Route

```
route responds to GET
Route::get('/', function() {return "Hello World!"; });

route for any HTTP
Route::any('/', function() {return "Hello World!";});

other methods
Route::post('user', function() {});
Route::put('user/(:num)', function($id) {});
Route::delete('user/(:num)', function($id) {});

many methods at once
Router::register(array('GET', 'POST'), $uri, $callback);
```

wildcards

```
only digits: Route::get('user/(:num)', function($id) {});
alfa numeric: Route::get('post/(:any)', function($title){});
optional:Route::get('page/(:any?)', function($page = 'index'){});

error 404
Event::listen('404', function()
{   return Response::error('404'); });
```

filters

```
registering a filter:
Route::filter('filter', function()
{ return Redirect::to('home'); });

attaching a filter to a route:
Route::get('blocked', array('before' => 'filter', function()
{   return View::make('blocked'); }));

attaching an "after" filter to a route:
Route::get('download', array('after' => 'log', function()
{   //  }));

attaching multiple filters to a route:
Route::get('create', array('before' => 'auth|csrf', function()
{   // }));

passing parameters to filters:
Route::get('panel', array('before' => 'role:admin', function()
{   // }));

pattern Filters: Route::filter('pattern: admin/*', 'auth');
```

global Filters

```
Route::filter('before','auth');
Route::filter('after','auth');
Route::filter('csrf',function(){});
Route::filter('auth',function(){});
```

attaching filter

```
->only(array('index', 'list'));
->except(array('add', 'posts'));
->on('post');
```

route Groups

```
Route::group(array('before' => 'auth'), function()
{   Route::get('panel', function()    { ...  });
    Route::get('dashboard', function()    { ....  });  });

name a route: Route::get('/', array('as' => 'home', function(){});

generating a URL to a named route: $url = URL::to_route('home');

redirecting to the named route: return Redirect::to_route('home');

if (Request::route()->is('home'))
{   // The "home" route is handling the request! }

HTTPS route: Route::get('login', array('https' => true, function()
{   return View::make('login'); }));
short-cut method: Route::secure('GET', 'login', function()
{   return View::make('login');});

registering a bundle to handle routes:
return array(   'admin' => array('handles' => 'admin'), );

Route::get('(:bundle)/panel', function(){
    return "I handle requests to admin/panel!";});

register all controllers for the application:
Route::controller(Controller::detect());

register all controllers for the "admin" bundle:
Route::controller(Controller::detect('admin'));

registering several controllers with the router:
Route::controller(array("dashboard.panel", "admin"));

registering a named route that points to a controller action:
Route::get('welcome', array('as' => 'home.welcome', 'uses' => 'home@index'));

calling a route via the Artisan CLI:
php artisan route:call get api/user/1
```

Controller

```
class Admin_Controller extends Base_Controller
{
    public function action_index()
    { .... }
}

bundle controller example (admin bundle - home controller)
class Admin_Home_Controller extends Base_Controller{ .... }

register: Route::controller('admin::home');
```

restful controller:

```
class Home_Controller extends Base_Controller
{
    public $restful = true;
    public function get_index() { ... }
    public function post_index() { ... }
}
```

Dependency injection

```
Application/start.php
IoC::register('controller: user', function()
{   return new User_Controller; });

singleton version: IoC::singleton('mailer', function()
{
    //
});

Event::listen(Controller::factory, function($controller)
{   return new $controller; });

$db = IoC::resolve('obiect');

register an existing instance: IoC::instance('mailer', $instance);
```

Views

```
views/home/index.php
return a view: return View::make('home.index');
check if a view exists: $exists = View::exists('home.index');
HTTP code: return Response::view('home', 200, $headers);
json: return Response::json(array('name' => 'Batman'));

bind data:
->with('name', 'James')
->with('votes', 25);

View::make('home', array('name' => 'James'));

nesting view:
View::make('home')->nest('footer', 'partials.footer');

$view = View::make('home');
$view->nest('content', 'orders', array('orders' => $orders));

naming view:
View::name('layouts.default', 'layout');
return View::of('layout', array('orders' => $orders));
View::composer('home', function($view)

register a view composer in application/routes.php
View::composer('home', function($view)
{   $view->nest('footer', 'partials.footer'); });
```

