
Laravel

# Outlines

- Routing in Laravel
- Controllers in Laravel
- View in Laravel
- Blade Template
- Components in laravel

app/Http/Controllers/ChirpController.php

```php
<?php
class ChirpController extends Controller
{
    public function store(Request $request)
    {
        $validated = $request->validate([
            'message' => 'required|string|max:255',
        ]);

        $request->user()->chirps()->create($validated);

        return redirect(route('chirps.index'));
    }
}
```

# Routing in laravel

# Routing in Laravel

## 1. Defining Routes

In Laravel, routes are defined in the `routes/web.php` file. Routes usually specify which URL should be handled by which controller or closure.

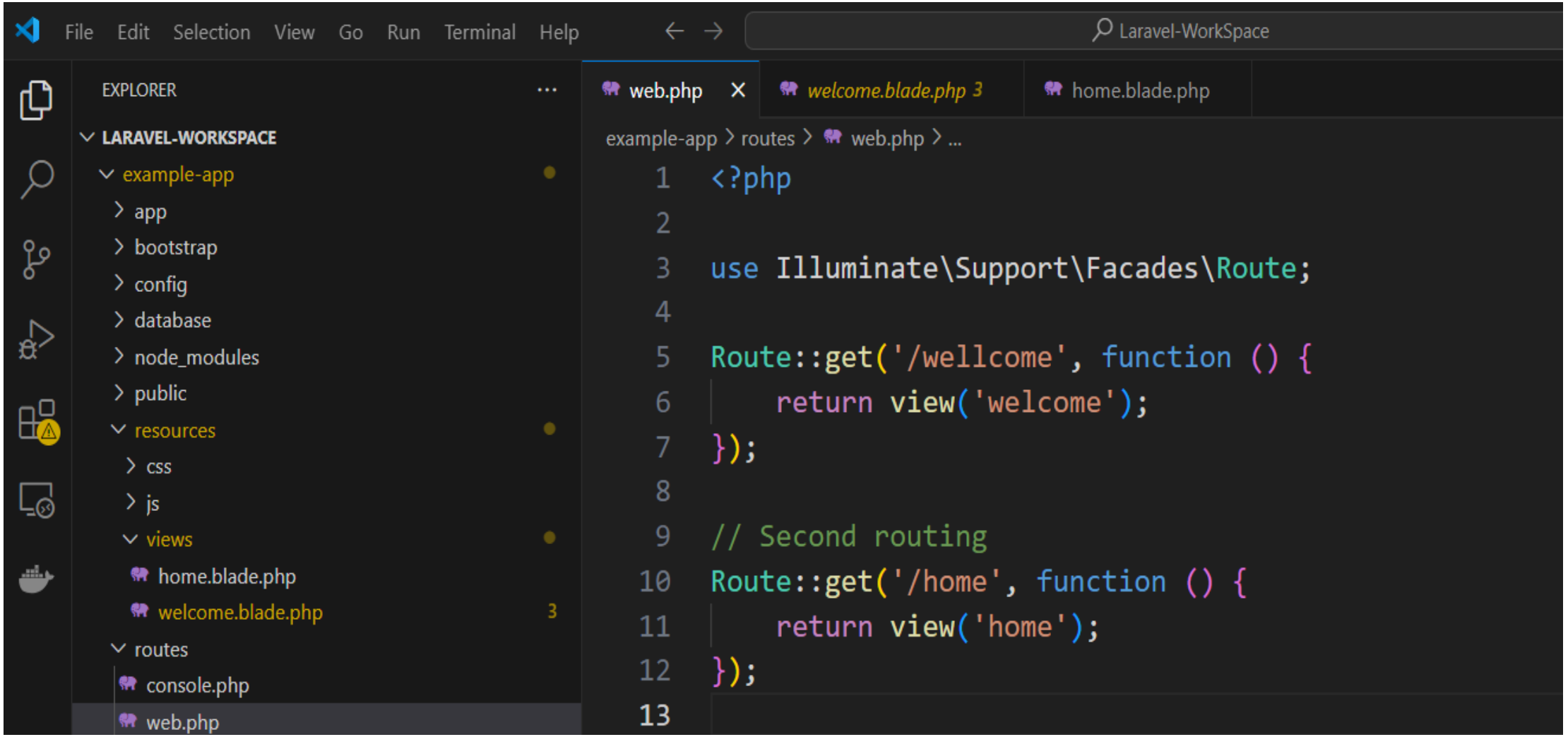
**Example: Basic Route Definition**

php Copy code

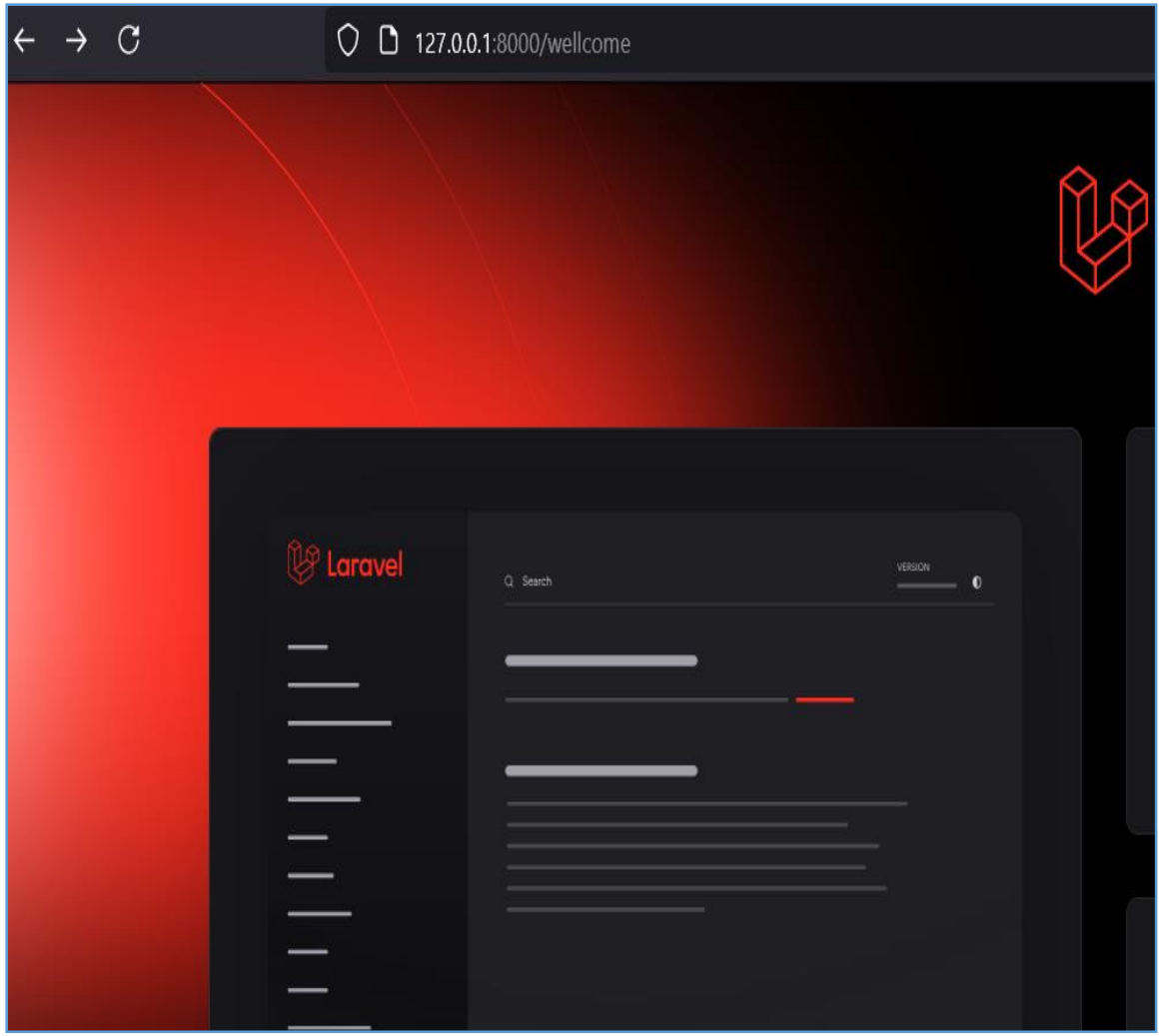
```php
Route::get('/welcome', function () {
    return 'Welcome to Laravel!';
});
```

- The above code defines a route that responds to a GET request to the `/welcome` URL and returns the text "Welcome to Laravel!".

# Example


File Edit Selection View Go Run Terminal Help
Laravel-WorkSpace
EXPLORER
LARAVEL-WORKSPACE
example-app
app
bootstrap
config
database
node_modules
public
resources
css
js
views
home.blade.php
welcome.blade.php 3
routes
console.php
web.php
web.php
welcome.blade.php 3
home.blade.php
example-app > routes > web.php > ...
1 <?php
2
3 use Illuminate\Support\Facades\Route;
4
5 Route::get('/wellcome', function () {
6 return view('welcome');
7 });
8
9 // Second routing
10 Route::get('/home', function () {
11 return view('home');
12 });
13

# Cont. …




127.0.0.1:8000/home

**Well come to Home**

# Cont. ...

## 2. Route Methods

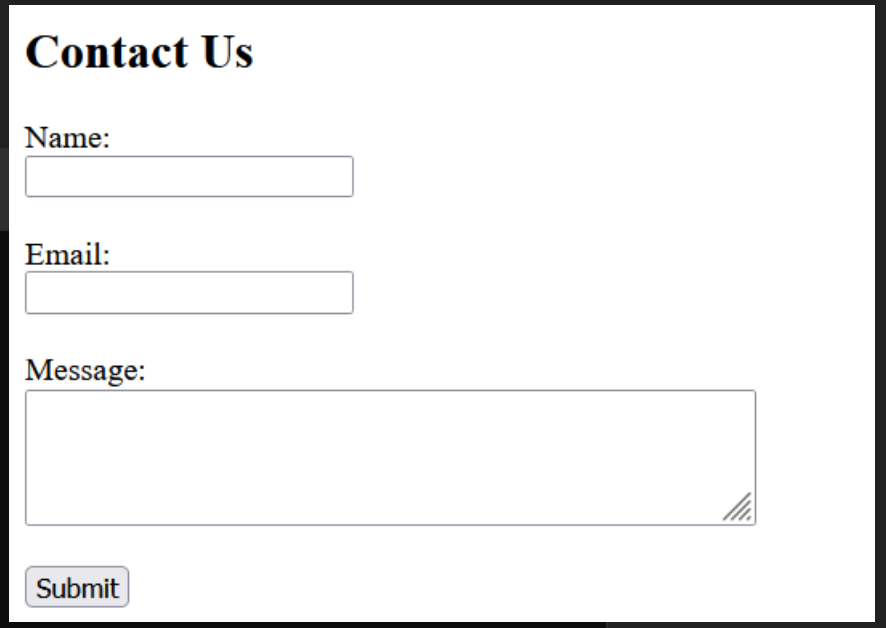
Laravel provides different HTTP methods (GET, POST, PUT, DELETE) to define routes for specific types of requests. These can be used to handle actions such as form submissions, data updates, and deletions.

**Example: Handling GET and POST requests**

php

```php
// GET route to display a form
Route::get('/form', function () {
    return view('form');
});

// POST route to handle form submission
Route::post('/submit', function () {
    // Logic to handle submitted form data
    return 'Form submitted!';
});
```




- The first route displays a form, while the second route handles form submissions using the `POST` method.

# Cont. ...

**Example: PUT and DELETE methods**

php Copy code

```php
// PUT route to update a resource
Route::put('/update/{id}', function ($id) {
    // Logic to update a resource with the given ID
    return "Updated resource with ID: $id";
});

// DELETE route to delete a resource
Route::delete('/delete/{id}', function ($id) {
    // Logic to delete a resource with the given ID
    return "Deleted resource with ID: $id";
});
```

- These routes handle updating and deleting resources based on their unique ID.

# Cont. …

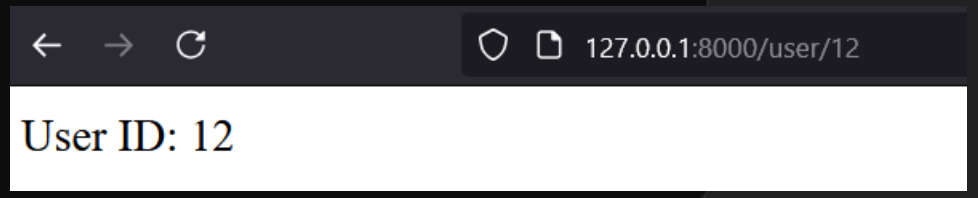
## 3. Route Parameters

Laravel allows you to define dynamic routes with parameters. These parameters are passed to the route's function or controller.

**Example: Route with Required Parameter**

php Copy code

```php
Route::get('/user/{id}', function ($id) {
    return "User ID: " . $id;
});
```




- This route captures a dynamic parameter `id`, which can be any value. For example, accessing `/user/5` will output "User ID: 5".

# Cont. ...

### Example: Route with Optional Parameter

php Copy code

```php
Route::get('/profile/{name?}', function ($name = 'Guest') {
    return "Profile Name: " . $name;
});
```

- In this example, the parameter `name` is optional. If no name is provided, it defaults to "Guest".

# Cont. ...

### Example: Defining a Named Route

php Copy code

```php
Route::get('/dashboard', function () {
    return 'Dashboard';
})->name('dashboard');
```

- You can now refer to this route using the route's name ( `dashboard` ) anywhere in the application, for example, when generating links.

### Example: Generating URLs Using Named Routes

php Copy code

```php
// In a Blade template or controller
$url = route('dashboard'); // Generates the URL for the dashboard route
```

# Cont. …

## 5. Route Groups

Route groups allow you to apply common attributes like middleware, namespaces, or prefixes to multiple routes.

### Example: Grouping Routes with a Common Prefix

php Copy code

```php
Route::prefix('admin')->group(function () {
    Route::get('/users', function () {
        return 'Admin Users';
    });

    Route::get('/settings', function () {
        return 'Admin Settings';
    });
});
```

- This defines a group of routes with the prefix `/admin`. The resulting URLs would be `/admin/users` and `/admin/settings`.

# Cont. ...

## 6. Route Redirects and Route Views

Laravel makes it easy to quickly define routes that either redirect to another URL or return a view directly.

### Example: Route Redirect

php Copy code

```php
Route::redirect('/home', '/dashboard');
```

- This redirects any request to `/home` to `/dashboard`.

# Cont. ...

### Example: Returning a View from a Route

php Copy code

```php
Route::view('/welcome', 'welcome');
```

- This route directly returns the `welcome.blade.php` view when the `/welcome` URL is accessed.

# Cont. ...

## 7. Route Middleware

Middleware is used to filter requests entering your application, like checking if a user is authenticated before allowing access to certain routes.

### Example: Applying Middleware to Routes

php Copy code

```php
Route::get('/dashboard', function () {
    return 'Dashboard';
})->middleware('auth');
```

- This route is only accessible to authenticated users, as enforced by the `auth` middleware.

# Cont. ...

## 8. Resourceful Routing

Laravel provides a simple way to create a controller that handles all CRUD operations with a single line of code using resourceful routes.

