queue::push`

Laravel `failed_jobs` `config/queue.php`

`failed_jobs` `queue:failed-table` Artisan

```
php artisan queue:failed-table
```

`queue:listen` `--tries`

```
php artisan queue:listen connection-name --tries=3
```

`Queue::failing` e-mail HipChat

```
Queue::failing(function($connection, $job, $data)
{
    //
});
```

`failed`

```
public function failed()
{
    // ……
}
```

`queue:failed`

```
php artisan queue:failed
```

`queue:failed` ID ID ID 5

```
php artisan queue:retry 5
```

`queue:forget`

```
php artisan queue:forget 5
```

`queue:flush`

```
php artisan queue:flush
```

HTTP Stateless session Laravel session API MemcachedRedis

session `config/session.php` session Laravel `file` session

Laravel `Redis` sessions Composer `predis/predis` (~1.0)

> session `encrypt` `true`

Laravel `flash` session session

Session

session HTTP request `session` `Session` facade `session` `session` session

```
session()->regenerate();
```

Session

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

session(['key' => 'value']);
```

Session

```
Session::push('user.teams', 'developers');
```

Session

```
$value = Session::get('key');

$value = session('key');
```

Session

```
$value = Session::get('key', 'default');

$value = Session::get('key', function() { return 'default'; });
```

Session

```
$value = Session::pull('key', 'default');
```

Session

```
$data = Session::all();
```

Session

```
if (Session::has('users'))
{
    //
}
```

Session

```
Session::forget('key');
```

Session

```
Session::flush();
```

Session ID

```
Session::regenerate();
```

Flash Data

`Session::flash`

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

```
Session::reflash();
```

```
Session::keep(array('username', 'email'));
```

Sessions

`database` session session `Schema`

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

Artisan `session:table` migration

```
php artisan session:table

composer dump-autoload

php artisan migrate
```

Session

session driver session Laravel

- `file` - sessions `storage/framework/sessions`
- `cookie` - sessions cookies
- `database` - sessions
- `memcached` / `redis` - sessions
- `array` - sessions PHP

> array unit tests session

Blade

Blade Laravel Blade (template inheritance) (sections) Blade `.blade.php`

Blade

```
<!-- Stored in resources/views/layouts/master.blade.php -->

<html>
    <body>
        @section('sidebar')
            This is the master sidebar.
        @show

        <div class="container">
            @yield('content')
        </div>
    </body>
</html>
```

Blade

```
@extends('layouts.master')

@section('sidebar')
    @@parent

    <p>This is appended to the master sidebar.</p>
@stop

@section('content')
    <p>This is my body content.</p>
@stop
```

`(extend)` Blade `@@parent`

@yield @yield

```
@yield('section', 'Default Content')
```

Blade

Blade Echoing

```
Hello, {{ $name }}.

The current UNIX timestamp is {{ time() }}.
```

:

```
{{ isset($name) ? $name : 'Default' }}

Blade

{{ $name or 'Default' }}
```

@ Blade

```
@{{ This will not be processed by Blade }}
```

,

```
Hello, {!! $name !!}.
```

: HTML

If

```
@if (count($records) === 1)
    I have one record!
@elseif (count($records) > 1)
    I have multiple records!
@else
    I don't have any records!
@endif

@unless (Auth::check())
    You are not signed in.
@endunless
```

```
@for ($i = 0; $i < 10; $i++)
    The current value is {{ $i }}
@endfor

@foreach ($users as $user)
    <p>This is user {{ $user->id }}</p>
@endforeach
```

```
@forelse($users as $user)
      <li>{{ $user->name }}</li>
@empty
      <p>No users</p>
@endforelse

@while (true)
    <p>I'm looping forever.</p>
@endwhile
```

```
@include('view.name')
```

```
@include('view.name', ['some' => 'data'])
```

`overwrite`

```
@extends('list.item.container')

@section('list.item.content')
    <p>This is an item of type {{ $item->type }}</p>
@overwrite
```

```
@lang('language.line')

@choice('language.line', 1)
```

```
{{-- This comment will not be in the rendered HTML --}}
```

Blade

Blade Blade , `str_replace`

Blade `createMatcher` `createPlainMatcher`

`createPlainMatcher` `@endif` `@stop` `createMatcher`

`@datetime($var)` , `$var` `->format()`

```
Blade::extend(function($view, $compiler)
{
    $pattern = $compiler->createOpenMatcher('datetime');

    return preg_replace($pattern, '$1<?php echo $2->format(\'m/d/Y H:i\')); ?>', $view);
});
```

Laravel PHPUnit `phpunit.xml`

`tests` Laravel `phpunit`

`tests` `TestCase` PHPUnit

```
class FooTest extends TestCase {

    public function testSomethingIsTrue()
    {
        $this->assertTrue(true);
    }

}
```

`phpunit`

> : `setUp` `parent::setUp`

Laravel `testing` Laravel `session` `cache` `array` () session cache

`testing` `phpunit.xml`

`call`

```
$response = $this->call('GET', 'user/profile');

$response = $this->call($method, $uri, $parameters,  $cookies, $files, $server, $content)
```

`Illuminate\Http\Response`

```
$this->assertEquals('Hello World', $response->getContent());
```

```
$response = $this->action('GET', 'HomeController@index');

$response = $this->action('GET', 'UserController@profile', array('user' => 1));
```

: `action` `App\Http\Controllers`

`getContent` `View` `original`

```
$view = $response->original;

$this->assertEquals('John', $view['name']);
```

`callSecure` HTTPS

```
$response = $this->callSecure('GET', 'foo/bar');
```

Facades

Laravel facade

```
public function getIndex()
{
    Event::fire('foo', ['name' => 'Dayle']);

    return 'All done!';
}
```

facade `shouldReceive` `Event` Mockery mock

Facade

```
public function testGetIndex()
{
    Event::shouldReceive('fire')->once()->with('foo', ['name' => 'Dayle']);

    $this->call('GET', '/');
}
```

> : `Request` facade `call`

Assertions

Laravel `assert`

Assert OK

```
public function testMethod()
{
    $this->call('GET', '/');

    $this->assertResponseOk();
}
```

Assert

```
$this->assertResponseStatus(403);
```

Assert

```
$this->assertRedirectedTo('foo');

$this->assertRedirectedToRoute('route.name');

$this->assertRedirectedToAction('Controller@method');
```

Assert

```
public function testMethod()
{
    $this->call('GET', '/');

    $this->assertViewHas('name');
    $this->assertViewHas('age', $value);
}
```

Assert Session

```
public function testMethod()
{
    $this->call('GET', '/');
```

```
    $this->assertSessionHas('name');
    $this->assertSessionHas('age', $value);
}
```

Assert Session

```
public function testMethod()
{
    $this->call('GET', '/');