Blade template

```
Hello, {{$name}}.
{{Asset::styles() }}
{{-- This is a comment --}}

@SECTION('SCRIPTS')
    <SCRIPT SRC="/JQUERY.JS"></SCRIPT>
@ENDSECTION

<HEAD>
    @YIELD('SCRIPTS')
</HEAD>

Loop and control structures
@foreach ($comments as $comment)
    The comment body is {{$comment->body}}.
@endforeach

@if (count($comments) > 0)
    I have comments!
@else
    I have no comments!
@endif

@for ($i =0; $i < count($comments) - 1; $i++)
    The comment body is {{$comments[$i]}}
@endfor

@while ($something)
    I am still looping!
@endwhile

@forelse ($posts as $post)
    {{$post->body }}
@empty
    There are not posts in the array!
@endforelse

<?php if ( ! Auth::check()): ?>:
@unless(Auth::check())
    {{HTML::link_to_route('login', 'Login'); }}
@endunless

Master.blade.php
<html>
    <ul class="navigation">
        @section('navigation')
            <li>Nav Item 1</li>
            <li>Nav Item 2</li>
        @yield_section
    </ul>

    <div class="content">
        @yield('content')
    </div>
</html>

using layout
@layout('master')

@section('navigation')
    @parent
    <li>Nav Item 3</li>
@endsection

@section('content')
    Welcome to the profile page!
@endsection
```

Redirect

```
to another URI: return Redirect::to('user/profile');
with a specific status: return Redirect::to('user/profile', 301);
to a secure URI: return Redirect::to_secure('user/profile');
to the root of your application: return Redirect::home();
back to the previous action: return Redirect::back();
to a named route: return Redirect::to_route('profile');
to a controller action: return Redirect::to_action('home@index');
to a named route with wildcard values:
return Redirect::to_route('profile', array($username));
to an action with wildcard values:
return Redirect::to_action('user@profile', array($username));
with error: return Response::error('404');
download: return Response::download('file/path.jpg', 'photo.jpg');

flash data:
return Redirect::to('profile')->with('status', 'Welcome Back!');
access message from the view: $status = Session::get('status');

assets: Asset::add('jquery', 'js/jquery.js');
registering an asset that has multiple dependencies:
Asset::add('jquery-ui', 'js/jquery-ui.js', array('first', 'second'));
Asset::container('footer')->add('example', 'js/example.js');
echo Asset::container('footer')->scripts();
Asset::container('foo')->bundle('admin');

pagination: $orders = DB::table('orders')->paginate($per_page);
pagination links: {{$orders->links(); }}
previous and next: {{$orders->previous().' '.$orders->next(); }}

HTML::link('user/profile', 'User Profile');
HTML::secure_link('user/profile', 'User Profile');
HTML::link('user/profile', 'User Profile', array('id' => 'profile_link'));
HTML::link_to_route('profile', array($username));
HTML::link_to_action('user@profile', array($username));
HTML::mailto('example@gmail.com', 'E-Mail Me!');
HTML::image('img/smile.jpg', $alt_text, array('id' => 'smile'));
HTML::ol(array('Get Peanut Butter', 'Get Chocolate', 'Feast'));

register macro: HTML::macro('my_element', function()
{   return '<article type="awesome">'; });
calling: {{HTML::my_element(); }}
```

Forms

```
secure form open: Form::open_secure('user/profile');
HTML attributes: Form::open('user/profile', 'POST', array('class' => 'awesome'));
file uploads: Form::open_for_files('users/profile');
file uploads and uses HTTPS: Form::open_secure_for_files('users/profile');
closing a form:  Form::close();

CSRF token:  Form::token();
Route::post('profile', array('before' => 'csrf', function() { .... });
retrieving token:  $token = Session::token();

Form::label('email', 'E-Mail Address', array('class' => 'awesome'));
Form::text('email', 'example@gmail.com');
Form::hidden('email', 'example@gmail.com');
Form::textarea('email', 'example@gmail.com');
Form::password('password');

Form::checkbox('name', 'value', true);
Form::select('size', array('L' => 'Large', 'S' => 'Small'), 'S');
Form::submit('Click Me!');
Form::macro('my_field', function()
{return '<input type="awesome">'; });
Form::my_field();
```

Input

```
get one field: $field = Input::get('field_name', 'Default_value');
get all: $input = Input::all();
set flash input: Input::flash('only', array('username', 'email'));
set flash input: Input::flash('except', array('password', 'credit_card'));
retrieving flash field: $name = Input::old('name');
redirect with fields: return Redirect::to('login')->with_input();
check if field exists: if (Input::has('name')) {...}
json format: Input::json();

$picture = Input::file('field_name');
specific field form file: Input::file('picture.size');
```