### Example: Defining a Resource Controller

php Copy code

```php
Route::resource('posts', 'PostController');
```

- This automatically sets up routes for all the common CRUD actions (index, create, store, show, edit, update, destroy) in the `PostController`.

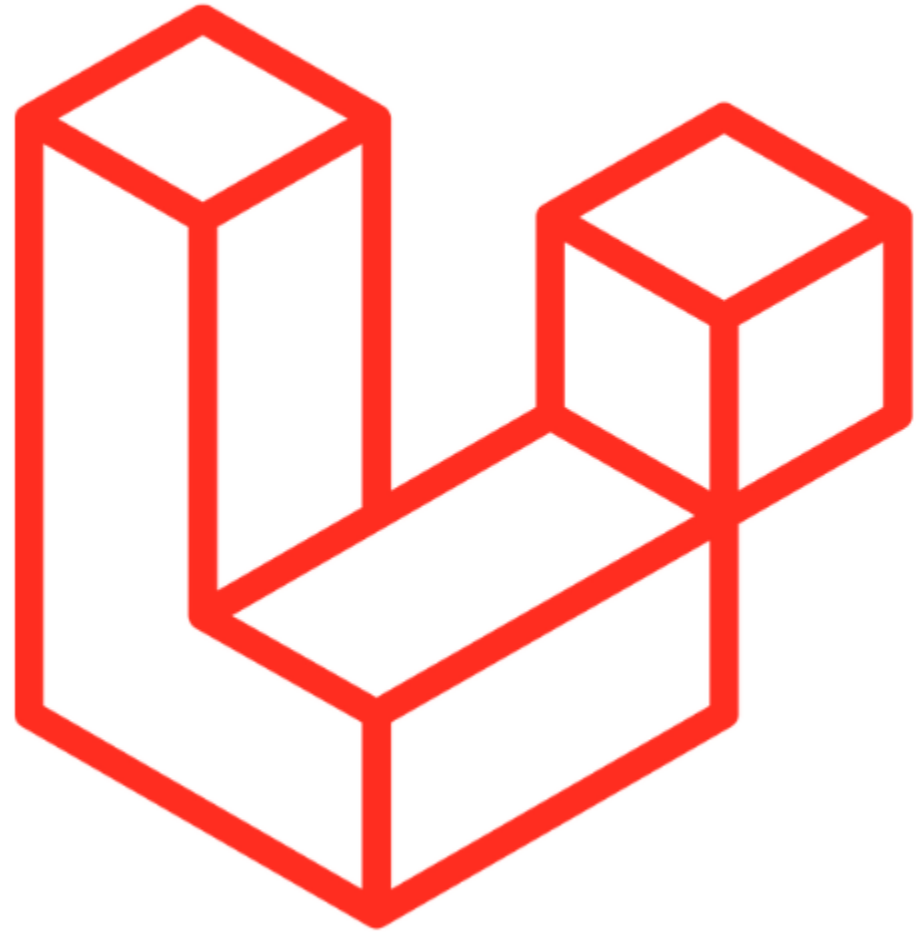
# Cont. ...

**Summary of Key Routing Concepts:**

- **Defining Routes**: Basic routes are created in `routes/web.php`, with support for GET, POST, PUT, DELETE methods.
- **Route Parameters**: Dynamic values are captured in routes, and parameters can be required or optional.
- **Named Routes**: Simplify route management by using names instead of hardcoded URLs.
- **Route Groups**: Apply common behavior like middleware or prefixes to multiple routes.
- **Middleware**: Routes can be filtered through middleware for tasks like authentication.
- **Redirects and Views**: Easily redirect users or return views directly from routes.
- **Resourceful Routing**: Handle all CRUD operations in a controller using resource routes.

# LARAVEL

## Controllers



# Controllers in Laravel

- In Laravel, controllers are a vital part of the MVC (Model-View-Controller) architecture, which separates the **application logic** from the user **interface**.
- Controllers handle the **incoming HTTP requests**, **process data** (usually with the help of models), and return responses (typically views or JSON data).
- Controllers allow you to organize your application logic more cleanly compared to defining logic in the route files.

# Creating Controller

## 1. Creating a Controller

To create a controller, you can use Laravel's Artisan command-line tool. You can define controllers to handle various HTTP requests and organize your application code.

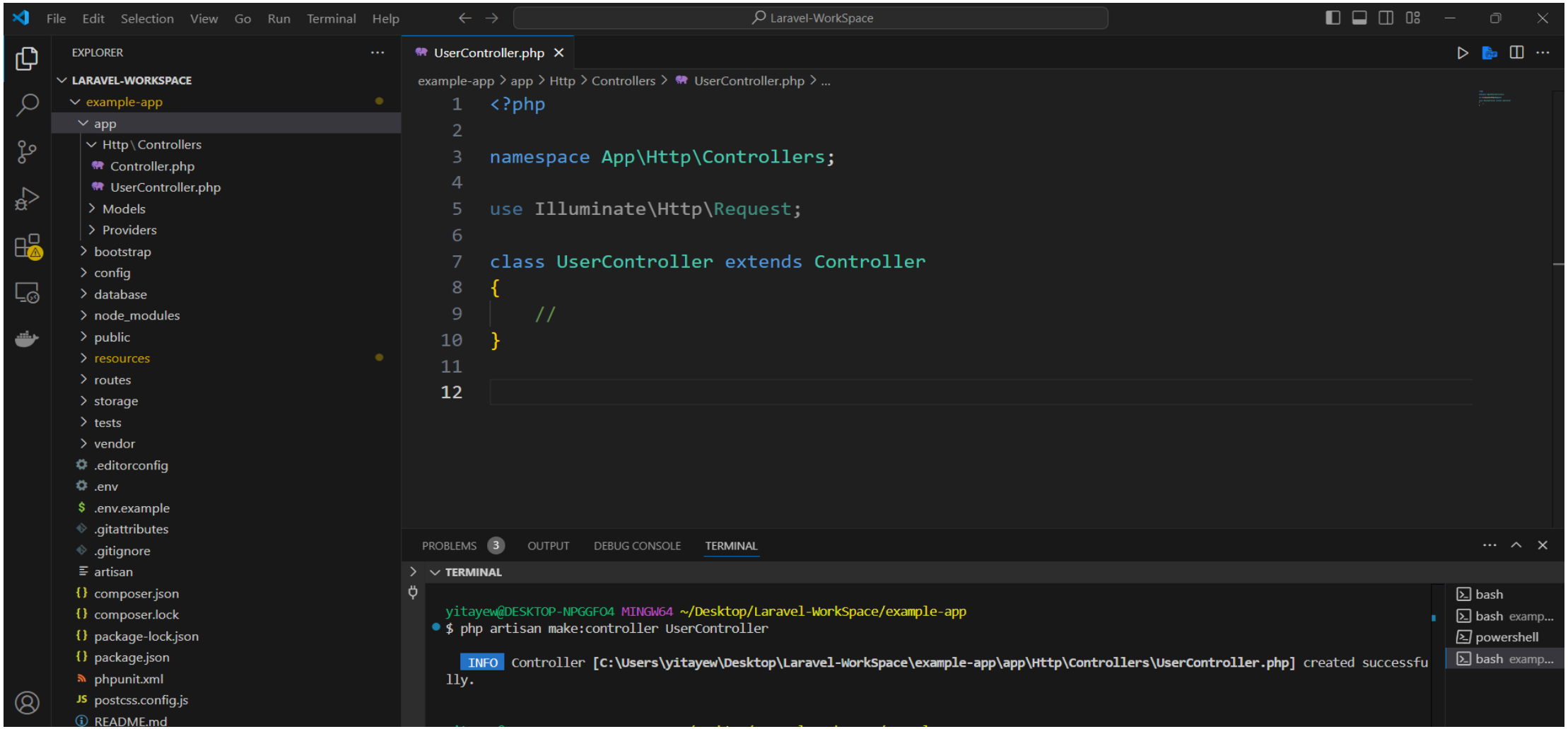
### Example: Creating a Basic Controller

bash Copy code

```
php artisan make:controller UserController
```

- This command creates a new `UserController` inside the `app/Http/Controllers` directory.

# Cont. …


File Edit Selection View Go Run Terminal Help
Laravel-WorkSpace
EXPLORER
LARAVEL-WORKSPACE
example-app
app
Http \ Controllers
Controller.php
UserController.php
Models
Providers
bootstrap
config
database
node_modules
public
resources
routes
storage
tests
vendor
.editorconfig
.env
.env.example
.gitattributes
.gitignore
artisan
composer.json
composer.lock
package-lock.json
package.json
phpunit.xml
postcss.config.js
README.md
UserController.php
example-app > app > Http > Controllers > UserController.php > ...
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class UserController extends Controller
8 {
9 //
10 }
11
12
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL
TERMINAL
yitayew@DESKTOP-NPGGFO4 MINGW64 ~/Desktop/Laravel-WorkSpace/example-app
$ php artisan make:controller UserController
INFO Controller [C:\Users\yitayew\Desktop\Laravel-WorkSpace\example-app\app\Http\Controllers\UserController.php] created successfully.
bash
bash examp...
powershell
bash examp...

# Cont. ...

## 2. Routing to a Controller

Once the controller is created, you can define routes that map to the controller's methods. This allows your application to handle various URLs and HTTP methods using controller actions.

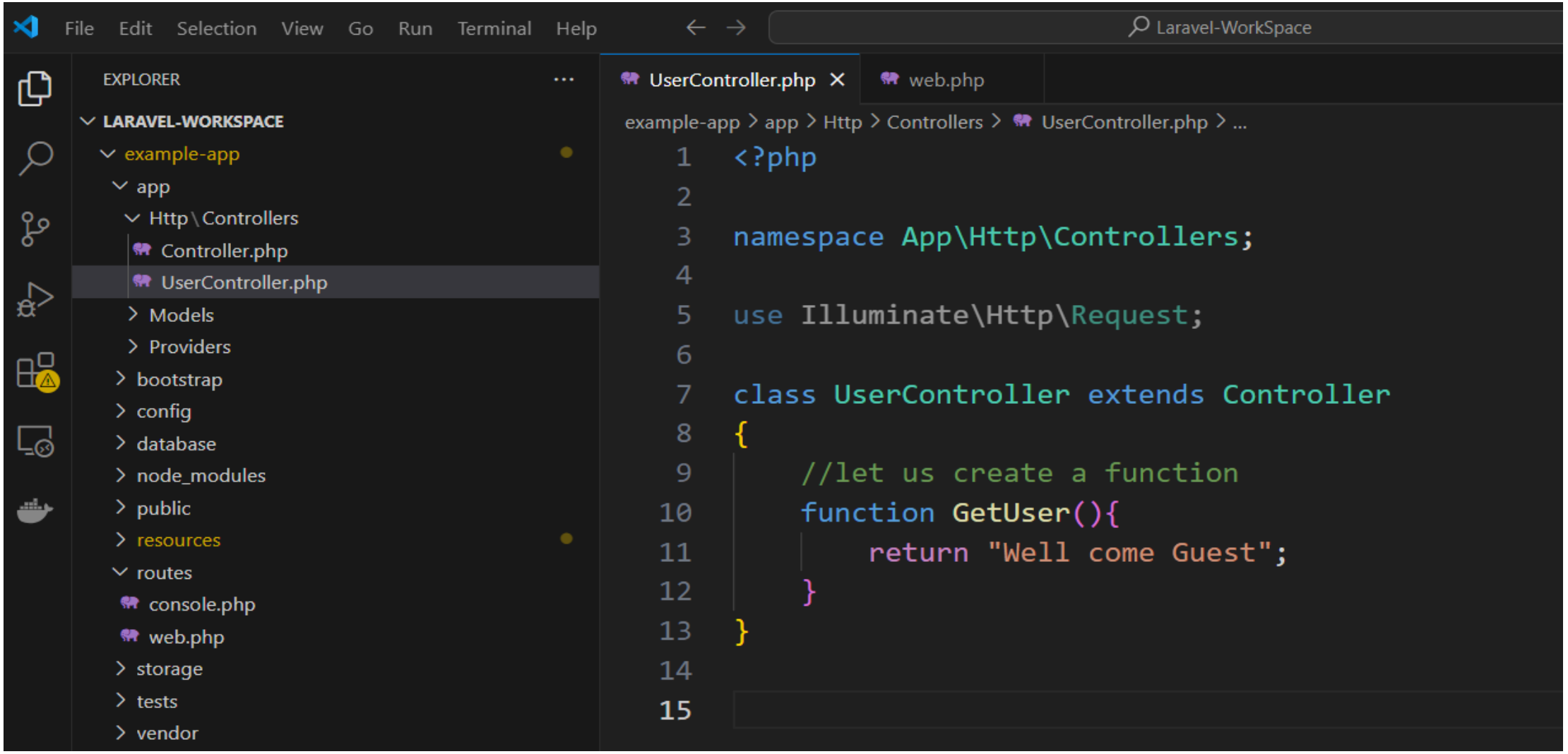
### Example: Defining Routes for Controller Methods

php Copy code

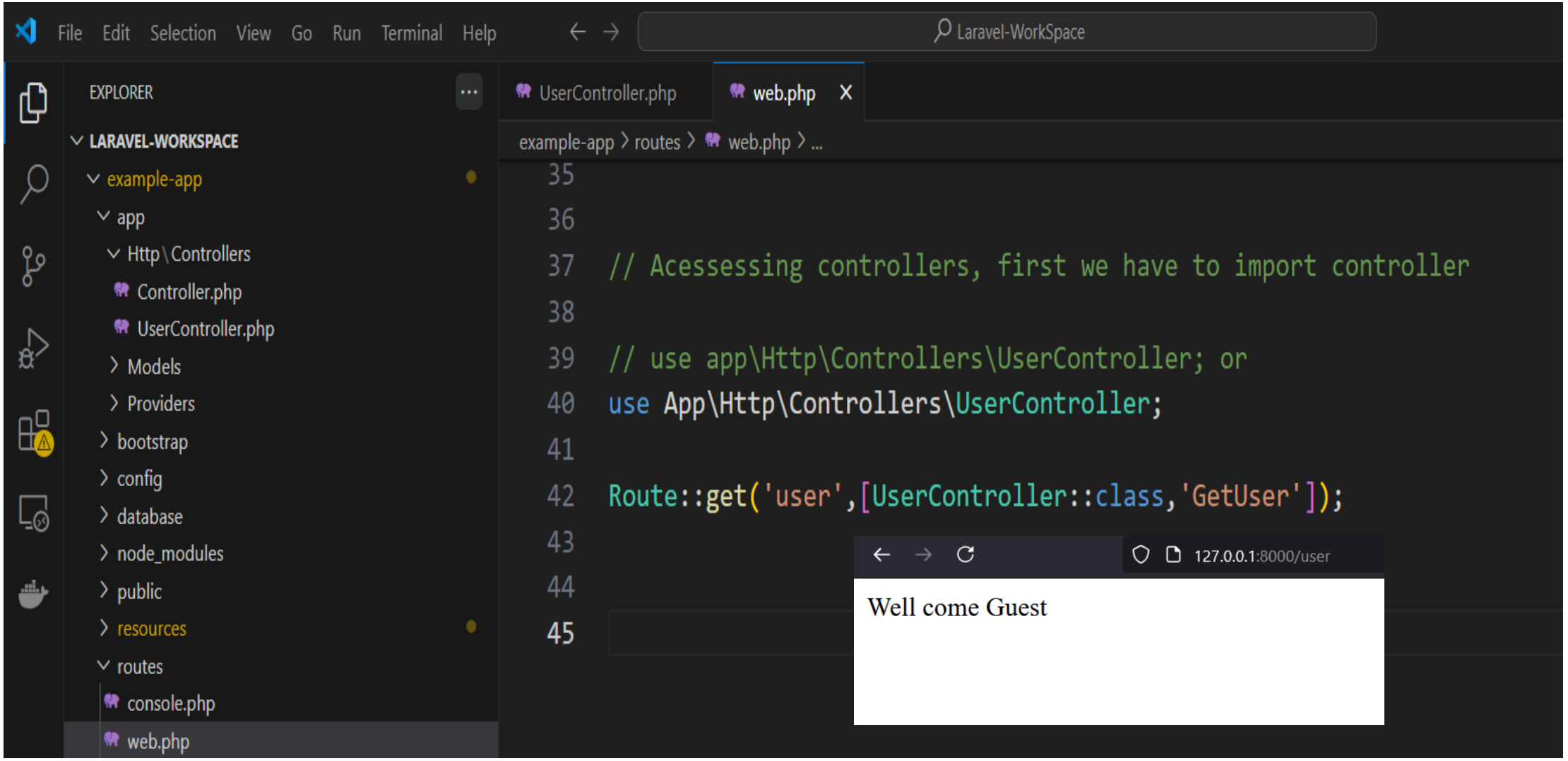
```php
// Defining routes for the UserController
Route::get('/users', [UserController::class, 'index']);
Route::get('/users/{id}', [UserController::class, 'show']);
```

- When accessing `/users`, the `index` method of the `UserController` is executed.
- When accessing `/users/{id}`, the `show` method is executed, displaying the user's ID.

