ITI Open-source
Day 05

Introduction

Content

- Request
- Response
- Session
- Restful API
- Form request

Rest API: intro

Rest acronym for Representational State transition, software architectural design uses **HTTP** protocol and **HTTP Verbs (**GET, POST, PUT, DELETE, …**)**, and HTTP **response code (**200 ok, 201 created, ..etc**)**

GET = get date from server
POST = create new resource on server
PUT/Patch = update data on server
DELETE = delete resource from server

Resource	HTTP Verb	URI	Description
Task	GET	/api/v1/tasks	Get all tasks
Task	GET	/api/v1/tasks/1	Get task with id 1
Task	POST	/api/v1/tasks	Create new task
Task	PUT/Patch	/api/v1/tasks/1	Update task
Task	DELETE	/api/v1/tasks/1	Delete task with id 1

Rest API: Practices

1. Use HTTP Verb (GET, POST, PUT, Delete …etc.)
2. Use API Versioning “v1”
3. Use plurals to describe resources
4. Use Response Codes and Error Handling “200, 201, 400”
5. Use well structure Json as default

@PRACTICE:

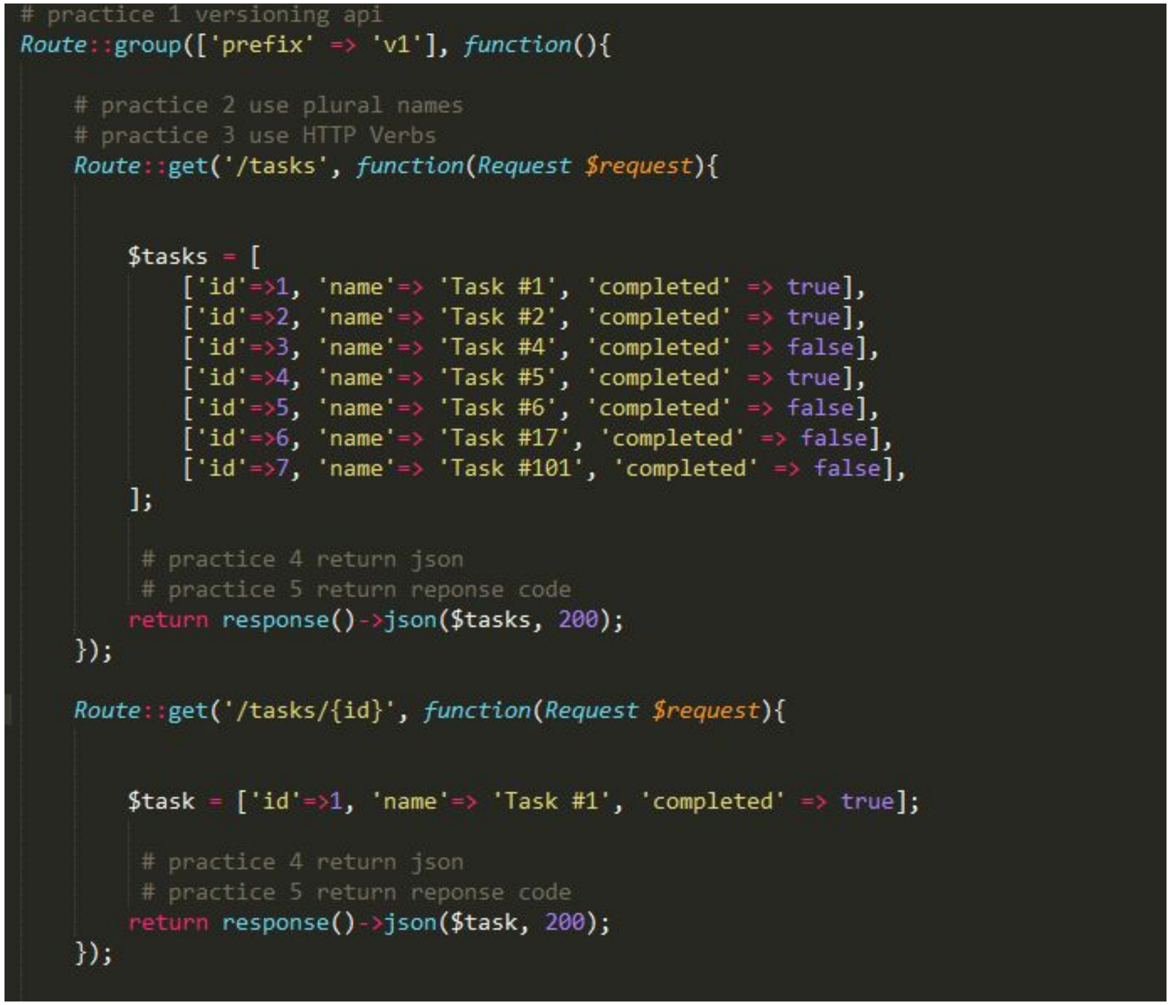
https://jsonplaceholder.typicode.com/posts
https://github.com/typicode/jsonplaceholder#how-to

@TODO:

https://en.wikipedia.org/wiki/List_of_HTTP_status_codes
https://dev.twitter.com/rest/public