Cookies

```
cookie get with default value: Cookie::get('name', 'Default_value');
cookie for 60 minutes: Cookie::put('name', 'Fred', 60);
permanent cookie: Cookie::forever('name', 'Fred');
delete cookie: Cookie::forget('name');

merging new data into the current input:
Input::merge(array('name' => 'Spock'));
replacing the entire input array with new data:
Input::merge(array('doctor' => 'Bones', 'captain' => 'Kirk'));
```

Bundle

```
Bundle/start.php
Autoloader::namespaces(array(
    'Admin' => Bundle::path('admin').'models',   ));
Autoloader::map(array(
    'User'     => path('app').'models/user.php',
    'Contact' => path('app').'models/contact.php',
));
register bundle application/bundle.php
'admin' => array('location' => 'userscape/admin'
'autostart'=>true),
listen for a bundle's start event:
Event::listen('laravel.started: admin', function()
{   // The "admin" bundle has started... });
disabling a bundle: Bundle::disable('admin');

bundle view: return View::make('bundle::view');
bundle configuration item: return Config::get('bundle::file.option');
bundle language line: return Lang::line('bundle::file.line');
bundle exists: Bundle::exists('admin');
installation location of a bundle: $location = Bundle::path('admin');
configuration array for a bundle: $config = Bundle::get('admin');
names of all installed bundles: $names = Bundle::names();
publish bundle assets: php artisan bundle:publish
```

URI

```
current URI for the request: URI::current();
getting a specific segment from the URI: URI::segment(1);
default value if the segment doesn't exist: URI::segment(10, 'Foo');
full request URI, including query string: URI::full();
determine if the URI is "home": if (URI::is('home')) {... }
determine if the URI begins with "dcs/": if URI::is('dcs/*')) {... }
current request method: Request::method();
$_SERVER global array: Request::server('http_referer');
the requester's IP address: Request::ip();
if the current request is using HTTPS: if (Request::secure()) {... }
if the current request is an AJAX request: if (Request::ajax()){.. }
if the current request is via the Artisan CLI: if (Request::cli()) {.... }
```

URL

```
$url = URL::base();
URL relative to the base URL: URL::to('user/profile');
HTTPS URL: URL::to_secure('user/login');
retrieving the current URL: URL::current();
current URL including query string: URL::full();
URL to a named route: URL::to_route('profile');
URL to an asset: URL::to_asset('js/jquery.js');
URL to a named route with wildcard values: route('profile', array($username));
URL to a controller action: $url = action('user@profile');
URL to an action with wildcard values:
$url = action('user@profile', array($username));
```

Validation

```
$rules = array(     'name'  => 'required|max:50',
        'email' => 'required|email|unique:users', );
bind validation: $validation = Validator::make($input, $rules);
check: if ($validation->fails()) {... }

validations:
required, alpha, alpha_num, alpha_dash, size:5, between:5,10, min:5, max:5
numeric, integer, in:red,green,blue , not_in:pink,purple, confirmed, accepted
same:age, different:age, match:/[a-z]+/, unique:table_name

$validation->errors->get('email', '<p>:message</p>');
 all of the error messages: $validation->errors->all();

Route::get('register', function()
{
    return View::make('user.register');
});

Route::post('register', function()
{
    $rules = array(...);
    $validation = Validator::make(Input::all(), $rules);
    if ($validation->fails())
    {
        return Redirect::to('register')->with_errors($validation);
    }
});

$messages = array(
    'required' => 'The :attribute field is required.',
);

$validation = Validator::make(Input::get(), $rules, $messages);
```

Files

```
get file extension: File::extension('data.zip');
checking file type: if (File::is('jpg', 'path/to/file.jpg')) {.. }
getting content: $contents = File::get('path/to/file');
writing content: File::put('path/to/file', 'file contents');
file append: File::append('path/file.extension', $data);
removing file: File::delete('path/to/file');
file upload: Input::upload('picture', 'path/to/pictures', 'filename.ext');
copy directory: File::cpdir($directory, $destination);
removing directory: File::rmdir($directory);
```