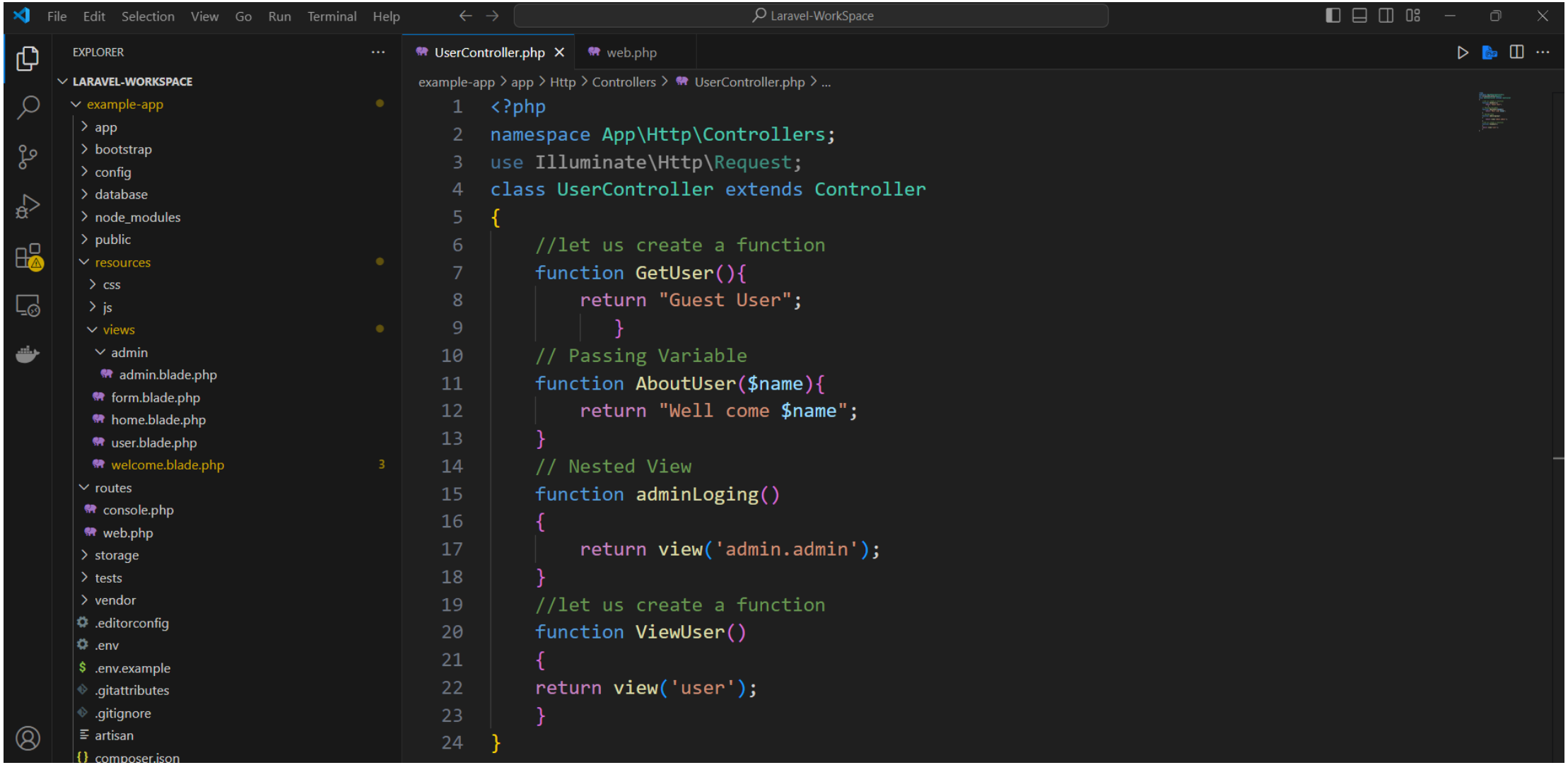
# Defining Controller (User Controller)



# Creating Route for UserController



# Calling From View



# Cont. ...

```php
// Acessessing controllers, first we have to import controller

// use app\Http\Controllers\UserController; or
use App\Http\Controllers\UserController;

Route::get('user',[UserController::class,'ViewUser']);
Route::get('about/{name}',[UserController::class,'AboutUser']);
Route::get('admin-login',[UserController::class,'adminLoging']);
```

# Cont. ...

## 3. Resource Controllers

Laravel provides a powerful way to manage CRUD (Create, Read, Update, Delete) operations using **Resource Controllers**. A resource controller automatically creates routes for all standard CRUD actions (index, create, store, show, edit, update, destroy).

### Example: Creating a Resource Controller

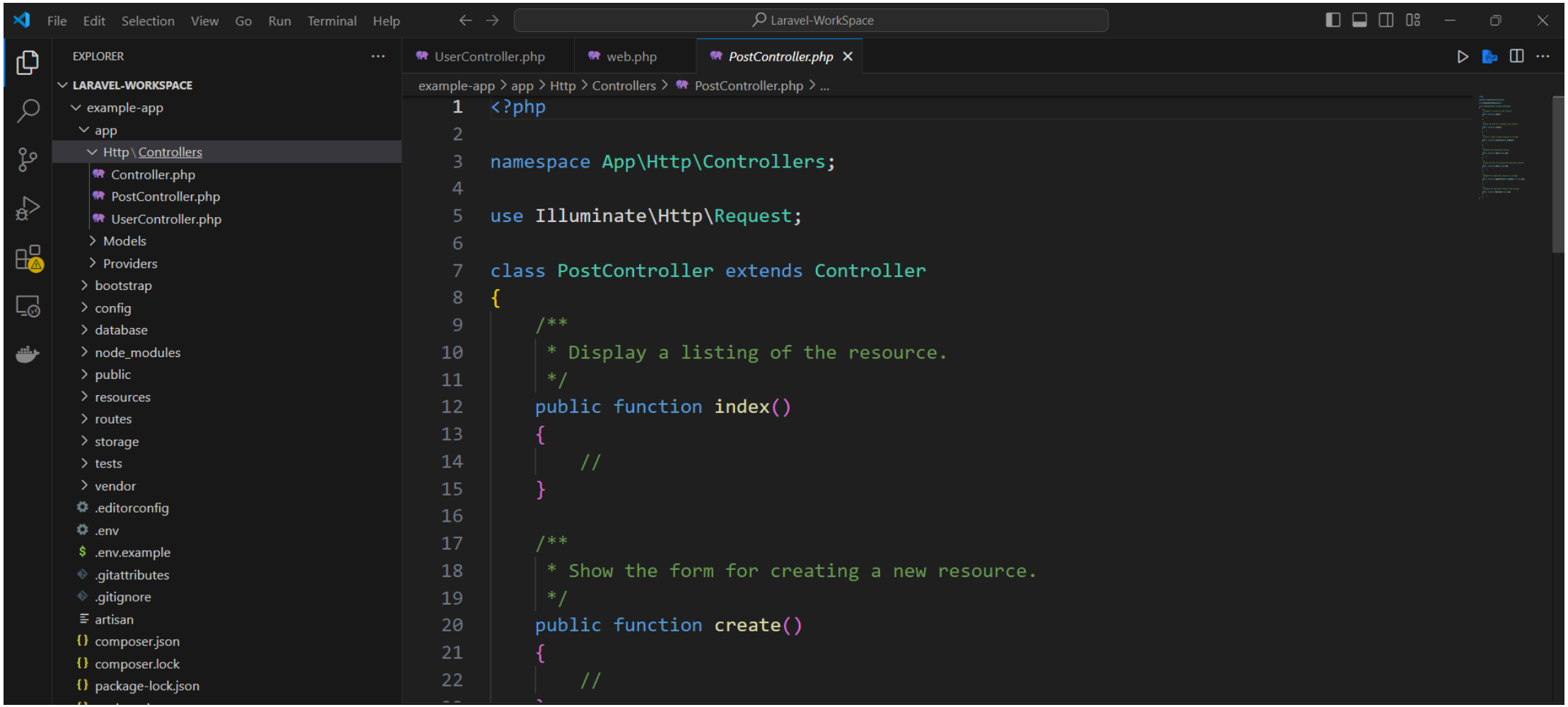
bash Copy code

```
php artisan make:controller PostController --resource
```

# Cont. ...

- This command creates a `PostController` with pre-defined methods for handling CRUD operations:
  - `index()` - List all posts.
  - `create()` - Show form for creating a new post.
  - `store()` - Save the new post to the database.
  - `show($id)` - Show a specific post.
  - `edit($id)` - Show form to edit a post.
  - `update($id)` - Update a specific post.
  - `destroy($id)` - Delete a specific post.

# Cont. ...


File Edit Selection View Go Run Terminal Help
Laravel-WorkSpace
EXPLORER
LARAVEL-WORKSPACE
example-app
app
Http \ Controllers
Controller.php
PostController.php
UserController.php
Models
Providers
bootstrap
config
database
node_modules
public
resources
routes
storage
tests
vendor
.editorconfig
.env
.env.example
.gitattributes
.gitignore
artisan
composer.json
composer.lock
package-lock.json
UserController.php
web.php
PostController.php
example-app > app > Http > Controllers > PostController.php > ...
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class PostController extends Controller
{
/**
* Display a listing of the resource.
*/
public function index()
{
//
}
/**
* Show the form for creating a new resource.
*/
public function create()
{
//

# Cont. ...

**Example: Defining a Resource Route**

php Copy code

```php
Route::resource('posts', PostController::class);
```

- This single line generates all the necessary routes for the above CRUD actions (index, show, store, update, etc.).

**Example: Generated Routes**

bash Copy code

```bash
GET       /posts           → index
GET       /posts/create    → create
POST      /posts           → store
GET       /posts/{id}      → show
GET       /posts/{id}/edit→ edit
PUT/PATCH /posts/{id}      → update
DELETE    /posts/{id}      → destroy
```

# Cont. ...

## 4. Single Action Controllers

In cases where you need a controller with only a single action, Laravel allows the creation of **invokable controllers**. These controllers use the `__invoke()` method and can be defined with a single method.

### Example: Creating an Invokable Controller

bash Copy code

```
php artisan make:controller ContactController --invokable
```

- This creates a controller with just one method `__invoke()`:

# Cont. ...

- This creates a controller with just one method `__invoke()`:

php Copy code

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ContactController extends Controller
{
    public function __invoke() {
        return view('contact');
    }
}
```

# Cont. ...

- Define the route like this:

php Copy code