    $this->assertSessionHasErrors();

    // Asserting the session has errors for a given key...
    $this->assertSessionHasErrors('name');

    // Asserting the session has errors for several keys...
    $this->assertSessionHasErrors(array('name', 'age'));
}
```

Assert

```
public function testMethod()
{
    $this->call('GET', '/');

    $this->assertHasOldInput();
}
```

`TestCase`

Sessions

```
$this->session(['foo' => 'bar']);

$this->flushSession();
```

`be`

```
$user = new User(array('name' => 'John'));

$this->be($user);
```

`seed`

```
$this->seed();

$this->seed($connection);
```

`$this->app` `refreshApplication` IoC mocks

-
-
-
-
- &
-
-
-
-

Laravel `Validation`

```
$validator = Validator::make(
    array('name' => 'Dayle'),
    array('name' => 'required|min:5')
);
```

`make`

"|"

```
$validator = Validator::make(
    array('name' => 'Dayle'),
    array('name' => array('required', 'min:5'))
);
```

```
$validator = Validator::make(
    array(
        'name' => 'Dayle',
        'password' => 'lamepassword',
        'email' => 'email@example.com'
    ),
    array(
        'name' => 'required',
        'password' => 'required|min:8',
        'email' => 'required|email|unique:users'
    )
);
```

`Validator` `fails` `passes`

```
if ($validator->fails())
{
    // The given data did not pass validation
}
```

```
$messages = $validator->messages();
```

`failed`

```
$failed = $validator->failed();
```

`Validator` `size`, `mimes`

`after`

```
$validator = Validator::make(...);

$validator->after(function($validator)
{
    if ($this->somethingElseIsInvalid())
    {
        $validator->errors()->add('field', 'Something is wrong with this field!');
    }
});

if ($validator->fails())
{
    //
}
```

`after`

`Validator` Laravel `App\Http\Controllers\Controller` `ValidatesRequests` trait trait HTTP

```
/**
 * Store the incoming blog post.
 *
 * @param  Request  $request
 * @return Response
```

```
     */
    public function store(Request $request)
    {
        $this->validate($request, [
            'title' => 'required|unique|max:255',
            'body' => 'required',
        ]);

        //
    }
```

`Illuminate\Contracts\Validation\ValidationException` session

AJAX 422 HTTP JSON

```
/**
 * Store the incoming blog post.
 *
 * @param  Request  $request
 * @return Response
 */
public function store(Request $request)
{
    $v = Validator::make($request->all(), [
        'title' => 'required|unique|max:255',
        'body' => 'required',
    ]);

    if ($v->fails())
    {
        return redirect()->back()->withErrors($v->errors());
    }

    //
}
```

session `formatValidationErrors` `Illuminate\Validation\Validator`

```
/**
 * {@inheritdoc}
 */
protected function formatValidationErrors(Validator $validator)
{
    return $validator->errors()->all();
}
```

"" Artisan `make:request`

```
php artisan make:request StoreBlogPostRequest
```

`app/Http/Requests` `rules`

```
/**
 * Get the validation rules that apply to the request.
 *
 * @return array
 */
public function rules()
{
    return [
        'title' => 'required|unique|max:255',
        'body' => 'required',
    ];
}
```

```
/**
 * Store the incoming blog post.
 *
 * @param  StoreBlogPostRequest  $request
 * @return Response
 */
public function store(StoreBlogPostRequest $request)
{
    // The incoming request is valid...
}
```

session AJAX 422 HTTP JSON

`authorize`

```
/**
 * Determine if the user is authorized to make this request.
 *
 * @return bool
 */
public function authorize()
{
    $commentId = $this->route('comment');

    return Comment::where('id', $commentId)
                ->where('user_id', Auth::id())->exists();
}
```

`route` URI `{comment}`

```
Route::post('comment/{comment}');
```

`authorize` `false` 403 HTTP

`authorize` `true`

```
/**
 * Determine if the user is authorized to make this request.
 *
 * @return bool
 */
public function authorize()
{
    return true;
}
```

session (`App\Http\Requests\Request`) `formatValidationErrors` `Illuminate\Validation\Validator`

```
/**
 * {@inheritdoc}
 */
protected function formatErrors(Validator $validator)
{
    return $validator->errors()->all();
}
```

`Validator` `messages` `MessageBag`

```
echo $messages->first('email');
```

```
foreach ($messages->get('email') as $message)
{
    //
}
```

```
foreach ($messages->all() as $message)
{
```

```
    //
}
```

```
if ($messages->has('email'))
{
    //
}
```

```
echo $messages->first('email', '<p>:message</p>');
```

> Bootstrap

```
foreach ($messages->all('<li>:message</li>') as $message)
{
    //
}
```

&

Laravel

```
Route::get('register', function()
{
    return View::make('user.register');
});

Route::post('register', function()
{
    $rules = array(...);

    $validator = Validator::make(Input::all(), $rules);

    if ($validator->fails())
    {
        return redirect('register')->withErrors($validator);
    }
});
```

`withErrors` `Validator` session

GET Laravel Session **`$errors`** `$errors` `$errors` `MessageBag`

`$errors`

```
<?php echo $errors->first('email'); ?>
```

`MessageBag` , `withErrors`

```
return redirect('register')->withErrors($validator, 'login');
```

`$errors` `MessageBag`

```
<?php echo $errors->login->first('email'); ?>
```

- Accepted
- Active URL
- After (Date)
- Alpha
- Alpha Dash
- Alpha Numeric
- Array
- Before (Date)
- Between
- Boolean
- Confirmed
- Date
- Date Format
- Different
- Digits
- Digits Between
- E-Mail
- Exists (Database)
- Image (File)
- In
- Integer
- IP Address
- Max
- MIME Types
- Min
- Not In
- Numeric
- Regular Expression
- Required

- Required If
- Required With
- Required With All
- Required Without
- Required Without All
- Same
- Size
- String
- Timezone
- Unique (Database)
- URL

accepted

yes, *on*, *1* ""

active_url

PHP `checkdnsrr`

after:*date*

PHP `strtotime`

alpha

alpha_dash

-_

alpha_num

array

before:*date*

PHP `strtotime`

between:*min*,*max*

min *max*

confirmed

`foo_confirmation` `password` `password_confirmation` `password`

date

PHP `strtotime`

date*format:_format*

PHP `date_parse_from_format` *format*

different:*field*

field

digits:*value*

value

digits*between:_min*,*max*

min *max*

boolean

`true` , `false` , `1` , `0` , `"1"` , `"0"`

email

email

exists:*table*,*column*

table *column*

Exists

```
'state' => 'exists:states'
```

```
'state' => 'exists:states,abbreviation'
```

"where"

```
'email' => 'exists:staff,email,account_id,1'
/*  email  staff  email  account_id=1 */
```

`NULL` "where" `NULL`