Strings

```
character limit: echo Str::limit($string, 10);
words limit: echo Str::words($string, 10);
random string/ alpha: Str::random(32); Str::random(32,'alpha');
slugs: return Str::slug('My First Blog Post!', '_');
```

Localization

```
echo __('marketing.welcome');
with lang selection: echo Lang::line('marketing.welcome')->get('sp');
placeholders: 'welcome' => 'Welcome to our website, :name!'
echo __('marketing.welcome', array('name' => 'Taylor'));
```

Encryption

```
$encrypted = Crypter::encrypt($value);
$decrypted = Crypter::decrypt($encrypted);
```

Database

```
$users = DB::query('select * from users where name = ?', array('test'));
get first row: DB::first($sql), one value from column:  DB::only($sql)
get all: DB::table('users')->get();
get where id=$id: DB::table('users')->find($id);
get one column: DB::table('users')->where('id', '=', 1)->only('email');

conditions
where_in, where_not_in, or_where_in, and or_where_not_in
where_null, where_not_null, or_where_null, and or_where_not_null

return DB::table('users') ->where('id', '=', 1)
->or_where('email', '=', 'example@gmail.com')->first();

join example: DB::table('users')
    ->left_join('phone', 'users.id', '=', 'phone.user_id')
    ->get(array('users.email', 'phone.number'));
```

Eloquent

```
class User extends Eloquent {
    public static $table = 'my_users';
    public static $timestamps = true;  // automatic generate timestamps columns
    public static $accessible = array('email', 'password', 'name'); // mass assignment filter
}

get data: $user = User::where('email', '=', $email)->first();
User::where_email($email)->first();
User::where_in('id', array(1, 2, 3))->or_where('email', '=', $email)->get();
User::order_by('votes', 'desc')->take(10)->get();

$users = User::all();
foreach ($users as $user)
{   echo $user->email;  }

insert data: DB::table('users')->insert(array('email' => 'example@gmail.com'));
$id = DB::table('users')->insert_get_id(array('email' => 'example@gmail.com'));

$user = new User;
$user->email = 'example@gmail.com';
$user->password = 'secret';
$user->save();

update data: $affected = DB::table('users')->where('id', '=', 1)
                ->update(array('email' => 'new_email@gmail.com'));

$user = User::find(1);
$user->email = 'new_email@gmail.com';
$user->password = 'new_secret';
$user->save();

delete records: DB::table('users')->where('id', '=', 1)->delete();
DB::table('users')->delete(1);
```

relations

one-to-one

```
class User extends Eloquent {
    public function phone()
    {
        return $this->has_one('Phone'); // with user_id foreign key
        return $this->has_one('Phone', 'my_foreign_key'); // with my_foreign_id foreign key
    }
}

class Phone extends Eloquent {
    public function user()
    {
        return $this->belongs_to('User');
    }
}

$phone = User::find(1)->phone()->first();
$phone = User::find(1)->phone;
```

one-to-many

```
class Post extends Eloquent {
    public function comments()
    {
        return $this->has_many('Comment'); // with post_id foreign key
        return $this->has_many('Comment', 'my_foreign_key');
    }
}
$comments = Post::find(1)->comments()->get();
$comments = Post::find(1)->comments;
echo Post::find(1)->comments()->order_by('votes', 'desc')->take(10)->get();
```

many-to-many

```
class User extends Eloquent {
    public function roles()
    {
        return $this->has_many_and_belongs_to('Role');
    }
}

class User extends Eloquent {
    public function roles()
    {
        return $this->has_many_and_belongs_to('Role', 'user_roles');
    }
}

$comments = array(
   array('message' => 'A new comment.'),
   array('message' => 'A second comment.'),
);
$post = Post::find(1);
$post->comments()->save($comments);

$role = new Role(array('title' => 'Admin'));
$user = User::find(1);
$user->roles()->insert($role);
```

intermediate table many-to-many

```
$user = User::find(1);
$pivot = $user->roles()->pivot();

remove date from intermedite table
$user = User::find(1);
$user->roles()->delete();
```

getter and setter

```
public function set_password($password)
{    $this->set_attribute('hashed_password', Hash::make($password));  }
$this->password = "my new password";

public function get_published_date()
{    return date('M j, Y', $this->get_attribute('published_at')); }
echo $this->published_date;
```

mass assignment

```
$user = new User;
$user->fill(array( 'username' => 'first last', 'password' => 'disgaea'));
$user->save();
```