```php
Route::get('/contact', ContactController::class);
```

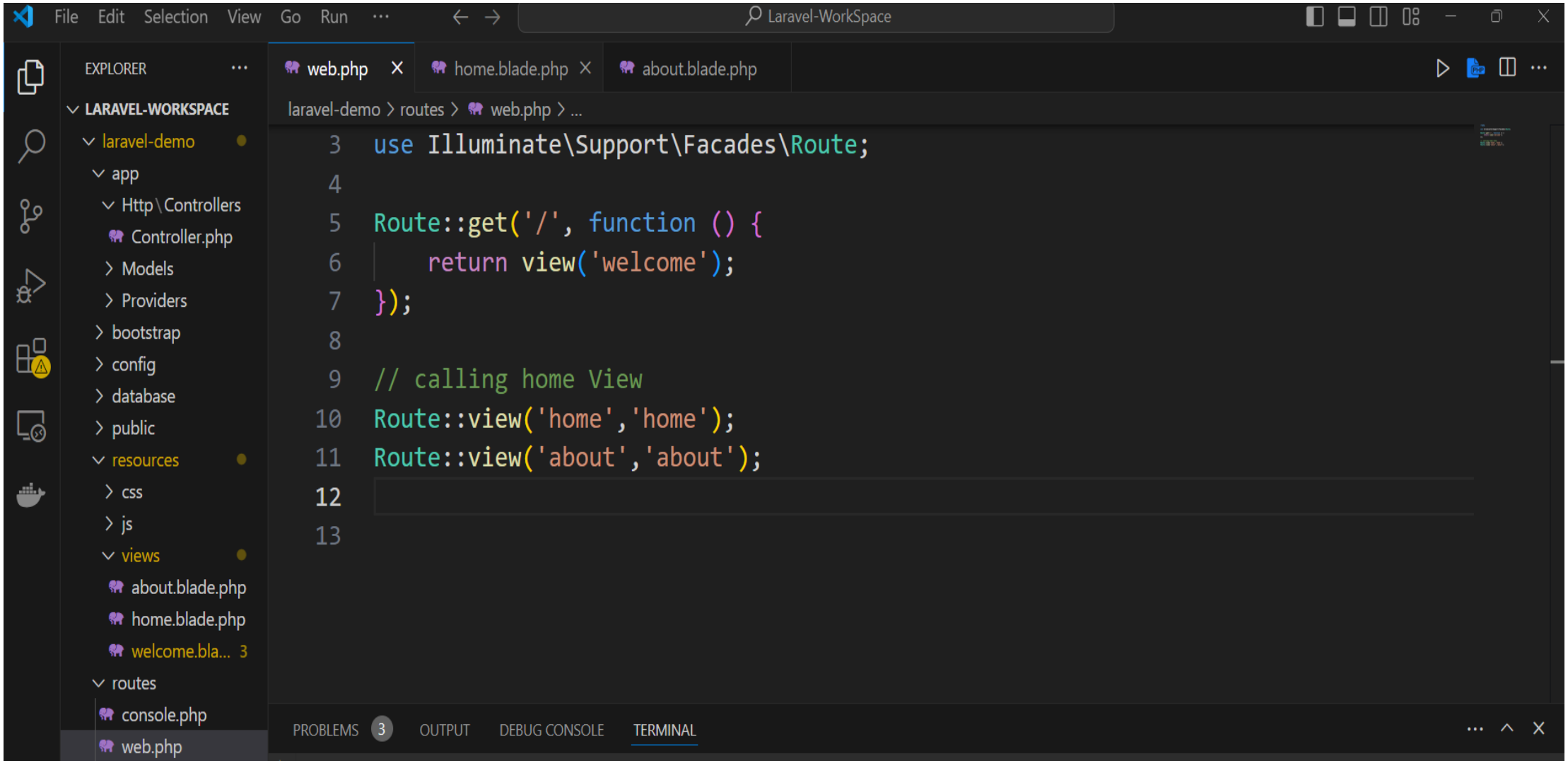
- Here, the controller is directly mapped to a route without specifying a method.


# Laravel Views

# View in Laravel

- In Laravel, views are used to separate the **application logic** from the **presentation layer**. They allow developers to create HTML output in organized templates, making it easier to build dynamic content.
- Views are stored as **.blade.php** files in the **resources/views** directory, where Blade templating enables efficient layouts, reusability, and simplified syntax.
- Here’s how to work with views in Laravel:

# Cont. ...


File Edit Selection View Go Run
Laravel-WorkSpace
EXPLORER
web.php
home.blade.php
about.blade.php
laravel-demo > routes > web.php > ...
LARAVEL-WORKSPACE
laravel-demo
app
Http\Controllers
Controller.php
Models
Providers
bootstrap
config
database
public
resources
css
js
views
about.blade.php
home.blade.php
welcome.bla... 3
routes
console.php
web.php
3 use Illuminate\Support\Facades\Route;
4
5 Route::get('/', function () {
6 return view('welcome');
7 });
8
9 // calling home View
10 Route::view('home','home');
11 Route::view('about','about');
12
13
PROBLEMS 3
OUTPUT
DEBUG CONSOLE
TERMINAL

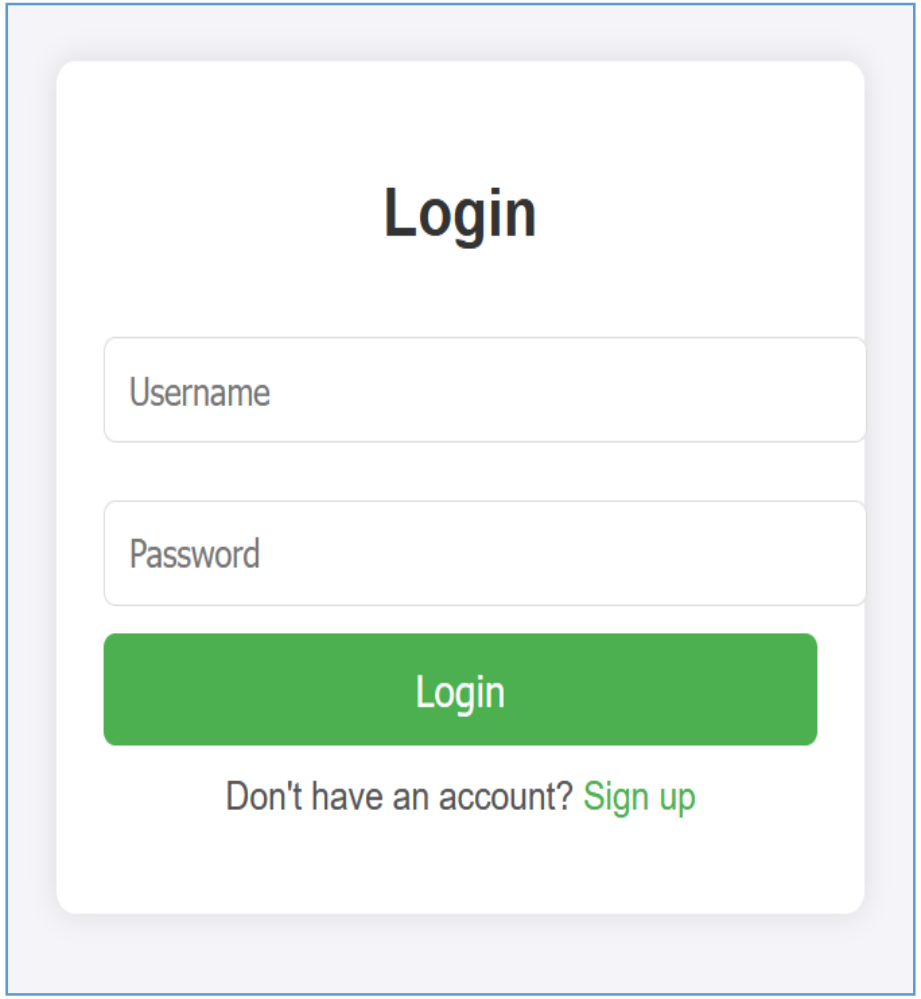
# Cont. …

web.php | home.blade.php | about.blade.php

laravel-demo > routes > web.php > ...

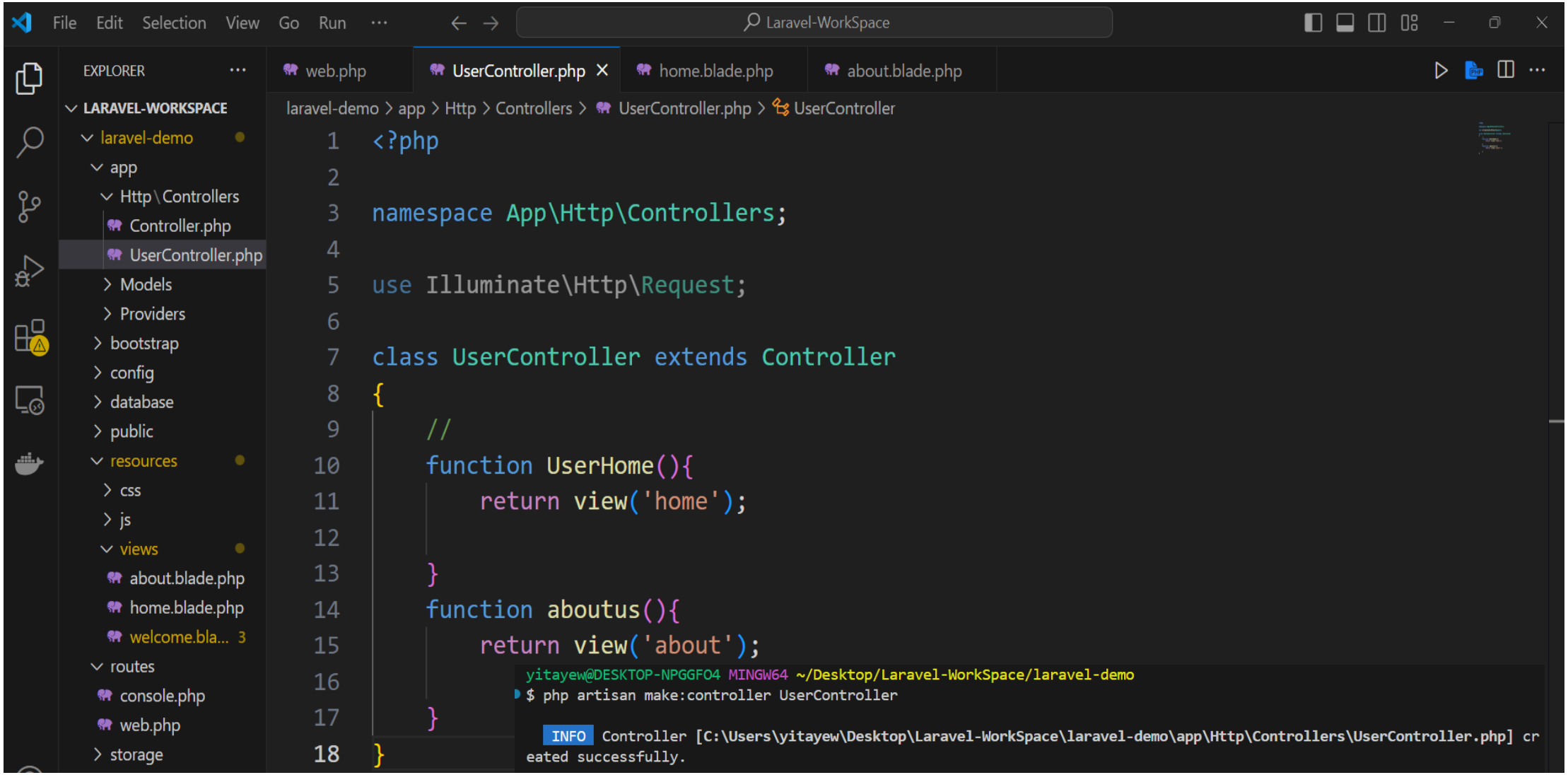
```php
Route::get('/', function () {
    return view('welcome');
});

// calling home View
Route::view('home','home');
```

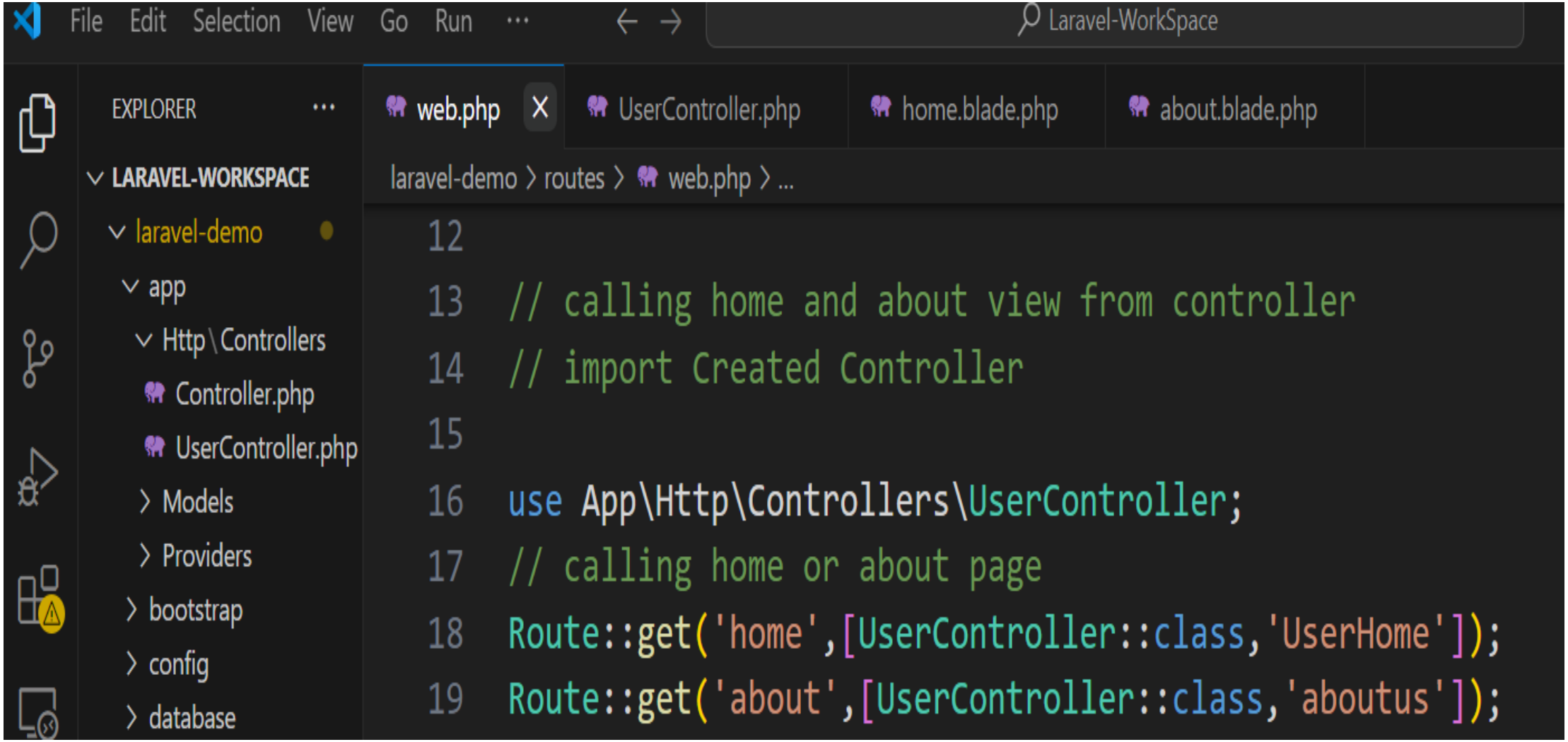



