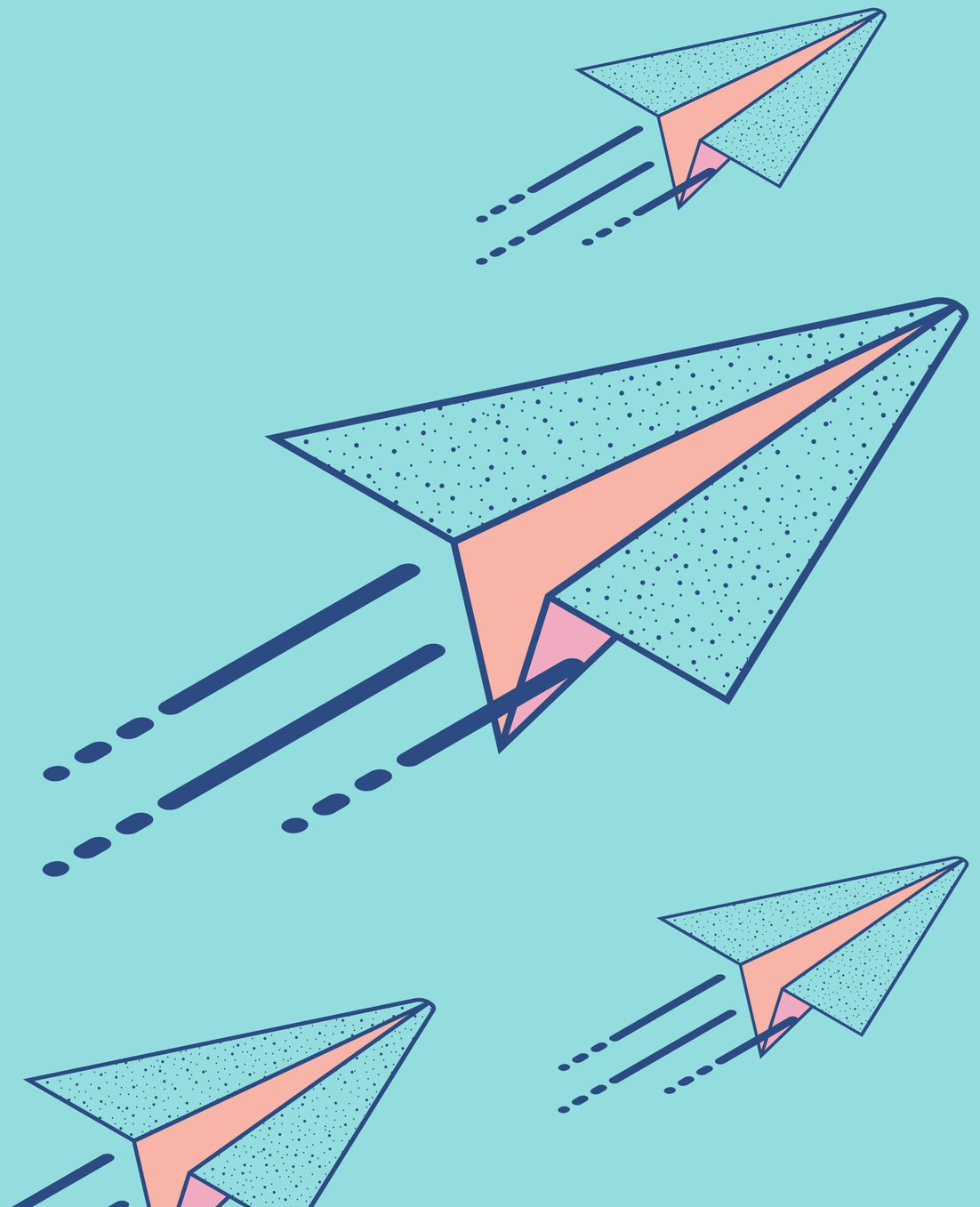
SETEC
វិទ្យាស្ថានស៊ីតិក
SETEC INSTITUTE

Laravel-Routing

By TRY VORN



ROUTING

Routing in Laravel includes the following categories –

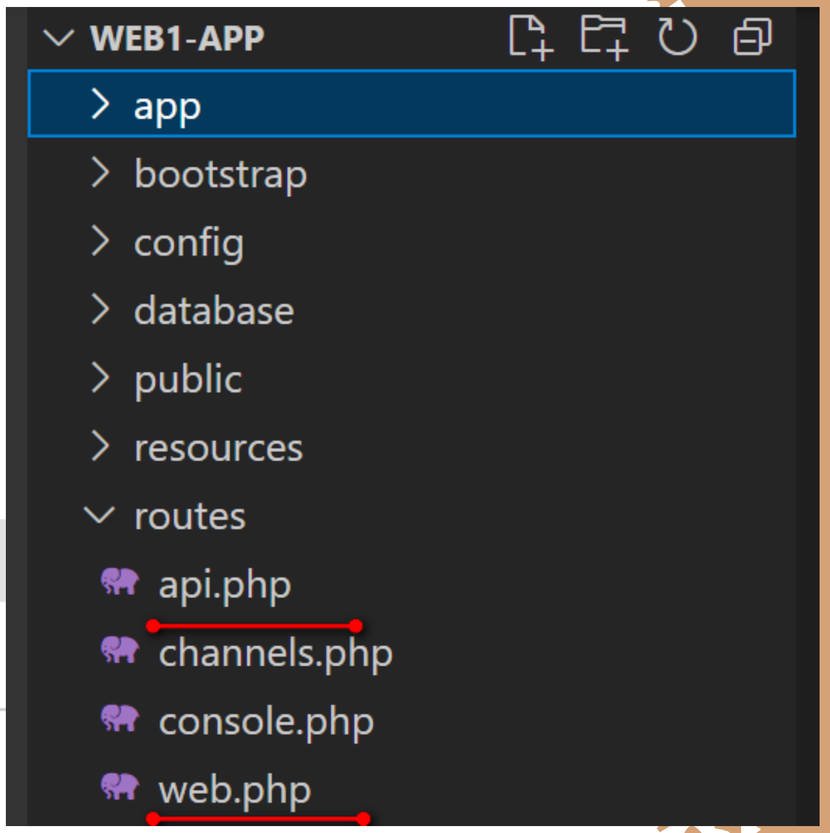
- Basic Routing
- Redirect routes
- View routes
- Route parameters
- Named Routes

BASIC ROUTING

All the application routes are registered within the **app/routes.ph**p file.

- the routes/web.php file defines routes that are for your web interface
- the routes/api.php file is stateless and is assigned the api middleware group

```
Route::get('/sample', function () {
    return 'Basic routing in laravel.';
});
```



AVAILABLE ROUTER METHODS

The best way to explain is to show how REST API works. Just imagine a **pet**

- Route::get('$uri, $callback) ==> GET/pet - get a pet
- Route::post('$uri, $callback) ==> POST/pet - store a pet
- Route::put('$uri, $callback) ==> PUST/pet - replace a pet
- Route::patch('$uri, $callback) ==> PATCH/pet - modify pet partially
- Route::delete('$uri, $callback) ==> DELETE/pet - delete a pet
- Route::options('$uri, $callback)

- Route::get('$uri, $callback) ==> GET/pets - get list pets
- Route::post('$uri, $callback) ==> POST/pets - post many pets
- Route::delete('$uri, $callback) ==> DELETE/pets - delete all pets

MATCH METHOD

Sometimes you may need to register a route that responds to multiple HTTP verbs. You may do so using the match method. Or, you may even register a route that responds to all HTTP verbs using the any method:

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

Route::any('/',function(){
 //
});
```

REDIRECT ROUTE

If you are definding a route that redirects to another URI, you may use the Route::redirect method.

```
Route::Redirect("/here","/there");

Route::Redirect("/here","/there",301);
```

VIEW ROUTE

If your route only needs to return a view, you may use the **Route::view** method

```
