الجمهورية العربية السورية
جامعة دمشق
كلية الهندسة المعلوماتية

Programming Languages

Lecture Three

PHP Laravel – Basics

Prepared By

Eng. Rawan Koroni

Eng. Ahmed Al-Ahmed

1- What is Laravel Framework?

- Laravel is an open-source PHP framework, which is robust and easy to understand
- It follows a model-view-controller design pattern.
- It consists of many folders, and each one contains some files which are responsible for a very specific actions/behavior.

2- Laravel Framework Folders

The most import folders (for beginners) are:

1. **App**: contains controllers, models, providers, notifications, jobs, events, exception handlers, and commands.
2. **Config**: contains all php pure configuration files, each file just return an array of keys to configure the whole application like the encryption key, hashing algorithms and storage folders.
3. **Database**: contains seeders, factories, and migration files.
4. **Resources:** contains the HTML content for emails, views and their assets like JS, CSS …etc.
5. **Routes:** contains the routing files which direct each request to the appropriate controller method.
6. **Storage:** contains caches and the files, which are stored locally.
7. **Vendor:** this folder is auto-generated and we **should not update it manually.**
8. **Tests:** contains the unit/feature testing codes.

3- Laravel Framework - MVC

- **Controller** is the class that receive a request from the user, call the appropriate function/method from the model (the database) and pass the result to the view component. It should not handle any logic except validating the request if it contains all the required parameters.
- **Model** is the class that has the ability to call database queries, for example : ***getUserByEmailAddress.***
- **View** contains HTML pages, for example if we want to create a login form, we put the page here and pass the data through it.

4- Laravel Framework - Routes

If we checked the file ***routes/web.php***, we can see something like:

```
Route::get('/', 'WebsiteController@index');
Route::get('/home', 'WebsiteController@index');
Route::get('blog', 'BlogController@index');
```

This means that if we sent a GET request to http://website.com/ , the request will be handled by the function index within ***WebsiteController*** class, and the same if we requested ***/home*** or ***/blog***.

5- Laravel Framework - Controller

Controller may have one or more handling methods for example:

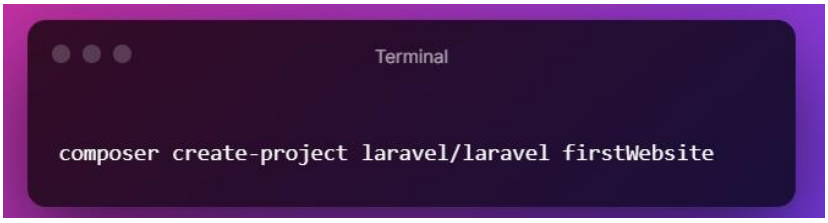
```
9
0   class WebsiteController extends Controller
1   {
2       public function index(Request $request)
3       {
4           $categories = Category::with(['last_posts'])->get();
5
6           return view('layouts.index', compact(
7               'categories'
8           ));
9       }
```

For example, in line 4 we can see that we get categories and render the view file ***resources/views/layouts.index.blade.php***. Where the view function assume that all the files are exist in ***resources/views*** folder and automatically add ***extension.blade.php*** to each file we use. The compact function in PHP is simply pass variable(s) to the view.

6- Laravel Framework – simple web service

In this section, we will implement a simple web-service, which accepts one parameter like name, and return a welcome message:

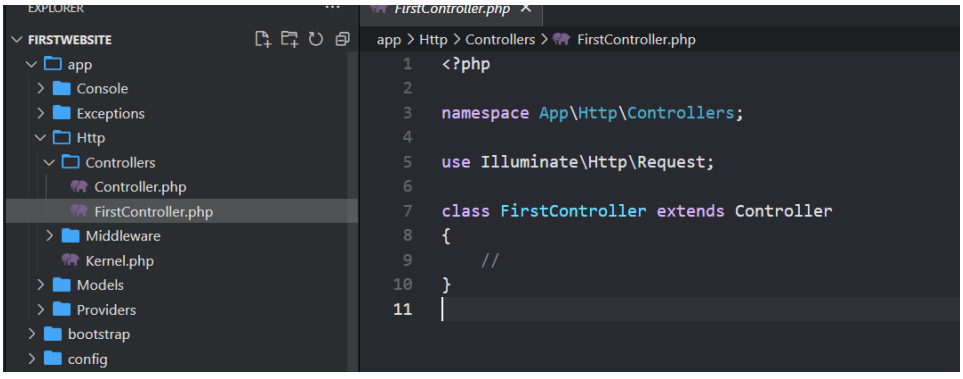
- First of all, create a new Laravel project (we can use the one we created in the first lecture) by running the follow command:



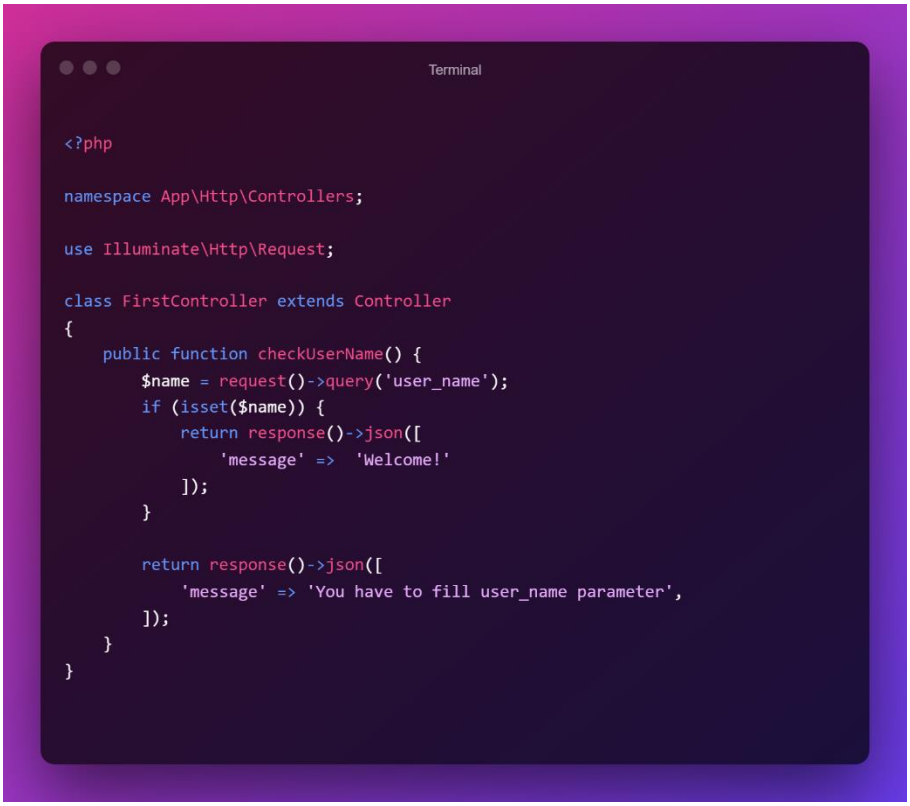
We can open it with ***VSCode*** or any text editor/IDE we want.

- Run **XAMPP Apache server**.
- Let's go to the project directory and run this command in Terminal:

this command will create a new controller class within controllers folder, this controller will looks like:

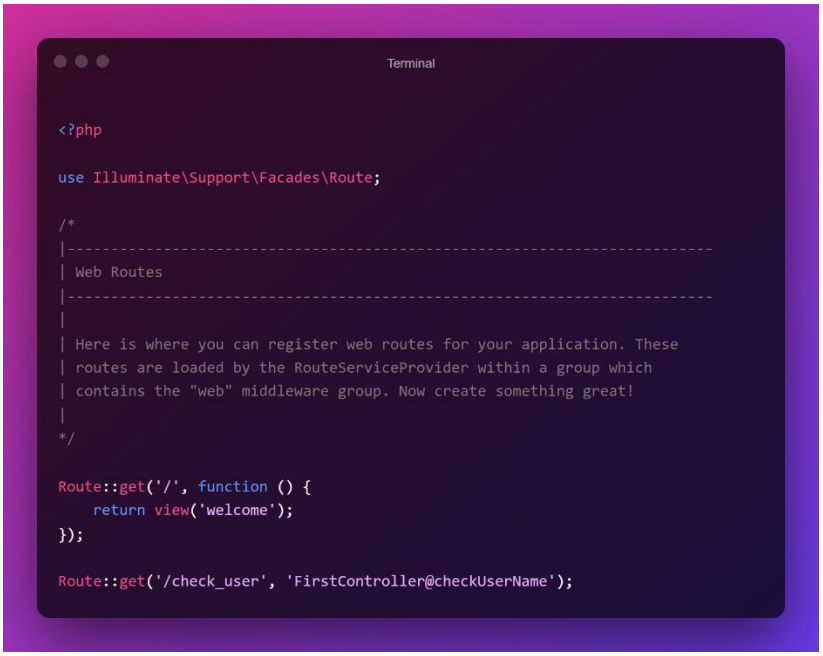


- Add a new method function with the following simple code:



We created a function called **checkUserName,** and get a query parameter from the *URL* called ***user_name***. After that, we checked if there is a value/parameter, we returned a *json* response to the user with a welcome message, otherwise we returned an error message.