```
'email' => 'exists:staff,email,deleted_at,NULL'
```

image

(jpeg, png, bmp, gif svg)

in:*foo*,*bar*,...

integer

ip

IP

max:*value*

value `size`

mimes:*foo*,*bar*,...

MIME

MIME

```
'photo' => 'mimes:jpeg,bmp,png'
```

min:*value*

value `size`

not*in:_foo*,*bar*,...

numeric

regex:*pattern*

: `regex` "|""|"

required

required_if:*field*,*value*

field *value*

required*with:_foo*,*bar*,...

required*with_all:_foo*,*bar*,...

required*without:_foo*,*bar*,...

required*without_all:_foo*,*bar*,...

same:*field*

field

size:*value*

value *value* *value* *value* kb)

timezone

PHP `timezone_identifiers_list`

unique:*table*,*column*,*except*,*idColumn*

`column`

(Unique)

```
'email' => 'unique:users'
```

```
'email' => 'unique:users,email_address'
```

ID

```
'email' => 'unique:users,email_address,10'
```

Where

"where"

```
'email' => 'unique:users,email_address,NULL,id,account_id,1'
```

`account_id` `1`

url

URL

> : PHP `filter_var`

`sometimes`

```
$v = Validator::make($data, array(
    'email' => 'sometimes|required|email',
));
```

`email` `$data`

100 `Validator`

```
$v = Validator::make($data, array(
    'email' => 'required|email',
    'games' => 'required|numeric',
));
```

`Validator` `sometimes`

```
$v->sometimes('reason', 'required|max:500', function($input)
{
    return $input->games >= 100;
});
```

`sometimes` Closure true

```
$v->sometimes(array('reason', 'cost'), 'required', function($input)
{
    return $input->games >= 100;
});
```

:

> `Closure` `$input` `Illuminate\Support\Fluent`

```
$messages = array(
    'required' => 'The :attribute field is required.',
);

$validator = Validator::make($input, $rules, $messages);
```

> `:attribute`

```
$messages = array(
    'same'    => 'The :attribute and :other must match.',
    'size'    => 'The :attribute must be exactly :size.',
    'between' => 'The :attribute must be between :min - :max.',
    'in'      => 'The :attribute must be one of the following types: :values',
);
```

```
$messages = array(
    'email.required' => 'We need to know your e-mail address!',
);
```

`Validator` `resources/lang/xx/validation.php` `custom`

```
'custom' => array(
    'email' => array(
        'required' => 'We need to know your e-mail address!',
    ),
),
```

Laravel `Validator::extend`

```
Validator::extend('foo', function($attribute, $value, $parameters)
{
    return $value == 'foo';
});
```

`$attribute()` `$value` `$parameters`

`extend`

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

Validator

Validator Validator `Illuminate\Validation\Validator` `validate`

```
<?php

class CustomValidator extends Illuminate\Validation\Validator {

    public function validateFoo($attribute, $value, $parameters)
    {
        return $value == 'foo';
    }

}
```

```
Validator::resolver(function($translator, $data, $rules, $messages)
{
    return new CustomValidator($translator, $data, $rules, $messages);
});
```

`replaceXXX`

```
protected function replaceFoo($message, $attribute, $rule, $parameters)
{
    return str_replace(':foo', $parameters[0], $message);
}
```

"replacer" Validator `Validator::replacer`

```
Validator::replacer('rule', function($message, $attribute, $rule, $parameters)
{
    //
});
```

-
- /
-
-
-
-

Laravel `config/database.php`

Laravel MySQLPostgresSQLite SQL Server

/

SELECT INSERT UPDATE DELETE Laravel Eloquent ORM

/

```
'mysql' => [
    'read' => [
        'host' => '192.168.1.1',
    ],
    'write' => [
        'host' => '196.168.1.2'
    ],
    'driver'    => 'mysql',
    'database'  => 'database',
    'username'  => 'root',
    'password'  => '',
    'charset'   => 'utf8',
    'collation' => 'utf8_unicode_ci',
    'prefix'    => '',
],
```

`read` `write` `host` `read` `write` `mysql` `read` `write`
`192.168.1.1` `192.168.1.2` `mysql`

`DB` facade

Select

```
$results = DB::select('select * from users where id = ?', [1]);
```

`select` `array`

Insert

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

Update

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

Delete

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

> `update` `delete`

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

`DB::listen`

```
DB::listen(function($sql, $bindings, $time)
{
    //
});
```

`transaction`

```
DB::transaction(function()
{
    DB::table('users')->update(['votes' => 1]);

    DB::table('posts')->delete();
});
```

> `transaction`

```
DB::beginTransaction();
```

`rollback`

```
DB::rollback();
```

`commit`

```
DB::commit();
```

`DB::connection`

```
$users = DB::connection('foo')->select(...);
```

PDO

```
$pdo = DB::connection()->getPdo();
```

```
DB::reconnect('foo');
```

PDO `max_connections` `disconnect` :

```
DB::disconnect('foo');
```

Laravel `enableQueryLog`

```
DB::connection()->enableQueryLog();
```

`getQueryLog`

```
$queries = DB::getQueryLog();
```

(query builder)

> : Laravel PDO SQL

Selects

```
$users = DB::table('users')->get();

foreach ($users as $user)
{
    var_dump($user->name);
}
```

```
DB::table('users')->chunk(100, function($users)
{
    foreach ($users as $user)
    {
        //
    }
});
```

`false`

```
DB::table('users')->chunk(100, function($users)
{
    //

    return false;
```

```
});
```

```
$user = DB::table('users')->where('name', 'John')->first();

var_dump($user->name);
```

```
$name = DB::table('users')->where('name', 'John')->pluck('name');
```

```
$roles = DB::table('roles')->lists('title');
```

role title

```
$roles = DB::table('roles')->lists('title', 'name');
```

(Select Clause)

```
$users = DB::table('users')->select('name', 'email')->get();

$users = DB::table('users')->distinct()->get();

$users = DB::table('users')->select('name as user_name')->get();
```

```
$query = DB::table('users')->select('name');

$users = $query->addSelect('age')->get();
```

where

```
$users = DB::table('users')->where('votes', '>', 100)->get();
```

or

```
$users = DB::table('users')
                    ->where('votes', '>', 100)
                    ->orWhere('name', 'John')
                    ->get();
```

Where Between

```
$users = DB::table('users')
                    ->whereBetween('votes', array(1, 100))->get();
```

Where Not Between

```
$users = DB::table('users')
                    ->whereNotBetween('votes', array(1, 100))->get();
```

Where In

```
$users = DB::table('users')
                    ->whereIn('id', array(1, 2, 3))->get();

$users = DB::table('users')
                    ->whereNotIn('id', array(1, 2, 3))->get();
```

Where Null