Laravel Rest API

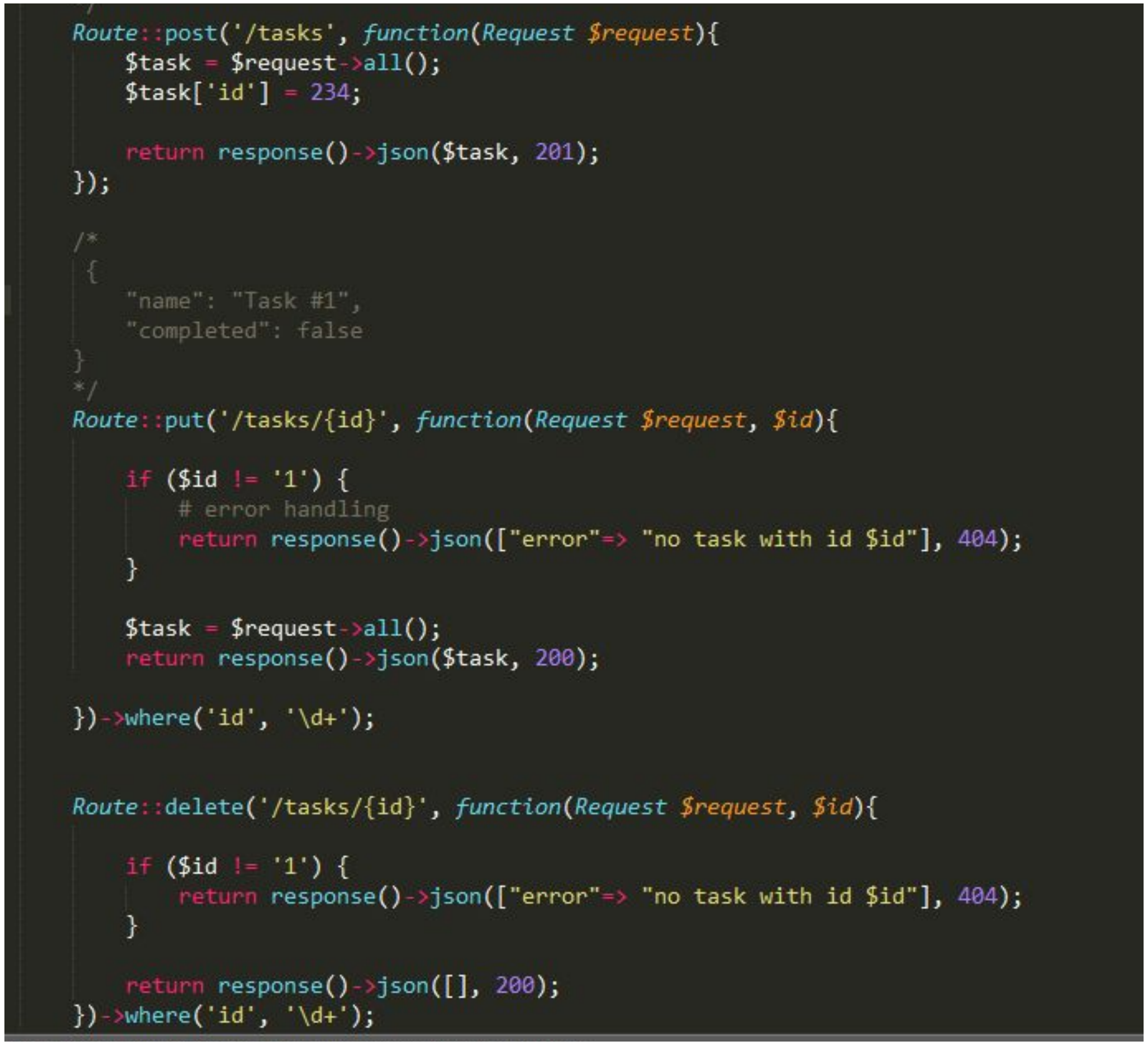
Simply with laravel we apply the prectices of RestAPI
/routes/api.php



```php
# practice 1 versioning api
Route::group(['prefix' => 'v1'], function(){

    # practice 2 use plural names
    # practice 3 use HTTP Verbs
    Route::get('/tasks', function(Request $request){


        $tasks = [
            ['id'=>1, 'name'=> 'Task #1', 'completed' => true],
            ['id'=>2, 'name'=> 'Task #2', 'completed' => true],
            ['id'=>3, 'name'=> 'Task #4', 'completed' => false],
            ['id'=>4, 'name'=> 'Task #5', 'completed' => true],
            ['id'=>5, 'name'=> 'Task #6', 'completed' => false],
            ['id'=>6, 'name'=> 'Task #17', 'completed' => false],
            ['id'=>7, 'name'=> 'Task #101', 'completed' => false],
        ];

         # practice 4 return json
         # practice 5 return reponse code
        return response()->json($tasks, 200);
    });

    Route::get('/tasks/{id}', function(Request $request){


        $task = ['id'=>1, 'name'=> 'Task #1', 'completed' => true];

         # practice 4 return json
         # practice 5 return reponse code
        return response()->json($task, 200);
    });
```

Laravel Rest API

/routes/api.php



```php
Route::post('/tasks', function(Request $request){
    $task = $request->all();
    $task['id'] = 234;

    return response()->json($task, 201);
});

/*
 {
    "name": "Task #1",
    "completed": false
}
*/
Route::put('/tasks/{id}', function(Request $request, $id){

    if ($id != '1') {
        # error handling
        return response()->json(["error"=> "no task with id $id"], 404);
    }

    $task = $request->all();
    return response()->json($task, 200);

})->where('id', '\d+');


Route::delete('/tasks/{id}', function(Request $request, $id){

    if ($id != '1') {
        return response()->json(["error"=> "no task with id $id"], 404);
    }

    return response()->json([], 200);
})->where('id', '\d+');
```

Day 5: Lab

Lab 5: Task Tracker

- Cont. Lab 4
    - Implement API
    - Middleware to check the register

Note

You need

1. Complete @TODO points
2. Visit @MANDATORY Laravel documentation for each part

HOME
NEW
VIEW
Task Tracker