Route::view("/welcome1","welcome");//welcome.blade.php

Route::view("/welcome2","welcome2",["name" =>"Vorn"]);//welcome2.blade.php
```

ROUTE PARAMETERS

The route parameters are enclosed within {} brackets, and parameters must contain alphabetic characters. It should not contain '-' character, and instead of using this character, you can use '_' character.

```
Route::get('/post/{id}', function($id)  {
  return "id number is : ". $id;
}  );

//We can also pass the multiple parameters.
Route::get('/post/{id}/{name}', function($id,$name)  {
  return "id number is : ". $id ." ".$name;
}  );
```

OPTIONAL PARAMETERS

To make the route parameter optional, you can place '?' operator after the parameter name. If you want to provide the optional parameter, and then make sure that you have also provided the default value to the variable.

Example 1:

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

Example 2:

```
Route::get('user/{name?}', function ($name = 'Vorn') {  // Vorn is default value
    return $name;
});
```

REGULAR EXPRESSION CONSTRAINTS

These are the constraints that can format the route parameters by using the where method on a route instance. The 'where' method accepts the name of the parameter and regular expression constraint that defines how the parameter should be constrained.

Suppose we want to pass the user name as a route parameter that contains only **alphabetical characters.**

```
Route::get('user/{name?}', function ($name=null) {
    return $name;
})->where('name','[a-zA-Z]+');
```

Let's consider an example that accepts only **numeric** values.

```
Route::get('user/{id?}', function ($id=null) {
    return "id is : ". $id;
})->where('id','[0-9]+');
```

Let's consider an example that accepts **alphanumeric** characters.

```
Route::get('user/{id}/{name}', function ($id,$name) {
    return "id is : ". $id ." ,".  "Name is : ".$name ;
})->where(['id'=>'[0-9]+', 'name'=>'[a-zA-Z]+']);
```