```
$users = DB::table('users')
                    ->whereNull('updated_at')->get();
```

(Order By)(Group By) Having

```
$users = DB::table('users')
                    ->orderBy('name', 'desc')
                    ->groupBy('count')
                    ->having('count', '>', 100)
                    ->get();
```

(Offset) (Limit)

```
$users = DB::table('users')->skip(10)->take(5)->get();
```

Joins

join

Join

```
DB::table('users')
            ->join('contacts', 'users.id', '=', 'contacts.user_id')
            ->join('orders', 'users.id', '=', 'orders.user_id')
            ->select('users.id', 'contacts.phone', 'orders.price')
            ->get();
```

Left Join

```
DB::table('users')
        ->leftJoin('posts', 'users.id', '=', 'posts.user_id')
        ->get();
```

join

```
DB::table('users')
        ->join('contacts', function($join)
        {
            $join->on('users.id', '=', 'contacts.user_id')->orOn(...);
        })
        ->get();
```

join where join `where` `orWhere` contacts user_id

```
DB::table('users')
        ->join('contacts', function($join)
        {
            $join->on('users.id', '=', 'contacts.user_id')
                 ->where('contacts.user_id', '>', 5);
        })
        ->get();
```

Wheres

where where existsLaravel

```
DB::table('users')
            ->where('name', '=', 'John')
            ->orWhere(function($query)
            {
                $query->where('votes', '>', 100)
                      ->where('title', '<>', 'Admin');
            })
            ->get();
```

SQL

```
select * from users where name = 'John' or (votes > 100 and title <> 'Admin')
```

Exists

```
DB::table('users')
            ->whereExists(function($query)
            {
```

```
                $query->select(DB::raw(1))
                      ->from('orders')
                      ->whereRaw('orders.user_id = users.id');
            })
            ->get();
```

SQL

```
select * from users
where exists (
    select 1 from orders where orders.user_id = users.id
)
```

`count` `max` `min` `avg` `sum`

```
$users = DB::table('users')->count();

$price = DB::table('orders')->max('price');

$price = DB::table('orders')->min('price');

$price = DB::table('orders')->avg('price');

$total = DB::table('users')->sum('votes');
```

SQL `DB::raw`

```
$users = DB::table('users')
                     ->select(DB::raw('count(*) as user_count, status'))
                     ->where('status', '<>', 1)
                     ->groupBy('status')
                     ->get();
```

```
DB::table('users')->insert(
    array('email' => 'john@example.com', 'votes' => 0)
);
```

(Auto-Incrementing) ID

ID `insertGetId` ID

```
$id = DB::table('users')->insertGetId(
    array('email' => 'john@example.com', 'votes' => 0)
);
```

> : PostgreSQL insertGetId id

```
DB::table('users')->insert(array(
    array('email' => 'taylor@example.com', 'votes' => 0),
    array('email' => 'dayle@example.com', 'votes' => 0),
));
```

```
DB::table('users')
            ->where('id', 1)
            ->update(array('votes' => 1));
```

```
DB::table('users')->increment('votes');

DB::table('users')->increment('votes', 5);

DB::table('users')->decrement('votes');

DB::table('users')->decrement('votes', 5);
```

```
DB::table('users')->increment('votes', 1, array('name' => 'John'));
```

```
DB::table('users')->where('votes', '<', 100)->delete();
```

```
DB::table('users')->delete();
```

```
DB::table('users')->truncate();
```

Unions

(union)

```
$first = DB::table('users')->whereNull('first_name');

$users = DB::table('users')->whereNull('last_name')->union($first)->get();
```

`unionAll` `union`

(Pessimistic Locking)

SELECT

SELECT Shard lock `sharedLock`

```
DB::table('users')->where('votes', '>', 100)->sharedLock()->get();
```

select (lock for update) `lockForUpdate`

```
DB::table('users')->where('votes', '>', 100)->lockForUpdate()->get();
```

Eloquent ORM

Laravel Eloquent ORM ActiveRecord

`config/database.php`

Eloquent `app` composer.json Eloquent `Illuminate\Database\Eloquent\Model`

Eloquent

```
class User extends Model {}
```

`make:model` Eloquent

```
php artisan make:model User
```

Eloquent User Eloquent `User` `users` `table`

```
class User extends Model {

    protected $table = 'my_users';

}
```

> Eloquent `id` `primaryKey` `connection`
> `updated_at` `created_at` `$timestamps` false

```
$users = User::all();
```

```
$user = User::find(1);

var_dump($user->name);
```

> Eloquent

, `App::error` 404

```
$model = User::findOrFail(1);

$model = User::where('votes', '>', 100)->firstOrFail();
```

`ModelNotFoundException`

```
use Illuminate\Database\Eloquent\ModelNotFoundException;

App::error(function(ModelNotFoundException $e)
{
    return Response::make('Not Found', 404);
});
```

Eloquent

```
$users = User::where('votes', '>', 100)->take(10)->get();

foreach ($users as $user)
{
    var_dump($user->name);
}
```

Eloquent

```
$count = User::where('votes', '>', 100)->count();
```

`whereRaw`

```
$users = User::whereRaw('age > ? and votes = 100', array(25))->get();
```

Eloquent `chunk`

```
User::chunk(200, function($users)
{
    foreach ($users as $user)
    {
        //
    }
});
```

Eloquent `on`

```
$user = User::on('connection-name')->find(1);
```

/ ,

```
$user = User::onWriteConnection()->find(1);
```

Eloquent

`fillable` `guarded`

`Fillable`

`fillable`

```
class User extends Model {

    protected $fillable = array('first_name', 'last_name', 'email');
```

```
}
```

Guarded

`guarded` `fillable`

```
class User extends Model {

    protected $guarded = array('id', 'password');

}
```

> `guarded` `Input::get()` `save` `update`

`id` `password` guard

```
protected $guarded = array('*');
```

`save`

```
$user = new User;

$user->name = 'John';

$user->save();
```

> Eloquent `incrementing` false

`create` `fillable` `guarded` Eloquent

`id`

```
$insertedId = $user->id;
```

Guarded

```
class User extends Model {

    protected $guarded = array('id', 'account_id');
```

```
}
```

Create

```
// ...
$user = User::create(array('name' => 'John'));

// ...
$user = User::firstOrCreate(array('name' => 'John'));

// ...
$user = User::firstOrNew(array('name' => 'John'));
```

`save`

```
$user = User::find(1);

$user->email = 'john@foo.com';

$user->save();
```

`push`

```
$user->push();
```

```
$affectedRows = User::where('votes', '>', 100)->update(array('status' => 2));
```

> Eloquent

`delete`

```
$user = User::find(1);

$user->delete();
```

```
User::destroy(1);
```

```
User::destroy(array(1, 2, 3));

