الجمهورية العربية السورية
جامعة دمشق
كلية الهندسة المعلوماتية

Programming Languages

Lecture Four

PHP Laravel – Request Methods

Prepared By

Eng. Rawan Koroni

Eng. Ahmed Al-Ahmed

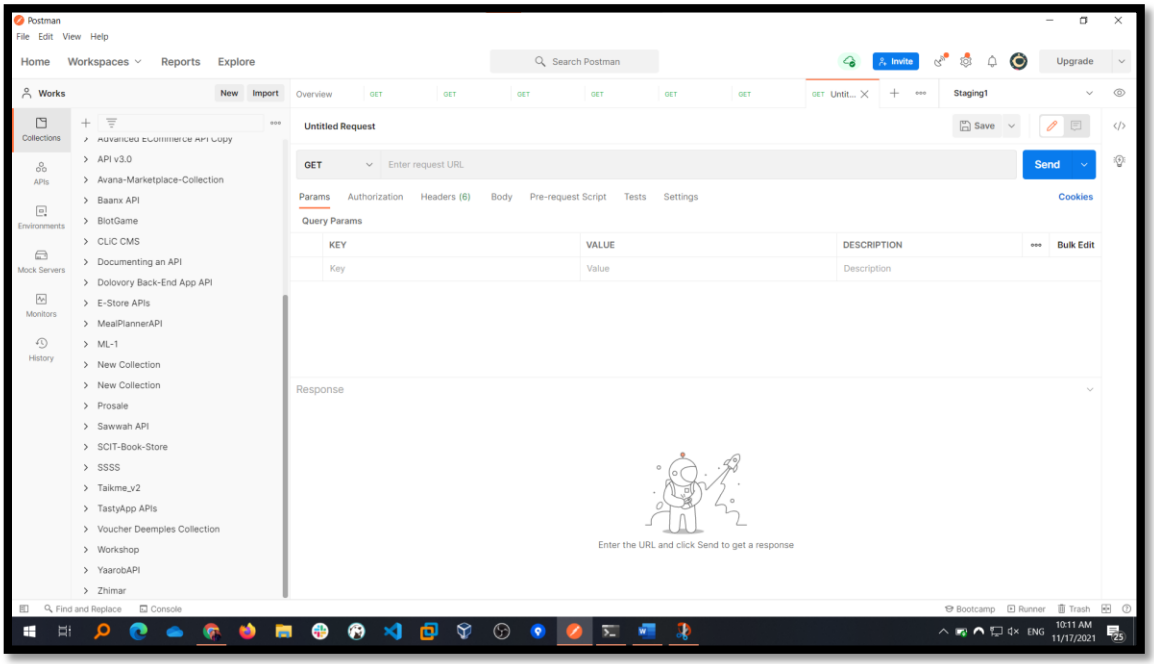
1- Introduction

- In this lecture, we will talk about request methods, how to use it, and when?
- Every HTTP request, which is sent from source A to server B, should have a method, and this method decides the expected action from the server B.
- There are many HTTP request methods, and the most commonly used are: GET, POST, and DELETE.
- GET: is commonly used to retrieve data without doing any persistent action on DB, SERVER.
- POST: is commonly used to create a new entry in the database, server … etc.
- DELETE: is a type of POST request, but it is commonly used to delete an entry from DB, SERVER … etc.
- For Example: If we have a simple system for creating products/ get last orders, and delete invalid orders, we will split these functions/end-points into the following:
    1. GET -> Retrieve last orders.
    2. POST -> Create products.
    3. DELETE -> Delete invalid orders.
- Laravel supports a great way to handle this kind of request methods just like the code bellow:

- We called these end-points RESTFull API.
- When we access any website from our browsers like Firefox, Chrome, Edge, … the browser by default sends GET request to the server for the requested page/path and the server responds with the result.

2- POSTMAN

- You cannot make a POST or DELETE request by using a web browser, as web browsers only directly support GET requests.
- One of the most common tools used as an API Client is POSTMAN.
- You can download POSTMAN using the following link: https://www.postman.com/downloads/
- After downloading POSTMAN and installing it, you can see the following screen:



- As we see, we can select the request method, API URL and parameters, body payload, …etc.