```
yitayew@DESKTOP-NPGGFO4 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo
$ php artisan make:view about
```

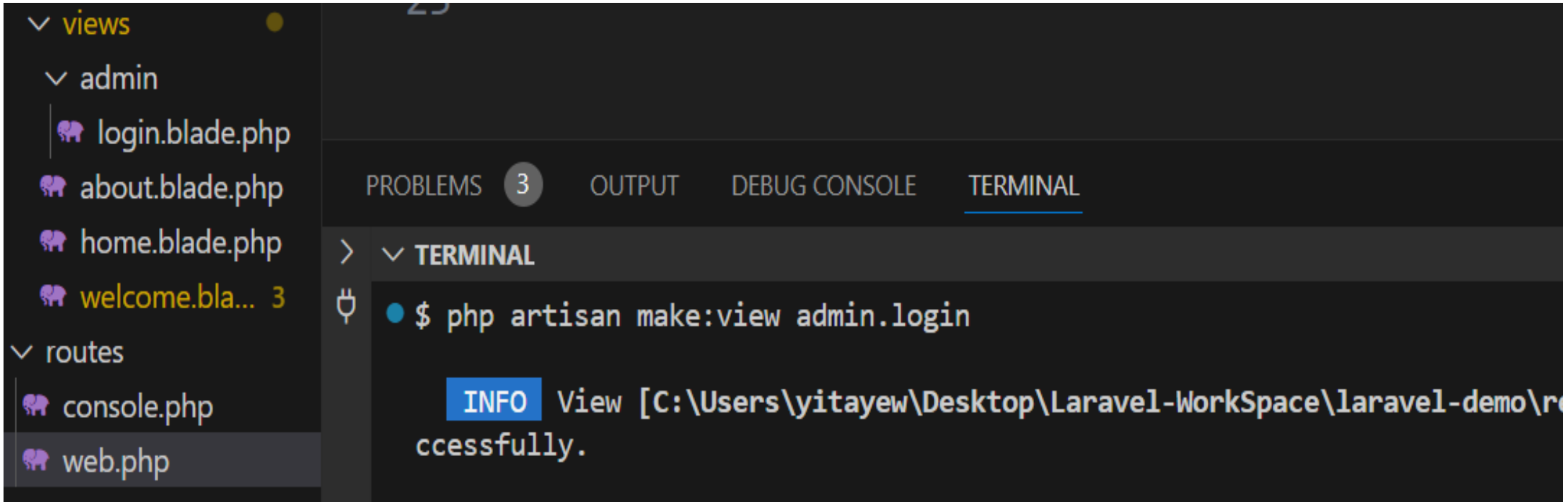
# Calling View From Controller



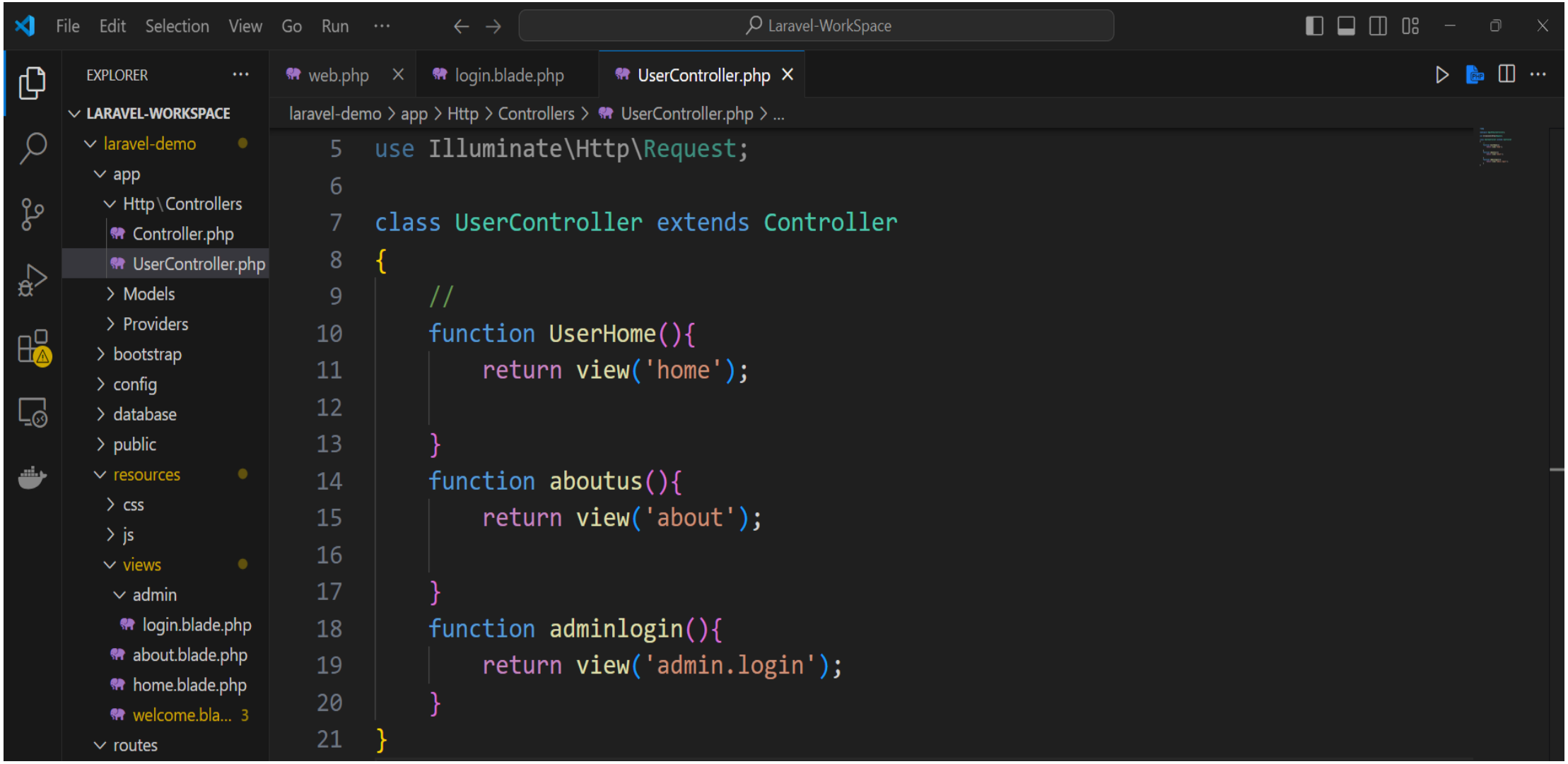
# Cont. ...


File Edit Selection View Go Run ···
Laravel-WorkSpace
EXPLORER
web.php
UserController.php
home.blade.php
about.blade.php
LARAVEL-WORKSPACE
laravel-demo
app
Http\Controllers
Controller.php
UserController.php
Models
Providers
bootstrap
config
database
laravel-demo > routes > web.php > ...
12
13 // calling home and about view from controller
14 // import Created Controller
15
16 use App\Http\Controllers\UserController;
17 // calling home or about page
18 Route::get('home',[UserController::class,'UserHome']);
19 Route::get('about',[UserController::class,'aboutus']);

# Nesting View



# Cont. ...


File Edit Selection View Go Run
Laravel-WorkSpace
EXPLORER
web.php
login.blade.php
UserController.php
laravel-demo > app > Http > Controllers > UserController.php > ...
LARAVEL-WORKSPACE
laravel-demo
app
Http\Controllers
Controller.php
UserController.php
Models
Providers
bootstrap
config
database
public
resources
css
js
views
admin
login.blade.php
about.blade.php
home.blade.php
welcome.bla... 3
routes
use Illuminate\Http\Request;
class UserController extends Controller
{
//
function UserHome(){
return view('home');
}
function aboutus(){
return view('about');
}
function adminlogin(){
return view('admin.login');
}
}

# Cont. ...

web.php | login.blade.php | UserController.php

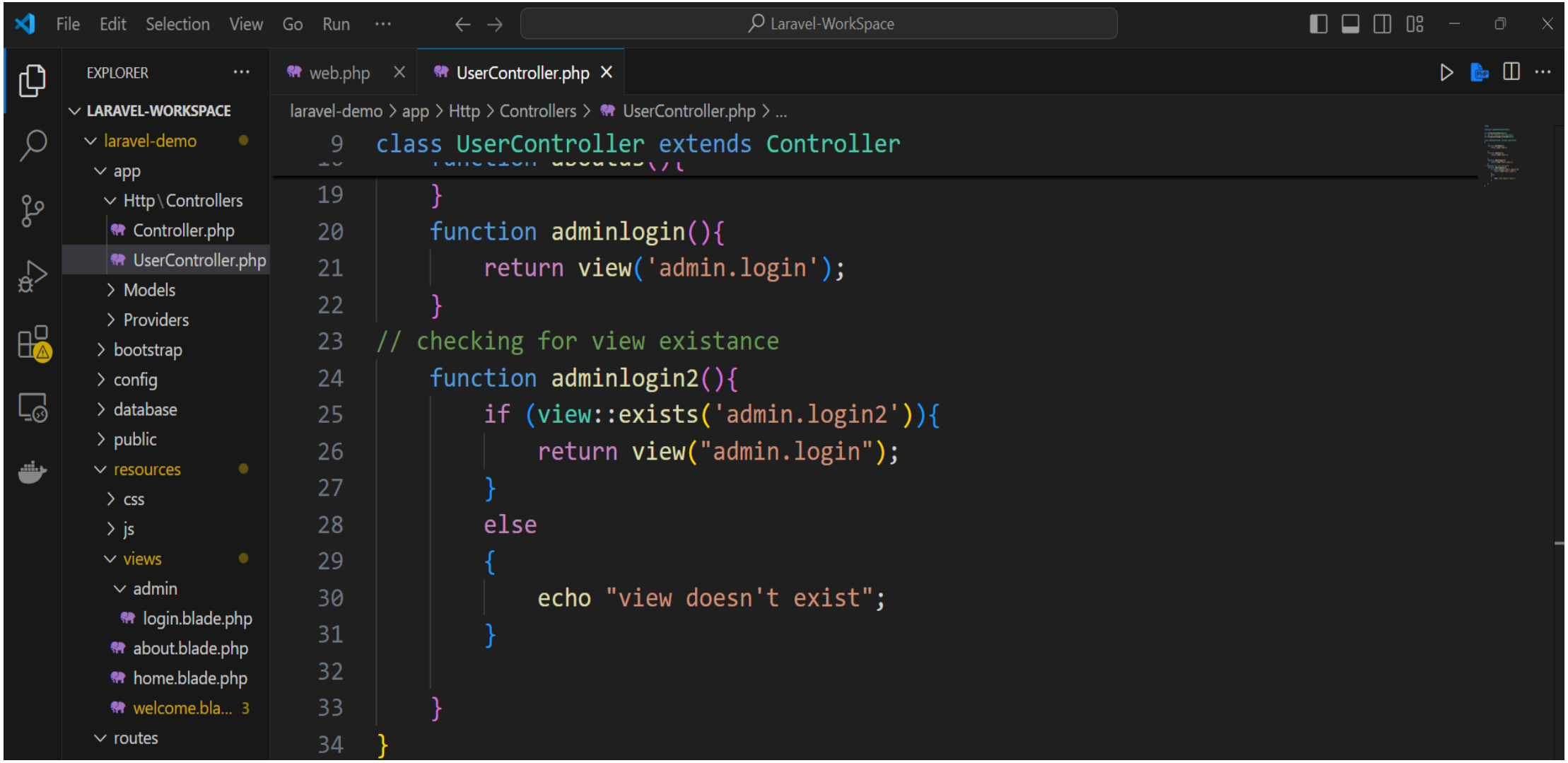
laravel-demo > routes > web.php > ...

```php
// calling home and about View from Route
Route::view('home','home');
Route::view('about','about');
Route::view('admin','admin.login');

// calling home and about view from controller
// import Created Controller

use App\Http\Controllers\UserController;
// calling home or about page
Route::get('home',[UserController::class,'UserHome']);
Route::get('about',[UserController::class,'aboutus']);
Route::get('adminlog',[UserController::class,'adminlogin']);
```

# Checking view Existence

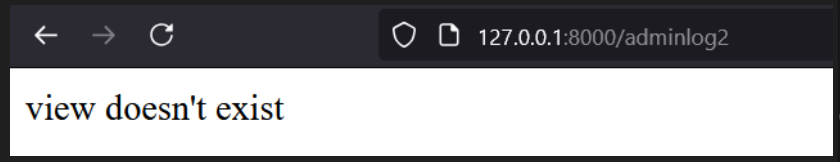


# Cont. ...

web.php | UserController.php

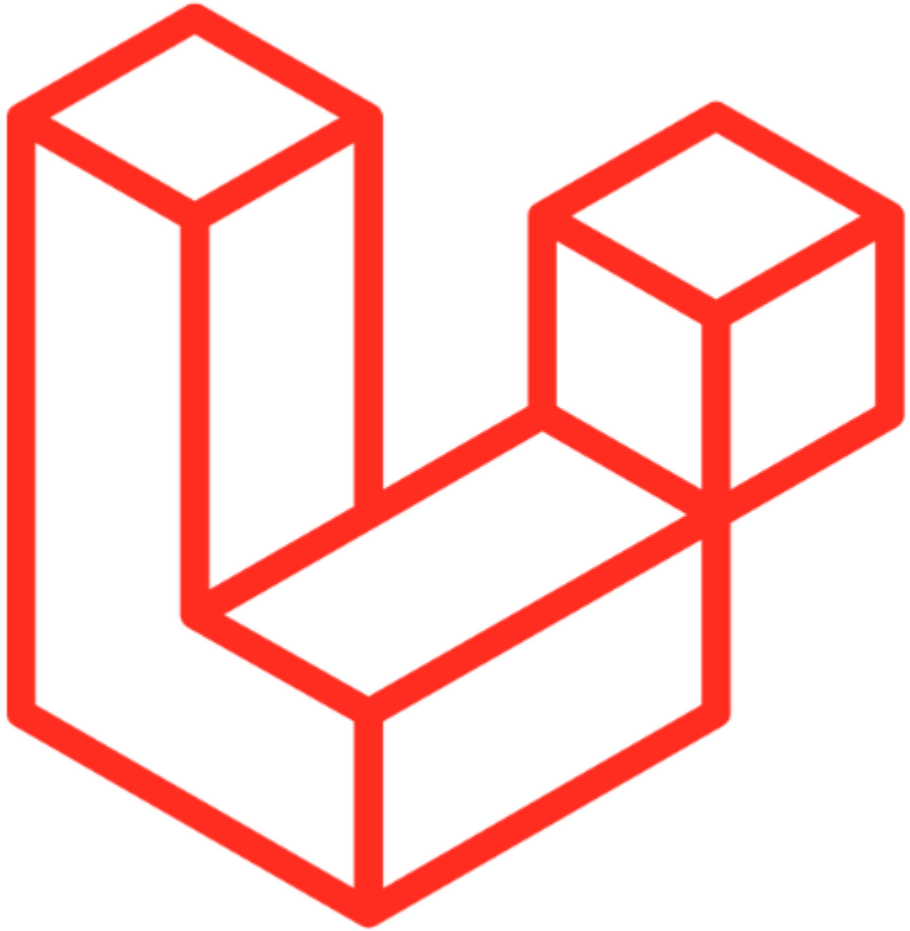
laravel-demo > routes > web.php > ...