User::destroy(1, 2, 3);
```

```
$affectedRows = User::where('votes', '>', 100)->delete();
```

`touch`

```
$user->touch();
```

`deleted_at` `SoftDeletingTrait`

```
use Illuminate\Database\Eloquent\SoftDeletes;

class User extends Model {

    use SoftDeletes;

    protected $dates = ['deleted_at'];

}
```

`deleted_at` S `oftDeletes`

```
$table->softDeletes();
```

`delete` `deleted_at`

`withTrashed`

```
$users = User::withTrashed()->where('account_id', 1)->get();
```

`withTrashed`

```
$user->posts()->withTrashed()->get();
```

`onlyTrashed`

```
$users = User::onlyTrashed()->where('account_id', 1)->get();
```

`restore`

```
$user->restore();
```

`restore`

```
User::withTrashed()->where('account_id', 1)->restore();
```

`withTrashed` `restore`

```
$user->posts()->restore();
```

`forceDelete`

```
$user->forceDelete();
```

`forceDelete`

```
$user->posts()->forceDelete();
```

`trashed`

```
if ($user->trashed())
{
    //
}
```

Eloquent `created_at` `updated_at` Eloquent Eloquent

```
class User extends Model {

    protected $table = 'users';

    public $timestamps = false;

}
```

`getDateFormat`

```
class User extends Model {

    protected function getDateFormat()
    {
        return 'U';
    }

}
```

`scope`

```
class User extends Model {

    public function scopePopular($query)
    {
        return $query->where('votes', '>', 100);
    }

    public function scopeWomen($query)
    {
        return $query->whereGender('W');
    }

}
```

```
$users = User::popular()->women()->orderBy('created_at')->get();
```

```
class User extends Model {

    public function scopeOfType($query, $type)
    {
        return $query->whereType($type);
    }

}
```

```
$users = User::ofType('member')->get();
```

Global Scopes

scope Eloquent ""Global scopes PHP traits
`Illuminate\Database\Eloquent\ScopeInterface`

trait Laravel `SoftDeletes`

```
trait SoftDeletes {

    /**
     * Boot the soft deleting trait for a model.
     *
     * @return void
     */
    public static function bootSoftDeletes()
    {
        static::addGlobalScope(new SoftDeletingScope);
    }

}
```

Eloquent trait trait `bootNameOfTrait` , Eloquent global scope scope
`ScopeInterface` `apply` `remove`

`apply` `Illuminate\Database\Eloquent\Builder` `Model` scope `where` `remove`
`Builder` `Model` `apply` `remove` `where` () `SoftDeletingScope`

```
/**
 * Apply the scope to a given Eloquent query builder.
 *
 * @param  \Illuminate\Database\Eloquent\Builder  $builder
 * @return void
 */
public function apply(Builder $builder, Model $model)
{
    $model = $builder->getModel();

    $builder->whereNull($model->getQualifiedDeletedAtColumn());
}

/**
 * Remove the scope from the given Eloquent query builder.
 *
 * @param  \Illuminate\Database\Eloquent\Builder  $builder
 * @return void
 */
public function remove(Builder $builder, Model $model)
{
    $column = $model->getQualifiedDeletedAtColumn();

    $query = $builder->getQuery();

    foreach ((array) $query->wheres as $key => $where)
```

```
    {
        // If the where clause is a soft delete date constraint, we will remove it from
        // the query and reset the keys on the wheres. This allows this developer to
        // include deleted model in a relationship result set that is lazy loaded.
        if ($this->isSoftDeleteConstraint($where, $column))
        {
            unset($query->wheres[$key]);

            $query->wheres = array_values($query->wheres);
        }
    }
}
```

---

blog Eloquent Laravel

-
-
-
-
-
-

`User` `Phone` Eloquent

```
class User extends Model {

    public function phone()
    {
        return $this->hasOne('App\Phone');
    }

}
```

`hasOne` Eloquent

```
$phone = User::find(1)->phone;
```

SQL

```
select * from users where id = 1

select * from phones where user_id = 1
```

Eloquent `Phone` `user_id` `hasOne`

```
return $this->hasOne('App\Phone', 'foreign_key');

return $this->hasOne('App\Phone', 'foreign_key', 'local_key');
```

`Phone` `belongsTo`

```
class Phone extends Model {

    public function user()
    {
        return $this->belongsTo('App\User');
    }

}
```

Eloquent `phones` `user_id` `belongsTo`

```
class Phone extends Model {

    public function user()
    {
        return $this->belongsTo('App\User', 'local_key');
    }

}
```

```
class Phone extends Model {

    public function user()
    {
        return $this->belongsTo('App\User', 'local_key', 'parent_key');
    }

}
```

Blog

```
class Post extends Model {

    public function comments()
    {
        return $this->hasMany('App\Comment');
    }

}
```

```
$comments = Post::find(1)->comments;
```

`comments`

```
$comments = Post::find(1)->comments()->where('title', '=', 'foo')->first();
```

`hasMany` `hasOne`

```
return $this->hasMany('App\Comment', 'foreign_key');

return $this->hasMany('App\Comment', 'foreign_key', 'local_key');
```

`Comment` `belongsTo`

```
class Comment extends Model {

    public function post()
    {
        return $this->belongsTo('App\Post');
    }

}
```

user role `users` `roles` `role_user` `role_user` `user_id` `role_id`

`belongsToMany`

```
class User extends Model {

    public function roles()
    {
        return $this->belongsToMany('App\Role');
    }

}
```

`User` roles

```
$roles = User::find(1)->roles;
```

`belongsToMany`

```
return $this->belongsToMany('App\Role', 'user_roles');
```

```
return $this->belongsToMany('App\Role', 'user_roles', 'user_id', 'foo_id');
```

`Role`

```
class Role extends Model {

    public function users()
    {
        return $this->belongsToMany('App\User');
    }

}
```

Has Many Through

`Country` `Users` `Posts`

```
countries
    id - integer
    name - string

users
    id - integer
    country_id - integer
    name - string

posts
    id - integer
    user_id - integer
    title - string
```

`posts` `country_id` `hasManyThrough` `$country->posts` country posts

```
class Country extends Model {

    public function posts()
    {
        return $this->hasManyThrough('App\Post', 'User');
    }

}
```

```
class Country extends Model {
```

```
    public function posts()
    {
        return $this->hasManyThrough('App\Post', 'User', 'country_id', 'user_id');
    }

}
```

photo staff order

```
class Photo extends Model {

    public function imageable()
    {
        return $this->morphTo();
    }

}

class Staff extends Model {

    public function photos()
    {
        return $this->morphMany('App\Photo', 'imageable');
    }

}

class Order extends Model {

    public function photos()
    {
        return $this->morphMany('App\Photo', 'imageable');
    }