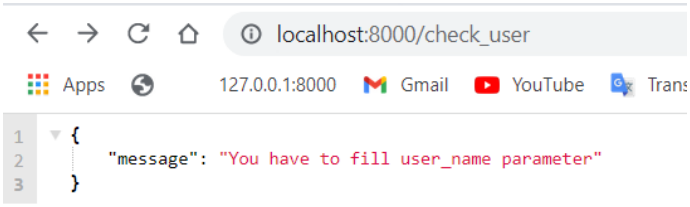
- Add a route/end-point to point to this function, so when the user hit that end-point, the application will redirect the whole traffic to that function.
  we need to edit ***routes/web.php*** to be like this:



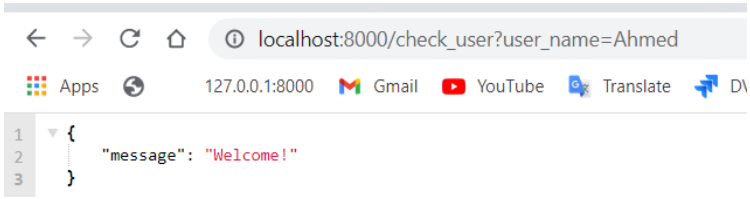
  In the above image we specified that we have an end-point called ***/check_user*** and this end-point will be handled by the controller ***FirstController*** and within the method ***checkUserName*** (there are many ways to specify the controller method, but this is the most commonly one).

- Now we need to update/override the namespace in the class: ***App/Providers/RouteServiceProvider,*** just uncomment the line number 29 **protected $namespace = 'App\\Http\\Controllers';**

- Now run the app with the command: **php artisan serve**
- Try to access that page via browser like this, and you will see the following message:



- Now, let's add the ***user_name*** parameter to the url:



7- Laravel Framework – web service with json file

We will now read a JSON file, saves its values in a variable, and check:

If the given name in the parameter ***user_name*** is exists in the file, then we will return a success message, otherwise we will return an error.

- Let’s create a JSON file within the main directory with the following content: (file path: **C:\xampp\htdocs\json_file.json**)

Terminal

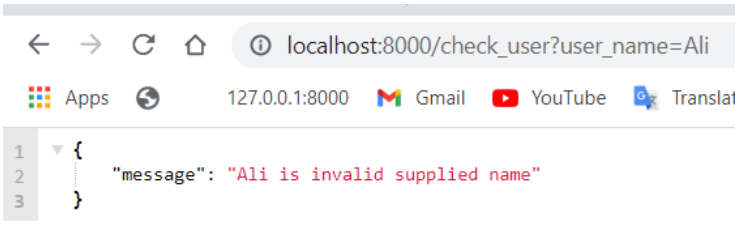
```
[
  "Ahmed",
  "Rawan",
  "Rimon"
]
```

- Let’s update our function to be like this:

Terminal

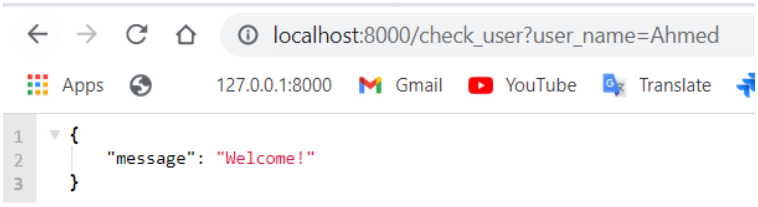
```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class FirstController extends Controller
{
    public function checkUserName() {

        $name = request()->query('user_name');
        if (!isset($name)) {
            return response()->json([
                'message' => 'You have to fill user_name parameter',
            ]);
        }

        $fileContent = file_get_contents('http://localhost:8080/json_file.json');
        $jsonContent = json_decode($fileContent, true);
        if (!in_array($name, $jsonContent)) {
            return response()->json([
                'message' => sprintf('%s is invalid supplied name', $name)
            ]);
        }

        return response()->json([
            'message' =>  'Welcome!'
        ]);
    }
}
```

- We used ***file_get_content*** function to read a file and we supplied its location via a HTTP request not locally.
- The response of that function will be ***string*** so we will decode that response as a JSON format.
- We checked if that name is not within the names array ***$jsonContent,*** then we raised an error response; otherwise we returned a welcome message.
- Now if we requested that end-point like this:



We will get an error since we do not have ***Ali*** in our JSON file.

- But if we changed ***Ali*** to ***Ahmed*** for example, we get a success response like:



8- Conclusion

In this lecture, we could know:

- How to create a web service in Laravel framework.
- How to work with json files.
- For each request from the browser, we will receive it by the router file, determine the appropriate method for it, and after that the controller will handle it within that method and return the response to the user as a JSON/Raw content.

9- Exercise

Create a new Laravel project with 2 end-points:

- GET request to get list of user's names after capitalize all names.
- Add a new JSON file with array of object, each one contains name, email and phone number, and return a success message only if the user sent an existed email or/and phone number.