3- Example

- We will create a simple example for managing products [as json file].
- We will create three APIs for add, delete and get products.
- Let's first create a new Laravel project.
- We will create an empty json file in (C:\xampp\htdocs), and let's name it products_list.json file.
- Let's create a new controller called ProductController and add 3 methods as following:

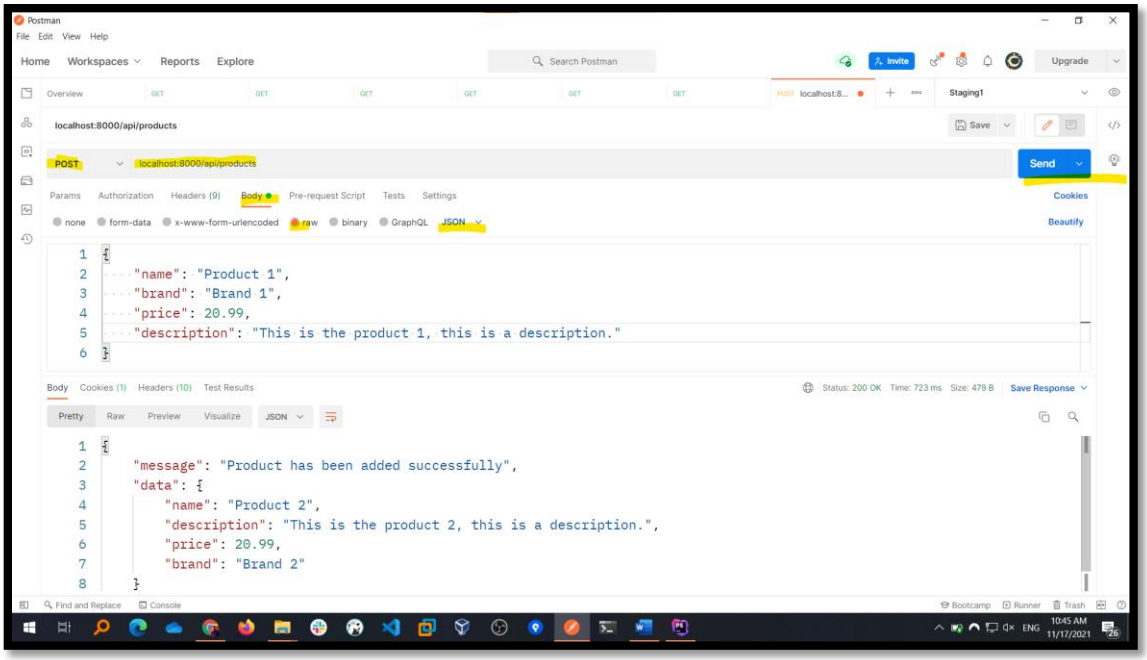
```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller
{
    public function createProduct(Request $request)
    {
        // here the code to create the product.
    }

    public function deleteProductById($productId)
    {
        // delete product by id.
    }

    public function listAllProducts()
    {
        // return a list of all products.
    }
}
```

- Let’s first implement createProduct function, at the beginning we need to get the product info from the body payload since in POST request we commonly send the data through the body.
- We will provide name, description, price and brand for each product.
- We need to modify api.php as following:

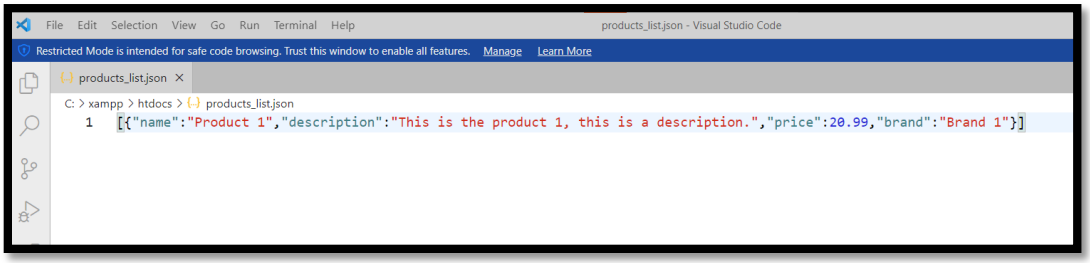
- And the method will be:

```php
public function createProduct(Request $request)
    {
        // Get info from the body payload.
        $name = $request->input('name');
        $description = $request->input('description');
        $price = $request->input('price');
        $brand = $request->input('brand');

        if (!$name || !$description || !$price || !$brand) {
            return response()->json(['message' => 'Invalid payload, all fields are required.',
'data' => null], 400);
        }

        $filePath = 'C:\xampp\htdocs\products_list.json';
        $fileContent = file_get_contents($filePath);
        $jsonContent = json_decode($fileContent, true);
        $payload = [
            'name' => $name,
            'description' => $description,
            'price' => $price,
            'brand' => $brand,
        ];

        if (!$jsonContent || !is_array($jsonContent)) {
            $content = [
                $payload
            ];
           file_put_contents($filePath, json_encode($content));

        } else {
            $jsonContent[] = $payload;
            file_put_contents($filePath, json_encode($jsonContent));
        }

        return response()->json(['message' => 'Product has been added successfully', 'data' =>
$payload]);
    }
```

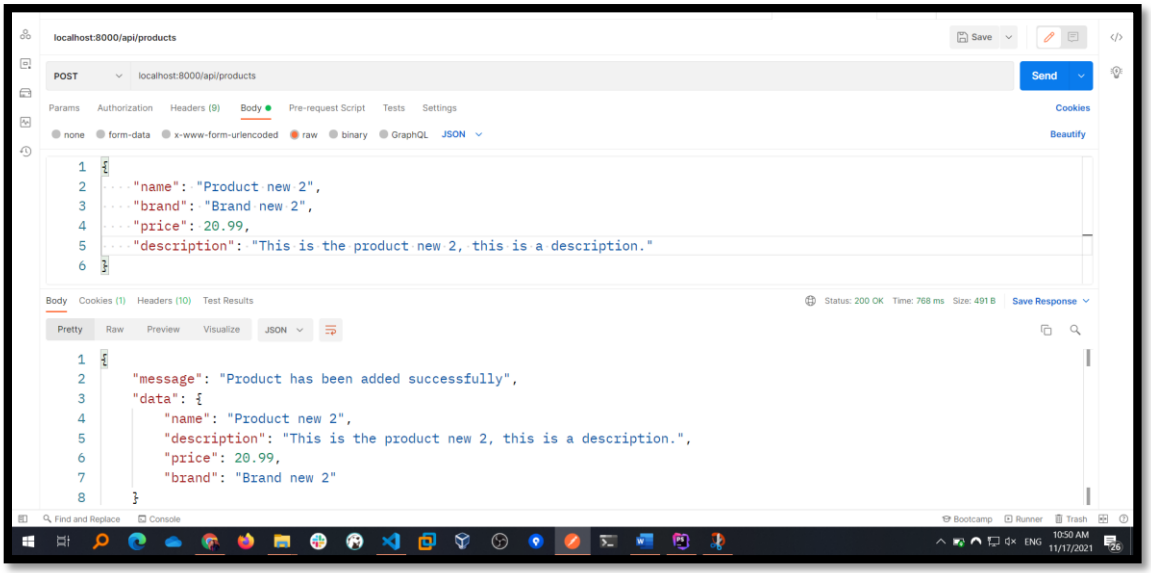
- Let’s test it with POSTMAN as the following:



- As the image describes we should change the request method from GET to POST, provide the URL/end-point [note that when we use api.php file we should pass api as a prefix for all end-points], then provide the payload as a JSON raw content and finally hit Send button.
- After sending the request, we can check the file content, and it will be like this:



- If we changed the payload content and re-send it:



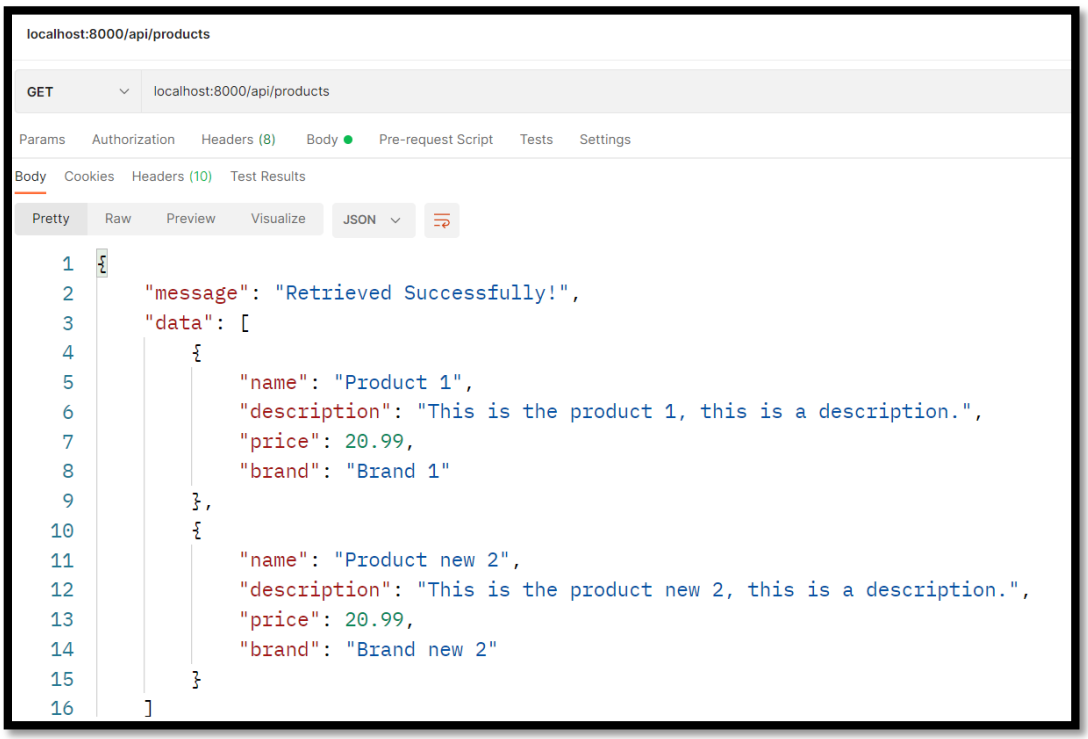
- The file content will be:

- Let's implement the GET API, which retrieve the whole content of the file.
- The router file [api.php] will looks like:

- And the controller function will be:

```php
public function listAllProducts()
{
    $filePath = 'C:\xampp\htdocs\products_list.json';
    $fileContent = file_get_contents($filePath);
    $jsonContent = json_decode($fileContent, true);
    return response()->json([
       'message' => 'Retrieved Successfully!',
       'data' => $jsonContent,
    ]);
}
```

- If we tested it with POSTMAN we will get the result like this:



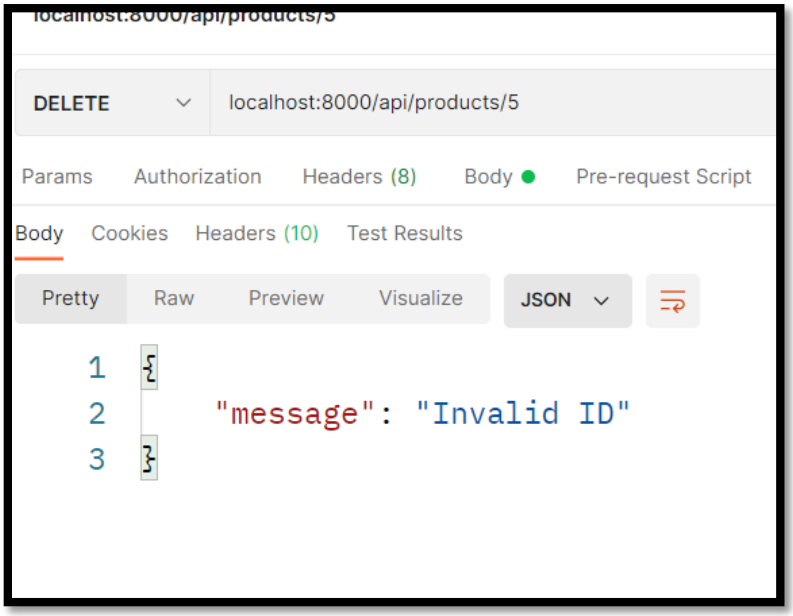
- Finally, let's handle the last request method (DELETE) which delete one entry from the products JSON file.
- We will delete the product which matches the given index/sort number.
- Let's add new end-point for the router file.

```php
<?php

use Illuminate\Support\Facades\Route;

Route::post('products', 'ProductController@createProduct');
Route::get('products', 'ProductController@listAllProducts');
Route::delete('products/{productId}', 'ProductController@deleteProductById');
```