}
```

staff order

```
$staff = Staff::find(1);

foreach ($staff->photos as $photo)
{
    //
}
```

`Photo` staff order

```
$photo = Photo::find(1);
```

```
$imageable = $photo->imageable;
```

Photo `imageable` `Staff` `Order`

```
staff
    id - integer
    name - string

orders
    id - integer
    price - integer

photos
    id - integer
    path - string
    imageable_id - integer
    imageable_type - string
```

`photos` `imageable_id` `imageable_type` ID staff order ID type ORM `imageable`

Polymorphic Many To Many Relation Table Structure

Blog `Post` `Video` `Tag`

```
posts
    id - integer
    name - string

videos
    id - integer
    name - string

tags
    id - integer
    name - string

taggables
    tag_id - integer
    taggable_id - integer
    taggable_type - string
```

`Post` `Video` `tags` `morphToMany`

```
class Post extends Model {

    public function tags()
    {
```

```
        return $this->morphToMany('App\Tag', 'taggable');
    }

}
```

`Tag`

```
class Tag extends Model {

    public function posts()
    {
        return $this->morphedByMany('App\Post', 'taggable');
    }

    public function videos()
    {
        return $this->morphedByMany('App\Video', 'taggable');
    }

}
```

Blog `has`

```
$posts = Post::has('comments')->get();
```

```
$posts = Post::has('comments', '>=', 3)->get();
```

"" `has`

```
$posts = Post::has('comments.votes')->get();
```

`whereHas` `orWhereHas` `has` "where"

```
$posts = Post::whereHas('comments', function($q)
{
    $q->where('content', 'like', 'foo%');

})->get();
```

Eloquent Eloquent `get` `first` `$phone`

```
class Phone extends Model {

    public function user()
    {
        return $this->belongsTo('App\User');
    }

}

$phone = Phone::find(1);
```

email

```
echo $phone->user()->first()->email;
```

```
echo $phone->user->email;
```

`Illuminate\Database\Eloquent\Collection`

N + 1 `Book` `Author`

```
class Book extends Model {

    public function author()
    {
        return $this->belongsTo('App\Author');
    }

}
```

```
foreach (Book::all() as $book)
{
    echo $book->author->name;
}
```

25 26

`with`

```
foreach (Book::with('author')->get() as $book)
{
    echo $book->author->name;
}
```

```
select * from books

select * from authors where id in (1, 2, 3, 4, 5, ...)
```

```
$books = Book::with('author', 'publisher')->get();
```

```
$books = Book::with('author.contacts')->get();
```

`author` author `contacts`

```
$users = User::with(array('posts' => function($query)
{
    $query->where('title', 'like', '%first%');

}))->get();
```

user posts post title "first"

```
$users = User::with(array('posts' => function($query)
{
    $query->orderBy('created_at', 'desc');

}))->get();
```

collection

```
$books = Book::all();

$books->load('author', 'publisher');
```

```
$books->load(['author' => function($query)
{
    $query->orderBy('published_date', 'asc');
}]);
```

---

comment post `post_id` `Post` comment

```
$comment = new Comment(array('message' => 'A new comment.'));

$post = Post::find(1);

$comment = $post->comments()->save($comment);
```

comment `post_id`

```
$comments = array(
    new Comment(array('message' => 'A new comment.')),
    new Comment(array('message' => 'Another comment.')),
    new Comment(array('message' => 'The latest comment.'))
);

$post = Post::find(1);

$post->comments()->saveMany($comments);
```

(Belongs To)

`belongsTo` `associate`

```
$account = Account::find(10);

$user->account()->associate($account);

$user->save();
```

(Many To Many)

`User` `Role` `attach` roles user

```
$user = User::find(1);

$user->roles()->attach(1);
```

```
$user->roles()->attach(1, array('expires' => $expires));
```

`attach` `detach`

```
$user->roles()->detach(1);
```

`attach` `detach` ID

```
$user = User::find(1);

$user->roles()->detach([1, 2, 3]);

$user->roles()->attach([1 => ['attribute1' => 'value1'], 2, 3]);
```

Sync

`sync` `sync` ID ID id

```
$user->roles()->sync(array(1, 2, 3));
```

Sync

ID

```
$user->roles()->sync(array(1 => array('expires' => true)));
```

`save`

```
$role = new Role(array('name' => 'Editor'));

User::find(1)->roles()->save($role);
```

`Role` `user`

```
User::find(1)->roles()->save($role, array('expires' => $expires));
```

`belongsTo` `Comment` `Post` `Comment` `Post` `updated_at`
Eloquent `touches`

```
class Comment extends Model {

    protected $touches = array('post');

    public function post()
    {
        return $this->belongsTo('App\Post');
    }

}
```

`Comment` `Post` `updated_at`

```
$comment = Comment::find(1);

$comment->text = 'Edit to this comment!';

$comment->save();
```

Eloquent `User` `Role` `pivot`

```
$user = User::find(1);

foreach ($user->roles as $role)
{
    echo $role->pivot->created_at;
}
```

`Role` `pivot` Eloquent

`pivot` pivot

```
return $this->belongsToMany('App\Role')->withPivot('foo', 'bar');
```

`Role` `pivot` `foo` `bar`

`created_at` `updated_at` `withTimestamps`

```
return $this->belongsToMany('App\Role')->withTimestamps();
```

`detach`

```
User::find(1)->roles()->detach();
```

`roles`

`updateExistingPivot`

```
User::find(1)->roles()->updateExistingPivot($roleId, $attributes);
```

Laravel Eloquent Eloquent Eloquent

```
public function newPivot(Model $parent, array $attributes, $table, $exists)
{
    return new YourCustomPivot($parent, $attributes, $table, $exists);
}
```

Eloquent `get` `relationship` `IteratorAggregate` PHP

`contains`

```
$roles = User::find(1)->roles;

if ($roles->contains(2))
{
    //
}
```

JSON

```
$roles = User::find(1)->roles->toArray();

$roles = User::find(1)->roles->toJson();
```

JSON

```
$roles = (string) User::find(1)->roles;
```

Eloquent

```
$roles = $user->roles->each(function($role)
{
    //
});
```

array_filter)

```
$users = $users->filter(function($user)
{
    return $user->isAdmin();
});
```

JSON `values`

```
$roles = User::find(1)->roles;

$roles->each(function($role)
{
    //
});
```

```
$roles = $roles->sortBy(function($role)
{
    return $role->created_at;
});
```

```
$roles = $roles->sortBy('created_at');
```

Eloquent `newCollection`

```
class User extends Model {

    public function newCollection(array $models = array())
    {
        return new CustomCollection($models);
    }

}
```

Eloquent `getFooAttribute` database

```
class User extends Model {

    public function getFirstNameAttribute($value)
    {
        return ucfirst($value);
    }