```
 8
 9  // calling home and about View from Route
10  Route::view('home','home');
11  Route::view('about','about');
12  Route::view('admin','admin.login');
13
14  // calling home and about view from controller
15  // import Created Controller
16
17  use App\Http\Controllers\UserController;
18  // calling home or about page
19  Route::get('home',[UserController::class,'UserHome']);
20  Route::get('about',[UserController::class,'aboutus']);
21  Route::get('adminlog',[UserController::class,'adminlogin']);
22  Route::get('adminlog2',[UserController::class,'adminlogin2']);
```



# LARAVEL

## Blade Templates



# Blade Template

- In Laravel, Blade is a simple yet powerful templating engine that provides an easy way to create **dynamic HTML pages**.
- It allows you to **structure your application's** layout and inject data from controllers, making it one of the key tools for Laravel developers.

# Cont. ...

- Key Features of Blade:

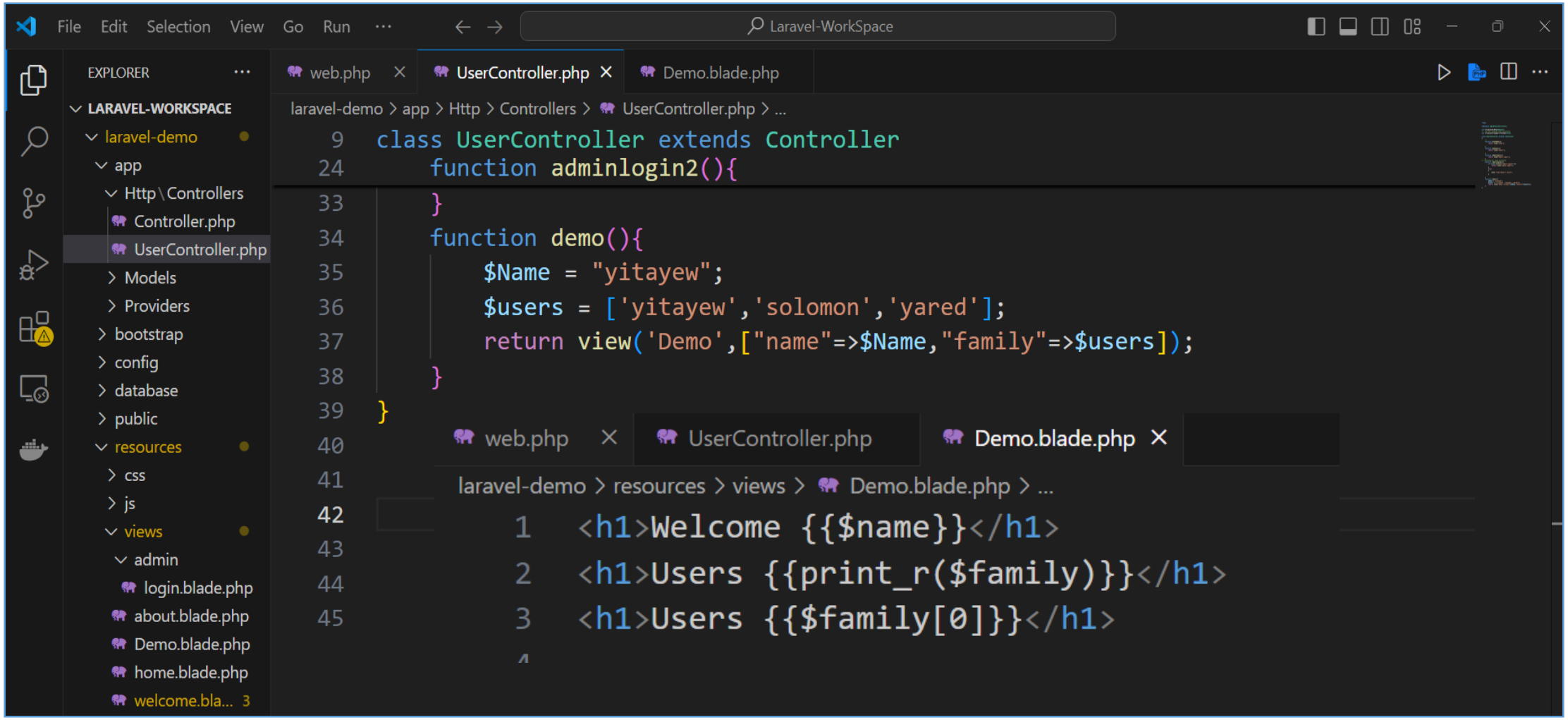
✓**Template Inheritance**: Enables layout reuse with sections for flexible content.

✓**Echoing Data**: Simple syntax for outputting variables and HTML.

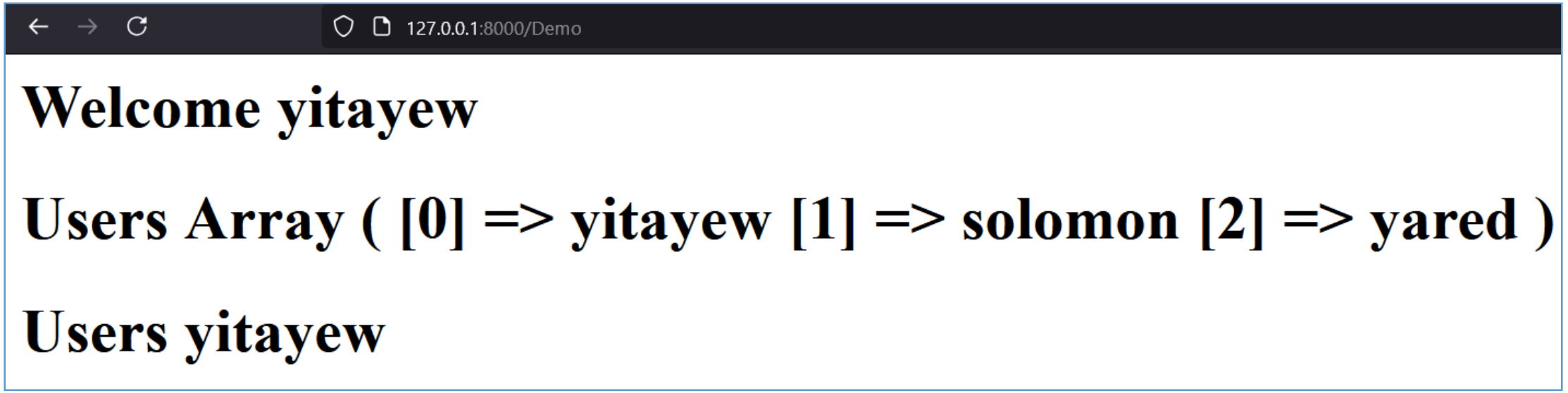
✓**Conditional Statements and Loops**: Control structures for dynamic content.

✓ **Components and Slots**: Reusable and customizable template fragments.

# Example


File Edit Selection View Go Run
Laravel-WorkSpace
EXPLORER
LARAVEL-WORKSPACE
laravel-demo
app
Http \ Controllers
Controller.php
UserController.php
Models
Providers
bootstrap
config
database
public
resources
css
js
views
admin
login.blade.php
about.blade.php
Demo.blade.php
home.blade.php
welcome.bla... 3
web.php
UserController.php
Demo.blade.php
laravel-demo > app > Http > Controllers > UserController.php > ...
9 class UserController extends Controller
24 function adminlogin2(){
33 }
34 function demo(){
35 $Name = "yitayew";
36 $users = ['yitayew','solomon','yared'];
37 return view('Demo',["name"=>$Name,"family"=>$users]);
38 }
39 }
40
41
42
43
44
45
web.php
UserController.php
Demo.blade.php
laravel-demo > resources > views > Demo.blade.php > ...
1 <h1>Welcome {{$name}}</h1>
2 <h1>Users {{print_r($family)}}</h1>
3 <h1>Users {{$family[0]}}</h1>
4

# Cont. …


127.0.0.1:8000/Demo
Welcome yitayew
Users Array ( [0] => yitayew [1] => solomon [2] => yared )
Users yitayew

# Example:

main.blade.php | home2.blade.php | HomeController.php | web.php

laravel-demo > resources > views > layouts > main.blade.php > ...

```
2   <html lang="en">
8   <body>
9       <header>
10          <h1>My Website</h1>
11          <nav>
12              <a href="/">Home</a>
13              <a href="/about">About</a>
14              <a href="/contact">Contact</a>
15          </nav>
16      </header>
17      <main>
18          @yield('content')
19      </main>
20      <footer>
21          <p>&copy; {{ date('Y') }} My Website</p>
22      </footer>
23  </body>
24  </html>
```

Here:

- `@yield('title')` and `@yield('content')` define placeholders for specific sections that can be populated in child templates.

# Cont. ...

main.blade.php | home2.blade.php | HomeController.php | web.php

laravel-demo > resources > views > home2.blade.php > ...

```
1  @extends('layouts.main')
2
3  @section('title', 'Home Page')
4
5  @section('content')
6      <h2>Welcome to My Website!</h2>
7      <p>This is the home page content.</p>
8
9      <p>Welcome, Guest!</p>
10
11     <h3>Our Services</h3>
12     <ul>
13         @foreach($services as $service)
14             <li>{{ $service }}</li>
15         @endforeach
16     </ul>
17 @endsection
```

In this example:

- `@extends('layouts.main')` specifies that this template extends `main.blade.php`.
- `@section('title', 'Home Page')` sets the page title.
- `@section('content')` provides the main content for the `@yield('content')` placeholder.

# Cont. …

main.blade.php | home2.blade.php | HomeController.php | web.php

laravel-demo > app > Http > Controllers > HomeController.php > ...

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class HomeController extends Controller
{
    public function index()
    {
        $services = ['Web Development', 'SEO', 'Consulting'];

        return view('home2', compact('services'));
    }
}
```

# Cont. ...


127.0.0.1:8000/home2
My Website
Home About Contact
Welcome to My Website!
This is the home page content.
Welcome, Guest!
Our Services
Web Development
SEO
Consulting
© 2024 My Website
Route::get('home2',[HomeController::class,'index']);

# Blade Commands in Laravel 11

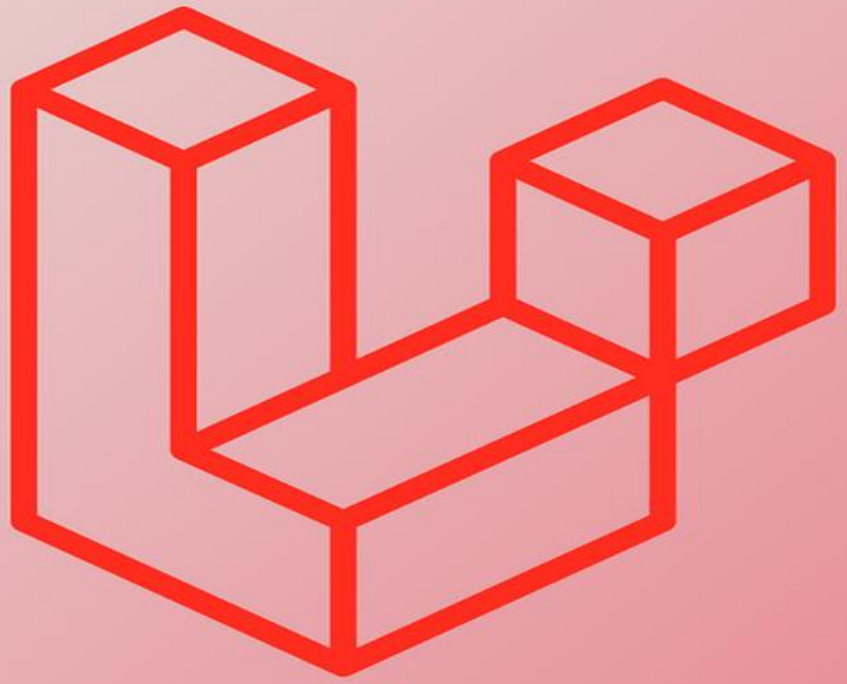
| Category | Blade Command | Description/Example |
|---|---|---|
| **Displaying Data** | `{{ $variable }}` | Echoes variable content with automatic HTML escaping. |
| | `{!! $variable !!}` | Echoes variable content without HTML escaping. |
| **Conditionals** | `@if(condition)` ... `@endif` | Executes code if the condition is true. |
| | `@elseif(condition)` / `@else` | Adds additional conditions or an else block. |
| | `@unless(condition)` ... `@endunless` | Executes code if the condition is false. |
| **Loops** | `@for(init; condition; increment)` | Standard for-loop. |
| | `@foreach($items as $item)` | Loops through a collection of items. |
| | `@forelse($items as $item)` ... `@empty` | Loops through items; shows alternative content if empty. |
| | `@while(condition)` | Executes a while loop. |

# Cont. …

| | | |
|---|---|---|
| Including Files | `@include('view')` | Includes a Blade file. |
| | `@includeIf('view')` | Includes a view only if it exists. |
| | `@includeWhen(condition, 'view')` | Includes a view based on a condition. |
| Layouts & Sections | `@extends('layout')` | Extends a layout file. |
| | `@section('name')` … `@endsection` | Defines a section to inject content. |
| | `@yield('name')` | Displays the content of a section. |
| Components | `<x-component-name />` | Renders a Blade component. |

# Cont. ...

| | | |
|---|---|---|
| **Forms** | `@csrf` | Adds a CSRF token field. |
| | `@method('PUT')` | Specifies an HTTP method (e.g., PUT, DELETE) in a form. |
| | `@error('field')` ... `@enderror` | Displays validation error messages for a specific field. |
| **Authentication** | `@auth` ... `@endauth` | Displays content if the user is authenticated. |
| | `@guest` ... `@endguest` | Displays content if the user is not authenticated. |
| **Miscellaneous** | `@php ... @endphp` | Executes raw PHP code inside Blade. |
| | `@dump($variable)` | Dumps a variable for debugging. |
| | `@switch(condition)` ... `@case` | Implements switch-case logic. |



# Components Laravel 11

# Components in laravel

- In Laravel, components are **reusable pieces of UI elements** designed to improve code organization, reusability, and modularity in Blade templates.
- They allow you to create custom HTML components that can be used throughout your application, making your views more **manageable** and **your code DRY** (Don’t Repeat Yourself).

# Cont. ...

- There are two main types of components in Laravel 11:

❖**Class-Based Components**: Components backed by a PHP class file, which allows you to add logic to the component.

❖**Anonymous Components**: Components without a backing class file, designed for simpler, static components.

# Cont. ...

## Creating a Component in Laravel 11

To create a component, Laravel offers an Artisan command:

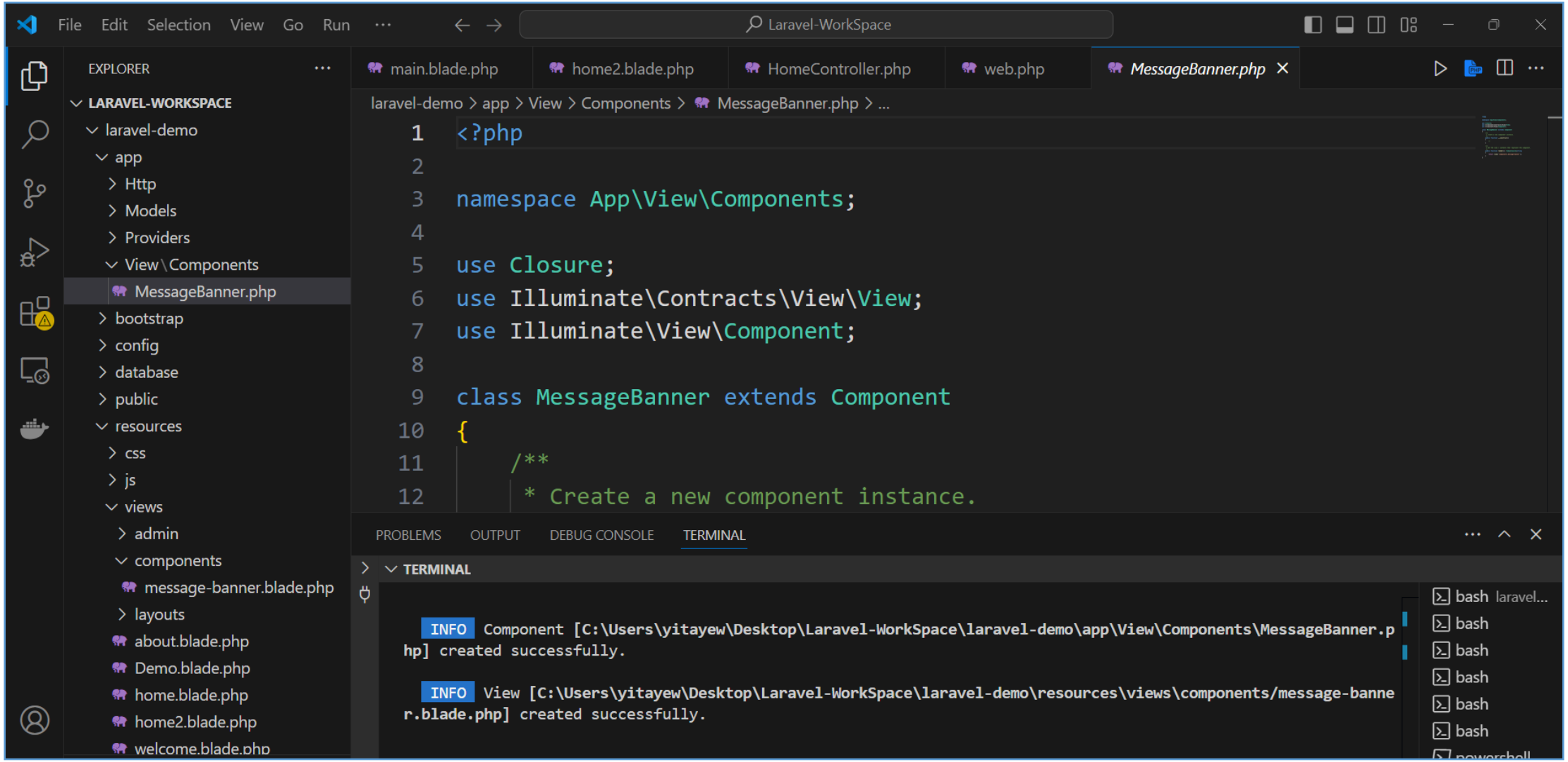
bash Copy code

```
php artisan make:component Alert
```

This will create:

- A class file in `app/View/Components/Alert.php`.
- A Blade view file in `resources/views/components/alert.blade.php`.