- And the controller function:

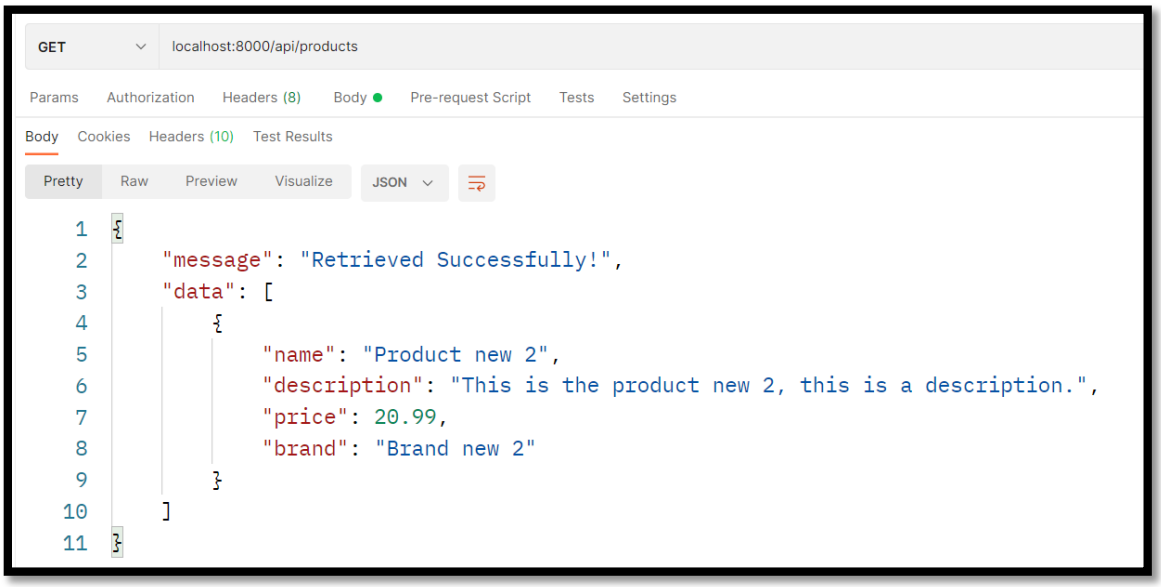
```php
public function deleteProductById($productId)
  {
      $filePath = 'C:\xampp\htdocs\products_list.json';
      $fileContent = file_get_contents($filePath);
      $jsonContent = json_decode($fileContent, true);
      if ($productId < 0 || $productId > count($jsonContent)) {
          return response()->json(['message' => 'Invalid ID'], 400);
      }

      unset($jsonContent[$productId-1]);
      file_put_contents($filePath, json_encode(array_values($jsonContent)));
      return  response()->json(['message' => 'Product has been deleted successfully']);

  }
```

- And if we tested it in POSTMAN:



- It will throw an error, since we don't have 5 items, so let's try to delete the first product.

- And if we tried to get the products again, we will retrieve only one product:



4- Exercise

- Add a new API to the previous example, which update the name of an existing product by getting “product index number” and the “new name” as parameters, and replace the product old name with the new one.
- Note that the new API should return an error if it could not find the product by its number.