}
```

first_name

```
class User extends Model {

    public function setFirstNameAttribute($value)
    {
        $this->attributes['first_name'] = strtolower($value);
    }

}
```

Eloquent `created_at` `updated_at` Carbon PHP `DateTime`

`getDates`

```
public function getDates()
{
    return array('created_at');
}
```

UNIX timestamp Y-m-d date-time `DateTime` `Carbon`

`getDates`

```
public function getDates()
{
    return array();
}
```

, `casts` `casts`

```
/**
 *
 *
 * @var array
 */
protected $casts = [
    'is_admin' => 'boolean',
];
```

`is_admin` `integer` , `real` , `float` , `double` , `string` , `boolean` , `object` `array`

JSON `array` TEXT JSON `array` , PHP

```
/**
 *
 *
 * @var array
 */
protected $casts = [
    'options' => 'array',
];
```

Eloquent

```
$user = User::find(1);

// $options ...
$options = $user->options;

// options  JSON...
$user->options = ['foo' => 'bar'];
```

Eloquent `creating` , `created` , `updating` , `updated` , `saving` , `saved` , `deleting` , `deleted` , `restoring` , `restored`

`creating` `created` `save` `updating` / `updated` `saving` / `saved`

`creating` `updating` `saving` `deleting` false

```
User::creating(function($user)
{
    if ( ! $user->isValid()) return false;
});
```

`EventServiceProvider`

```
/**
 * Register any other events for your application.
 *
 * @param  \Illuminate\Contracts\Events\Dispatcher  $events
 * @return void
 */
public function boot(DispatcherContract $events)
{
    parent::boot($events);

    User::creating(function($user)
    {
        //
    });
}
```

`creating` `updating` `saving`

```
class UserObserver {

    public function saving($model)
    {
        //
    }

    public function saved($model)
    {
        //
    }

}
```

`observe`

```
User::observe(new UserObserver);
```

URL

`route` `action` URI

```
Route::get('user/{user}', 'UserController@show');

action('UserController@show', [$user]);
```

`$user->id` URL `{user}` ID `getRouteKey`

```
public function getRouteKey()
{
    return $this->slug;
}
```

/ JSON

JSON API JSONEloquent `toArray`

```
$user = User::with('roles')->first();

return $user->toArray();
```

```
return User::all()->toArray();
```

JSON

JSON `toJson`

```
return User::find(1)->toJson();
```

JSON Eloquent

```
Route::get('users', function()
{
    return User::all();
});
```

JSON

JSON `hidden`

```
class User extends Model {

    protected $hidden = array('password');

}
```

`visible`

```
protected $visible = array('first_name', 'last_name');
```

```
public function getIsAdminAttribute()
{
    return $this->attributes['admin'] == 'yes';
}
```

`appends`

```
protected $appends = array('is_admin');
```

`appends` JSON `appends` `visible` `hidden`

-
-
-
-
-
-
-
-
-
-
-

Laravel (`Schema`) Laravel API

`Schema::create`

```
Schema::create('users', function($table)
{
    $table->increments('id');
});
```

`create` `Closure` `Blueprint`

`rename`

```
Schema::rename($from, $to);
```

`Schema::connection`

```
Schema::connection('foo')->create('users', function($table)
{
    $table->increments('id');
});
```

`Schema::drop`

```
Schema::drop('users');

Schema::dropIfExists('users');
```

`Schema::table`

```
Schema::table('users', function($table)
{
    $table->string('email');
});
```

`$table->bigIncrements('id');`	ID big integer
`$table->bigInteger('votes');`	BIGINT
`$table->binary('data');`	BLOB
`$table->boolean('confirmed');`	BOOLEAN
`$table->char('name', 4);`	CHAR
`$table->date('created_at');`	DATE
`$table->dateTime('created_at');`	DATETIME
`$table->decimal('amount', 5, 2);`	DECIMAL
`$table->double('column', 15, 8);`	DOUBLE 15 8
`$table->enum('choices', array('foo', 'bar'));`	ENUM
`$table->float('amount');`	FLOAT
`$table->increments('id');`	Incrementing ()
`$table->integer('votes');`	INTEGER
`$table->json('options');`	JSON
`$table->longText('description');`	LONGTEXT
`$table->mediumInteger('numbers');`	MEDIUMINT
`$table->mediumText('description');`	MEDIUMTEXT
`$table->morphs('taggable');`	`taggable_id` `taggable_type`
`$table->nullableTimestamps();`	`timestamps()` NULL
`$table->smallInteger('votes');`	SMALLINT
`$table->tinyInteger('numbers');`	TINYINT
`$table->softDeletes();`	**deleted_at**
`$table->string('email');`	VARCHAR
`$table->string('name', 100);`	VARCHAR
`$table->text('description');`	TEXT
`$table->time('sunrise');`	TIME
`$table->timestamp('added_on');`	TIMESTAMP
`$table->timestamps();`	**created_at** **updated_at**

`$table->rememberToken();`	`remember_token` VARCHAR(100) NULL
`->nullable()`	NULL
`->default($value)`	
`->unsigned()`	

MySQL After

MySQL `after`

```
$table->string('name')->after('email');
```

`change` `name` 25 50

```
Schema::table('users', function($table)
{
    $table->string('name', 50)->change();
});
```

NULL

```
Schema::table('users', function($table)
{
    $table->string('name', 50)->nullable()->change();
});
```

`renameColumn` `composer.json` `doctrine/dbal`

```
Schema::table('users', function($table)
{
    $table->renameColumn('from', 'to');
});
```

: `enum`

`dropColumn` `composer.json` `doctrine/dbal`

```
Schema::table('users', function($table)
{
```

```
    $table->dropColumn('votes');
});
```

```
Schema::table('users', function($table)
{
    $table->dropColumn(array('votes', 'avatar', 'location'));
});
```

`hasTable` `hasColumn`

```
if (Schema::hasTable('users'))
{
    //
}
```

```
if (Schema::hasColumn('users', 'email'))
{
    //
}
```

```
$table->string('email')->unique();
```

`$table->primary('id');`	(primary key)
`$table->primary(array('first', 'last'));`	(composite keys)
`$table->unique('email');`	(unique index)
`$table->index('state');`	(index)

```
$table->integer('user_id')->unsigned();
$table->foreign('user_id')->references('id')->on('users');
```

`user_id` `users` `id`

on deleteon update

```
$table->foreign('user_id')
      ->references('id')->on('users')
      ->onDelete('cascade');
```

`dropForeign`

```
$table->dropForeign('posts_user_id_foreign');
```

> : `unsigned`

Laravel

`$table->dropPrimary('users_id_primary');`	users
`$table->dropUnique('users_email_unique');`	users
`$table->dropIndex('geo_state_index');`	geo

`timestamps` `nullableTimestamps` `softDeletes`

`$table->dropTimestamps();`	**created_at** **updated_at**
`$table->dropSoftDeletes();`	**deleted_at**

`engine`