# Cont. ...


File Edit Selection View Go Run
Laravel-WorkSpace
EXPLORER
LARAVEL-WORKSPACE
laravel-demo
app
Http
Models
Providers
View\Components
MessageBanner.php
bootstrap
config
database
public
resources
css
js
views
admin
components
message-banner.blade.php
layouts
about.blade.php
Demo.blade.php
home.blade.php
home2.blade.php
welcome.blade.php
main.blade.php
home2.blade.php
HomeController.php
web.php
MessageBanner.php
laravel-demo > app > View > Components > MessageBanner.php > ...
1 <?php
2
3 namespace App\View\Components;
4
5 use Closure;
6 use Illuminate\Contracts\View\View;
7 use Illuminate\View\Component;
8
9 class MessageBanner extends Component
10 {
11 /**
12 * Create a new component instance.
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
TERMINAL
INFO Component [C:\Users\yitayew\Desktop\Laravel-WorkSpace\laravel-demo\app\View\Components\MessageBanner.php] created successfully.
INFO View [C:\Users\yitayew\Desktop\Laravel-WorkSpace\laravel-demo\resources\views\components/message-banner.blade.php] created successfully.
bash laravel...
bash
bash
bash
bash
bash

# How to use the Component

message-banner.blade.php × | Demo.blade.php

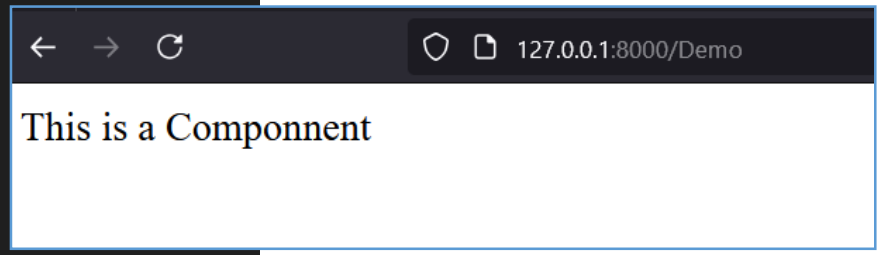
laravel-demo > resources > views > components > message-banner.blade.php > ...

```
1  <div>
2      <span>This is a Componnent</span>
3      <!-- If you do not have a consistent goal in life, you can not live
4  </div>
```

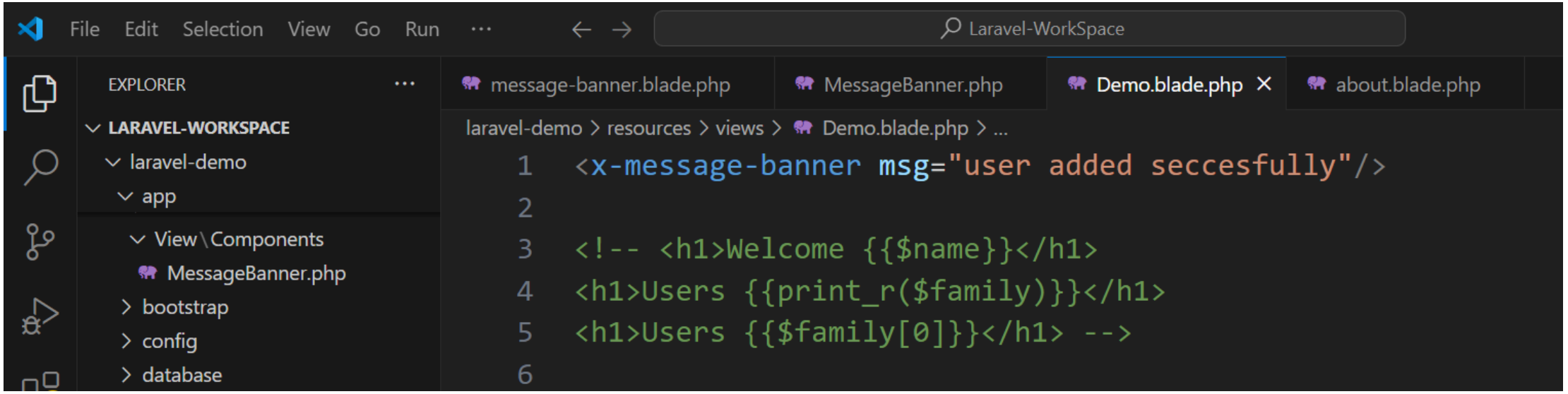
message-banner.blade.php | Demo.blade.php ×

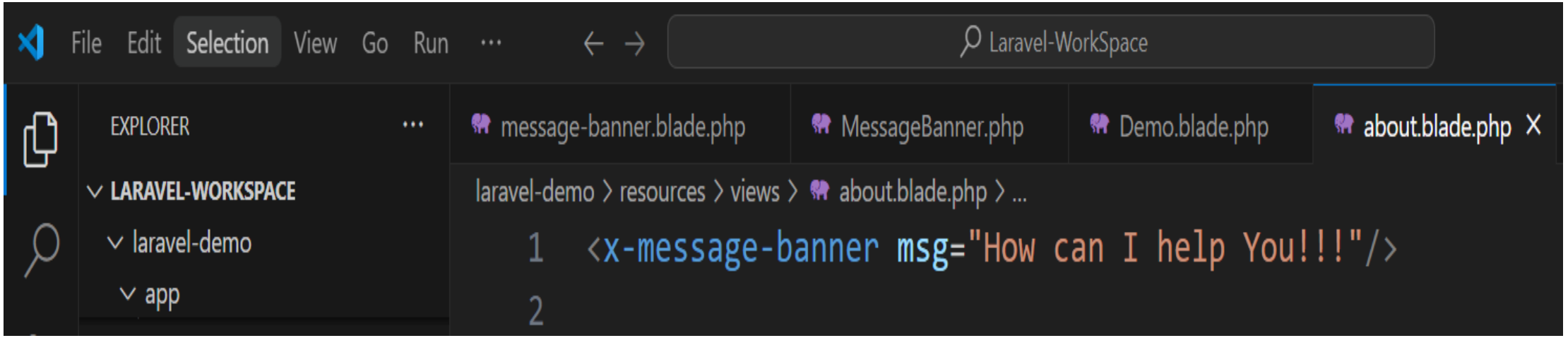
laravel-demo > resources > views > Demo.blade.php > ...

```
1  <x-message-banner/>
2
3  <!-- <h1>Welcome {{$name}}</h1>
4  <h1>Users {{print_r($family)}}</h1>
5  <h1>Users {{$family[0]}}</h1> -->
```

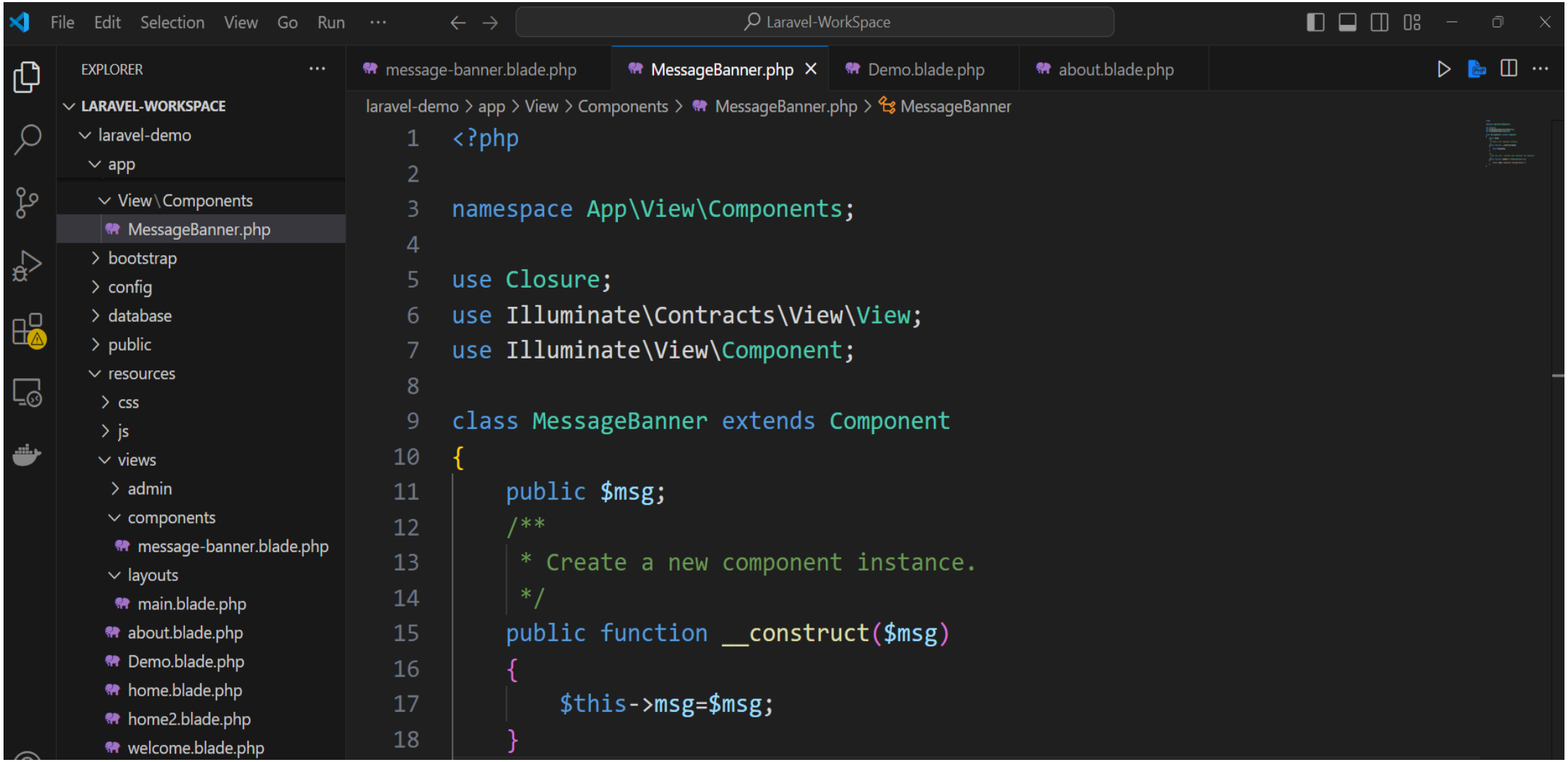


# Dynamic Component

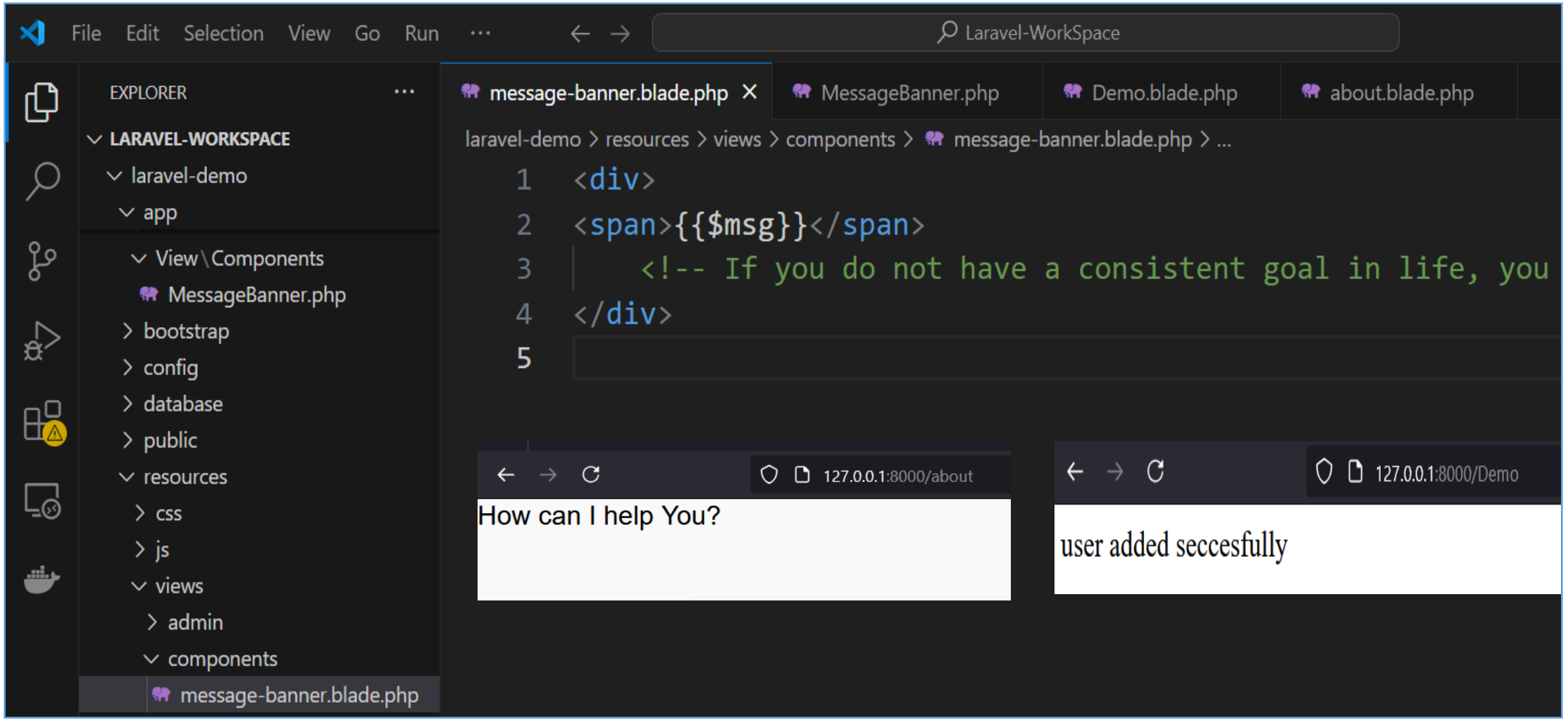


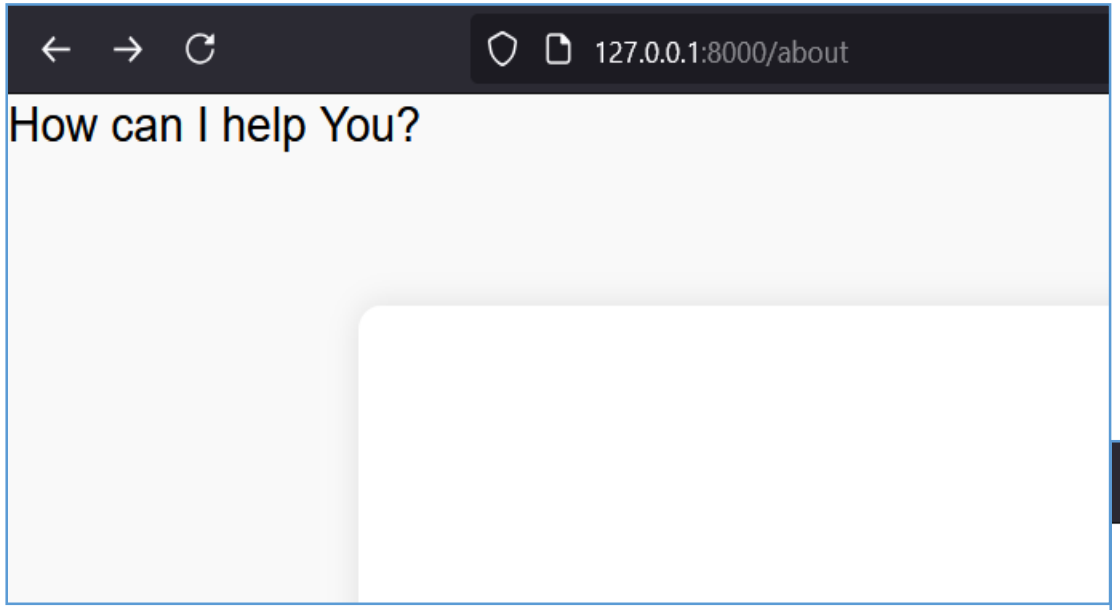
# Cont. ...

