**LARAVEL\_LUMEN (REST API)**

**Definition:**

**REST: Representational Source Transfer.**

**API: Application Programming interface.**

**WHY: To use single database in various platforms IOT (internet of things) Devices. Transfer or use data from one software to another software.**

**Properties:**

**Client-server: Works Between Client & SERVER.**

**Stateless: Never Store any Data inside API Script.**

**Cacheable: Device Can Store Cache but API not.**

**Uniform interface: Various methods can operate from single URI.**

**Tools for Start Working:**

* **xampp**: For Apache & MySQL;
* **composer**: Php Package Manager**;**
* **Sequence**: First XAMPP then composer, composer depends on XAMPP php.
* **Postman**: **For API Practice.**

**Installation Command:**

* **install\_command**: **composer create-project --prefer-dist laravel/lumen** blog
* **Development server**: **php -S localhost:8000 -t public**

|  |  |
| --- | --- |
| METHOD OF REST API | |
| $route->get($uri, $callback) | To view Data |
| $route->post($uri, $callback) | To insert Data |
| $route->put($uri, $callback) | To update Data |
| $route->delete($uri, $callback) | To delete Data |

**Route Parameters:**

* **Required**: $router->get('/**{name}/{age}**', function($name,$age){});
* **Optional**: $router->get('/**[{name}]**',function(**$name=null**){});

**Make Controller Ready:**

**Create: Create New Controller.**

**use: use App\User;**

**Response Area:**

**Header: Confidential Data pass and receive.**

**Param: Single data pass & receive.**

**Body: Average Data pass and receive.**

**Response Type:**

* **Simple String Response.**
* **JSON Response. \***
* **XML Response.**
* **Download. \***
* **Redirect. \***

**Simple String Response:**

**Body: response($value) Can receive only value.**

**Header: ->header(‘key’, ‘value’) Need Two Parameter.**

**Syntax:**

**public function ApiCheck($name){**

**return response($name)**

**->header('name',$name)**

**->header('email','n@gmail.com');**

**}**

**Json Response:**

**Body: Json Object array will generate inside body.**

**Header: json Response will not work inside header.**

**Syntax:**

**public function ApiCheck($name){**

**$ associative\_array =[‘key’=> ‘value’];**

**return response()->json($associative\_array);**

**}**

**Redirect Response:**

**Set Route:**

* **$router-> get('/First','ApiController@First');**
* **$router-> get('/Second','ApiController@Second');**

**Redirect Syntax:**

**public function First(){**

**return redirect('/Second'); }**

**public function Second(){**

**return "API Developer"; }**

**Download Response:**

**File Path: Keep File inside index folder;**

**Syntax:**

**public function Download(){**

**$path = 'coder.png';**

**return response()->download($path);**

**}**

**DATA SENDING & CATCHING:**

**Process: Parameter(visible), Header(hide), Body\_Json.**

**Required Class: use Illuminate\Http\Request**

**catch Only Body & Parameter Data:**

**public function \_\_invoke(Request $req){**

**return $req;}**

**Catch Only Header Data:**

**public function \_\_invoke(Request $req){**

**return $req->header('name');}**

**//Specify Name to get the specified Data.**

**Database CRUD Operation:**

**Requirement: use Illuminate\Support\Facades\DB**

**Comment Out: Bootstrap > app [ $app->withFacades() ]**

**Receive JSON BODY DATA:**

**$request: Ensure Only body or param data received.**

**Input(): Slice json object properties & Receive Param data;**

**function \_\_invoke(Request $request){**

**$name= $request -> input('name');**

**$email= $request -> input('email');**

**$cell= $request -> input('cell');**

**$address= $request -> input('address'); }**

**Laravel Lumen Generator:**

**Install: composer require flipbox/lumen-generator**

**Why: To create files using cmd.**

**Configure: Add in Bootstrap/app.php file:**

**$app->register(Flipbox\LumenGenerator\**

**LumenGeneratorServiceProvider::class);**

**Lumen Authentication**

**Authentication Steps:**

* **Client Server: Sent Request To authentication Server.**
* **Auth Server: Validate then sent access token to client.**
* **Client Server: Sent Request To resource Server with Access token.**
* **Resource Server: Validate access token to Auth server.**
* **Resource Server: Finally, Response to Client’s Request.**

**Authentication Architecture:**

* **Uncommand Bootstrap/App: Authenticate middleware, AuthServiceProvider.**
* **App/Provider/AuthServiceProvider: method boot () {Access token will go inside boot method then method will check that key, if authorized or not value will transfer to middleware/ Authenticate.php}**
* **App/Middleware/Authenticate: method handle () {This method will receive boot() Logic value & declare if the authentication is success allow to go next step (Create App/User.php Class Object) or authentication is false return unauthorized 403.**
* **Authenticate Route Example:**

**$router->get('/',['middleware'=>'auth','uses'=>'AuthControler']);**

* **Middleware => ‘auth’ detect authentication success or not.**
* **If success, then uses call controller.**

**JWT Authentication:**

* **JWT: JSON WEB TOKEN.**
* **Why: JWT is a standard process to transfer json data or secure Json Object transmitting system. Encrypt & decrypt json object for security purpose is the main goal of JWT.**
* **Key: Encrypt & decrypt signed with public/private a key.**
* **Uses:** 
  + **Authentication Purpose.**
  + **Secure Data Transfer.**

**JWT Structure 3 part:**

* **Header:** 
  + **Algorithm.**
  + **Type.**
* **Payload (Main Data):** 
  + **Subject.**
  + **Name.**
  + **May be a lot of Properties.**
* **Signature:** 
  + **Encryption type or Encoding type.**
* **Token:** 
  + **Finally Created a token with a key.**
  + **Must have to use that key to decode.**

**JWT Resource:**

* **ENV (Environmental variable):** 
  + **SET.env: KEY=value**
  + **USE.env**: env(‘key’)
  + **Why**: Set important constant in.env
  + **Set JWT token key inside.env**
* **Website:** 
  + **Website: JWT.io**
  + **Version: created by firebase company**
  + **Command: composer require firebase/php-jwt**
  + **Why: To use JWT Build in functions.**
  + **"iss" (Issuer)**
  + **"sub" (Subject)**
  + **"aud" (Audience)**
  + **"exp" (Expiration Time)**
  + **"nbf" (Not Before)**
  + **"iat" (Issued At)**
  + **"jti" (JWT ID)**
* **Basic Syntax:**

**use \Firebase\JWT\JWT;**

**$key = "example\_key";**

**$payload = [**

**"iss" => "https://developer.com",**

**"sub" => "login Information",**

**"iat" => time();**

**"exp" => time()+3600;**

**$jwt = JWT::encode($payload, $key);**

**$decoded = JWT::decode($jwt, $key, array('HS256'));**