```
Schema::create('users', function($table)
{
    $table->engine = 'InnoDB';

    $table->string('email');
});
```

-
-
-
-
-

Artisan CLI `make:migrate`

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

`database/migrations`

`--path`

```
php artisan make:migration foo --path=app/migrations
```

`--table` `--create`

```
php artisan make:migration add_votes_to_users_table --table=users

php artisan make:migration create_users_table --create=users
```

```
php artisan migrate
```

> : class not found `composer dump-autoload`

(Production)

`--force`

```
php artisan migrate --force
```

```
php artisan migrate:rollback
```

```
php artisan migrate:reset
```

```
php artisan migrate:refresh

php artisan migrate:refresh --seed
```

Laravel seed seed `database/seeds` `UserTableSeeder` `DatabaseSeeder` `call` seed

Seed

```
class DatabaseSeeder extends Seeder {

    public function run()
    {
        $this->call('UserTableSeeder');

        $this->command->info('User table seeded!');
    }

}

class UserTableSeeder extends Seeder {

    public function run()
    {
        DB::table('users')->delete();

        User::create(array('email' => 'foo@bar.com'));
    }

}
```

Artisan CLI `db:seed`

```
php artisan db:seed
```

`db:seed` `DatabaseSeeder` seed `--class`

```
php artisan db:seed --class=UserTableSeeder
```

`migrate:refresh`

```
php artisan migrate:refresh --seed
```

Redis

-
-
-
-

Redis

Redis Composer `predis/predis` Laravel

> PECL Redis PHP extension `config/app.php` Redis

Redis `config/database.php` **redis** Redis

```
'redis' => array(

    'cluster' => true,

    'default' => array('host' => '127.0.0.1', 'port' => 6379),

),
```

Redis host port

`cluster` Laravel Redis Redis client-side sharding RAM

Redis Redis `password`

`Redis::connection` Redis

```
$redis = Redis::connection();
```

Redis `connection` Redis

```
$redis = Redis::connection('other');
```

Redis Redis Laravel

```
$redis->set('name', 'Taylor');
```

```
$name = $redis->get('name');

$values = $redis->lrange('names', 5, 10);
```

`command`

```
$values = $redis->command('lrange', array(5, 10));
```

`Redis`

```
Redis::set('name', 'Taylor');

$name = Redis::get('name');

$values = Redis::lrange('names', 5, 10);
```

> Redis Laravel

`pipeline`

```
Redis::pipeline(function($pipe)
{
    for ($i = 0; $i < 1000; $i++)
    {
        $pipe->set("key:$i", $i);
    }
});
```

Artisan

Artisan Laravel Symfony Console

Artisan `list`

```
php artisan list
```

`help`

```
php artisan help migrate
```

`--env`

```
php artisan migrate --env=local
```

Laravel

`--version` Laravel

```
php artisan --version
```

Artisan HTTP Artisan `Artisan` facade

```
Route::get('/foo', function()
{
    $exitCode = Artisan::call('command:name', ['--option' => 'foo']);
```

```
    //
});
```

Artisan

```
Route::get('/foo', function()
{
    Artisan::queue('command:name', ['--option' => 'foo']);

    //
});
```

Artisan

Cron SSH Cron Laravel Laravel Cron

`app/Console/Kernel.php` `schedule` `Schedule` Cron

```
* * * * * php /path/to/artisan schedule:run 1>> /dev/null 2>&1
```

Cron Laravel Laravel

```
$schedule->call(function()
{
    // ...

})->hourly();
```

```
$schedule->exec('composer self-update')->daily();
```

Cron

```
$schedule->command('foo')->cron('* * * * *');
```

```
$schedule->command('foo')->everyFiveMinutes();
```

```
$schedule->command('foo')->everyTenMinutes();

$schedule->command('foo')->everyThirtyMinutes();
```

```
$schedule->command('foo')->daily();
```

(24)

```
$schedule->command('foo')->dailyAt('15:00');
```

```
$schedule->command('foo')->twiceDaily();
```

```
$schedule->command('foo')->weekdays();
```

```
$schedule->command('foo')->weekly();

//  (0-6) ...
$schedule->command('foo')->weeklyOn(1, '8:00');
```

```
$schedule->command('foo')->monthly();
```

```
$schedule->command('foo')->mondays();
$schedule->command('foo')->tuesdays();
$schedule->command('foo')->wednesdays();
$schedule->command('foo')->thursdays();
$schedule->command('foo')->fridays();
$schedule->command('foo')->saturdays();
$schedule->command('foo')->sundays();
```

```
$schedule->command('foo')->monthly()->environments('production');
```

```
$schedule->command('foo')->monthly()->evenInMaintenanceMode();
```

true

```
$schedule->command('foo')->monthly()->when(function()
{
    return true;
});
```

E-mail

```
$schedule->command('foo')->emailOutputTo('foo@example.com');
```

```
$schedule->command('foo')->sendOutputTo($filePath);
```

Ping URL

```
$schedule->command('foo')->thenPing($url);
```

Artisan

-
-
-

Artisan `app/Console/commands` `composer.json`

Class

`make:console` Artisan Command stub

```
php artisan make:console FooCommand
```

`app/Console/FooCommand.php`

`--command`

```
php artisan make:console AssignUsers --command=users:assign
```

`name` `description` Artisan `list`

`fire`

`getArguments` `getOptions`

`arguments`

```
array($name, $mode, $description, $defaultValue)
```

`mode` `InputArgument::REQUIRED` `InputArgument::OPTIONAL`

`options`

```
array($name, $shortcut, $mode, $description, $defaultValue)
```

`mode` `InputOption::VALUE_REQUIRED`, `InputOption::VALUE_OPTIONAL`, `InputOption::VALUE_IS_ARRAY`, `InputOption::VALUE_NONE`

`VALUE_IS_ARRAY`

```
php artisan foo --option=bar --option=baz
```

`VALUE_NONE`

```
php artisan foo --option
```

`argument` `option`

```
$value = $this->argument('name');
```

```
$arguments = $this->argument();
```

```
$value = $this->option('name');
```

```
$options = $this->option();
```

`info` `comment` `question` `error` ANSI

```
$this->info('Display this on the screen');
```

```
$this->error('Something went wrong!');
```

`ask` `confirm`

```
$name = $this->ask('What is your name?');
```

```
$password = $this->secret('What is the password?');
```

```
if ($this->confirm('Do you wish to continue? [yes|no]'))
{
    //
}
```

`confirm` `true` `false`

```
$this->confirm($question, true);
```

`call`

```
$this->call('command:name', ['argument' => 'foo', '--option' => 'bar']);
```

Artisan

Artisan `app/Console/Kernel.php` `commands` Artisan service container Artisan