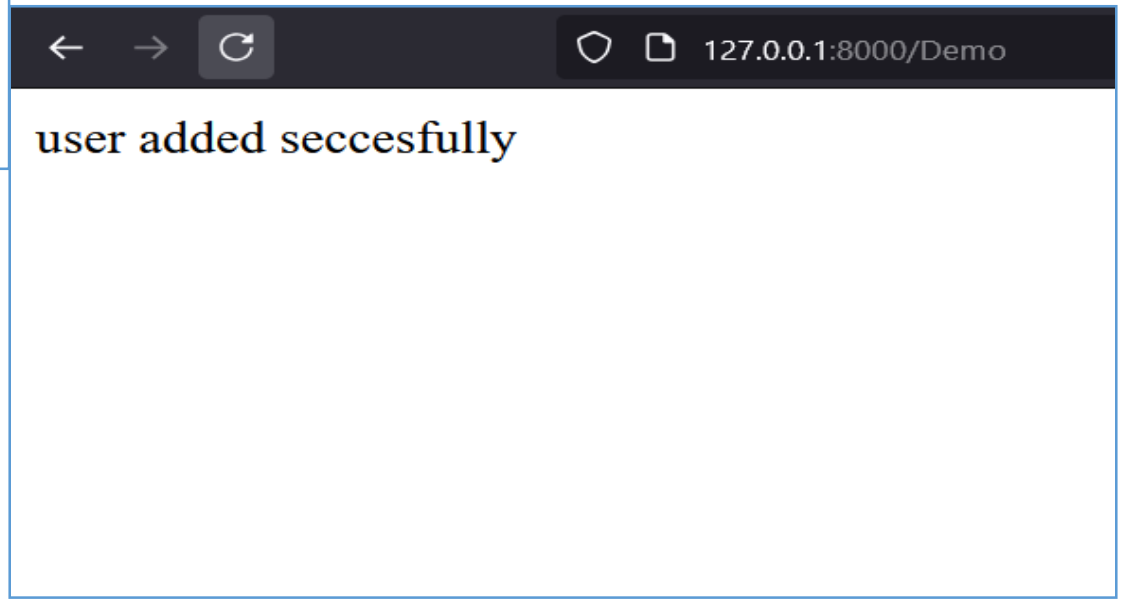

File Edit Selection View Go Run ···
Laravel-WorkSpace
EXPLORER
LARAVEL-WORKSPACE
laravel-demo
app
View\Components
MessageBanner.php
bootstrap
config
database
public
resources
css
js
views
admin
components
message-banner.blade.php
layouts
main.blade.php
about.blade.php
Demo.blade.php
home.blade.php
home2.blade.php
welcome.blade.php
message-banner.blade.php
MessageBanner.php
Demo.blade.php
about.blade.php
laravel-demo > app > View > Components > MessageBanner.php > MessageBanner
1 <?php
2
3 namespace App\View\Components;
4
5 use Closure;
6 use Illuminate\Contracts\View\View;
7 use Illuminate\View\Component;
8
9 class MessageBanner extends Component
10 {
11 public $msg;
12 /**
13 * Create a new component instance.
14 */
15 public function __construct($msg)
16 {
17 $this->msg=$msg;
18 }

# Cont. ...

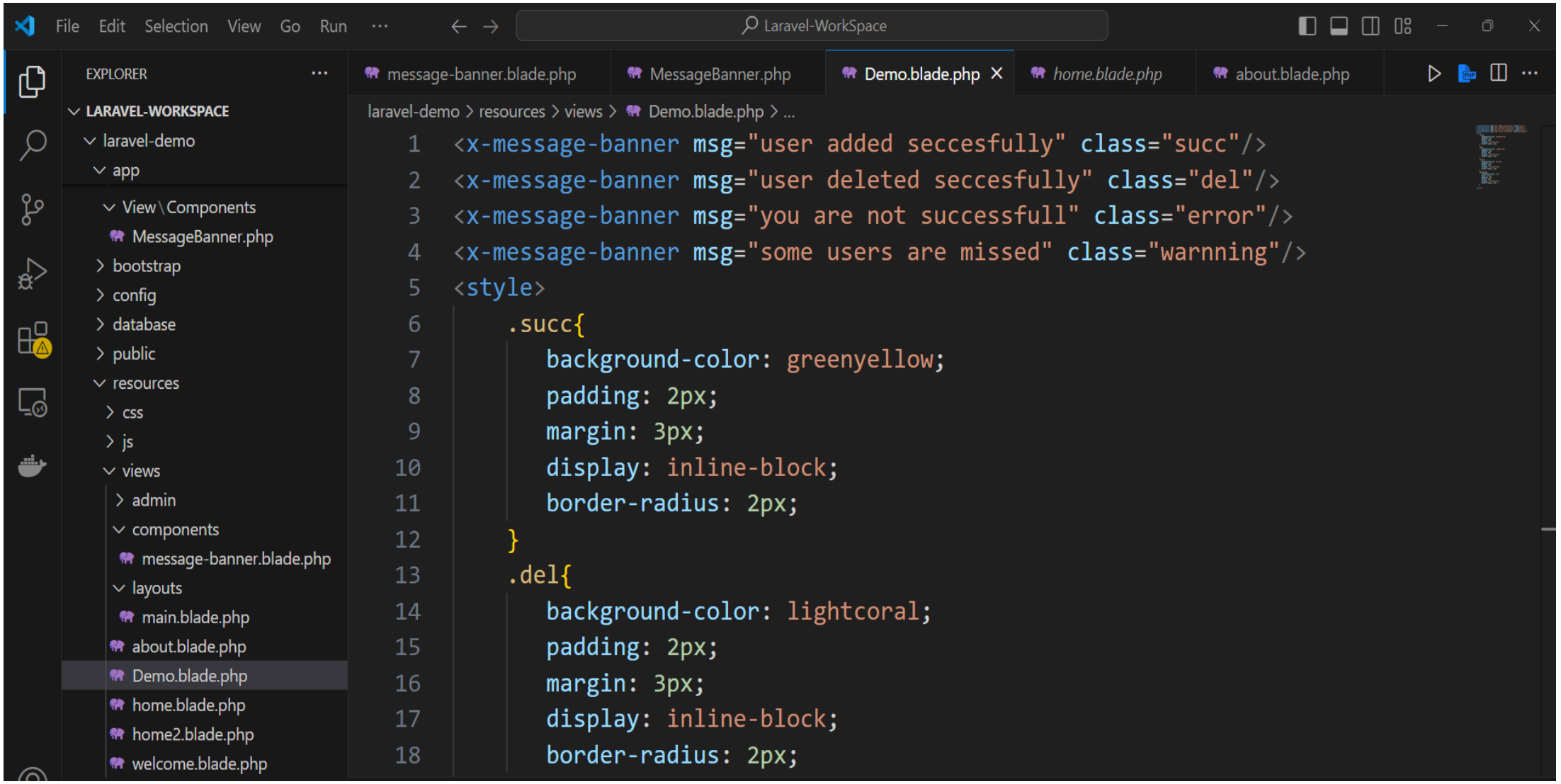

File Edit Selection View Go Run ···
Laravel-WorkSpace
EXPLORER
LARAVEL-WORKSPACE
laravel-demo
app
View\Components
MessageBanner.php
bootstrap
config
database
public
resources
css
js
views
admin
components
message-banner.blade.php
message-banner.blade.php
MessageBanner.php
Demo.blade.php
about.blade.php
laravel-demo > resources > views > components > message-banner.blade.php > ...
1 
2 {{$msg}}
3 <!-- If you do not have a consistent goal in life, you
4 
5
127.0.0.1:8000/about
How can I help You?
127.0.0.1:8000/Demo
user added seccesfully

# Cont. ...

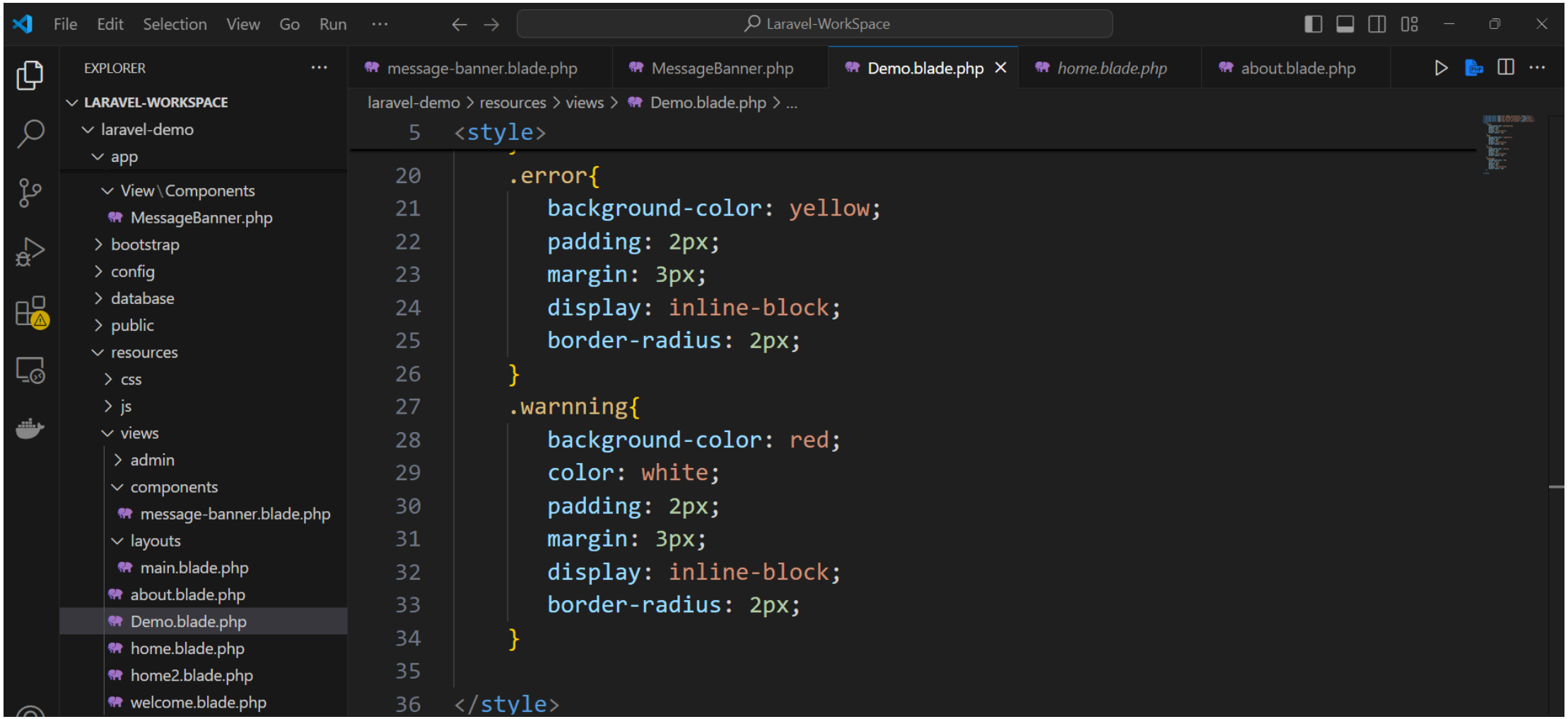

127.0.0.1:8000/about
How can I help You?


127.0.0.1:8000/Demo
user added seccesfully

# Applying Common Style



# Cont. ...


File Edit Selection View Go Run
Laravel-WorkSpace
EXPLORER
LARAVEL-WORKSPACE
laravel-demo
app
View \ Components
MessageBanner.php
bootstrap
config
database
public
resources
css
js
views
admin
components
message-banner.blade.php
layouts
main.blade.php
about.blade.php
Demo.blade.php
home.blade.php
home2.blade.php
welcome.blade.php
message-banner.blade.php
MessageBanner.php
Demo.blade.php
home.blade.php
about.blade.php
laravel-demo > resources > views > Demo.blade.php > ...
5 <style>
20 .error{
21 background-color: yellow;
22 padding: 2px;
23 margin: 3px;
24 display: inline-block;
25 border-radius: 2px;
26 }
27 .warnning{
28 background-color: red;
29 color: white;
30 padding: 2px;
31 margin: 3px;
32 display: inline-block;
33 border-radius: 2px;
34 }
35
36 </style>

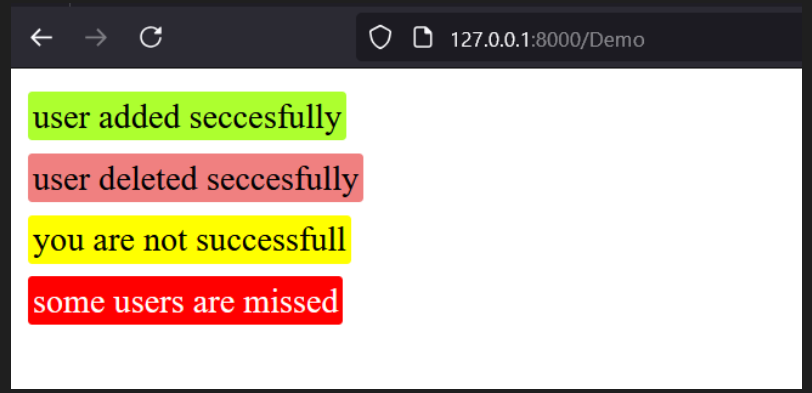
# Cont. ...

message-banner.blade.php | MessageBanner.php | Demo.blade.php | home.blade.php | about.blade.php

laravel-demo > app > View > Components > MessageBanner.php > MessageBanner

```php
use Closure;
use Illuminate\Contracts\View\View;
use Illuminate\View\Component;

class MessageBanner extends Component
{
    public $msg;
    public $class;
    /**
     * Create a new component instance.
     */
    public function __construct($msg, $class)
    {
        $this->msg=$msg;
        $this -> class = $class;
    }
```



# Thank you!

Appreciate your action.