



# Laravel



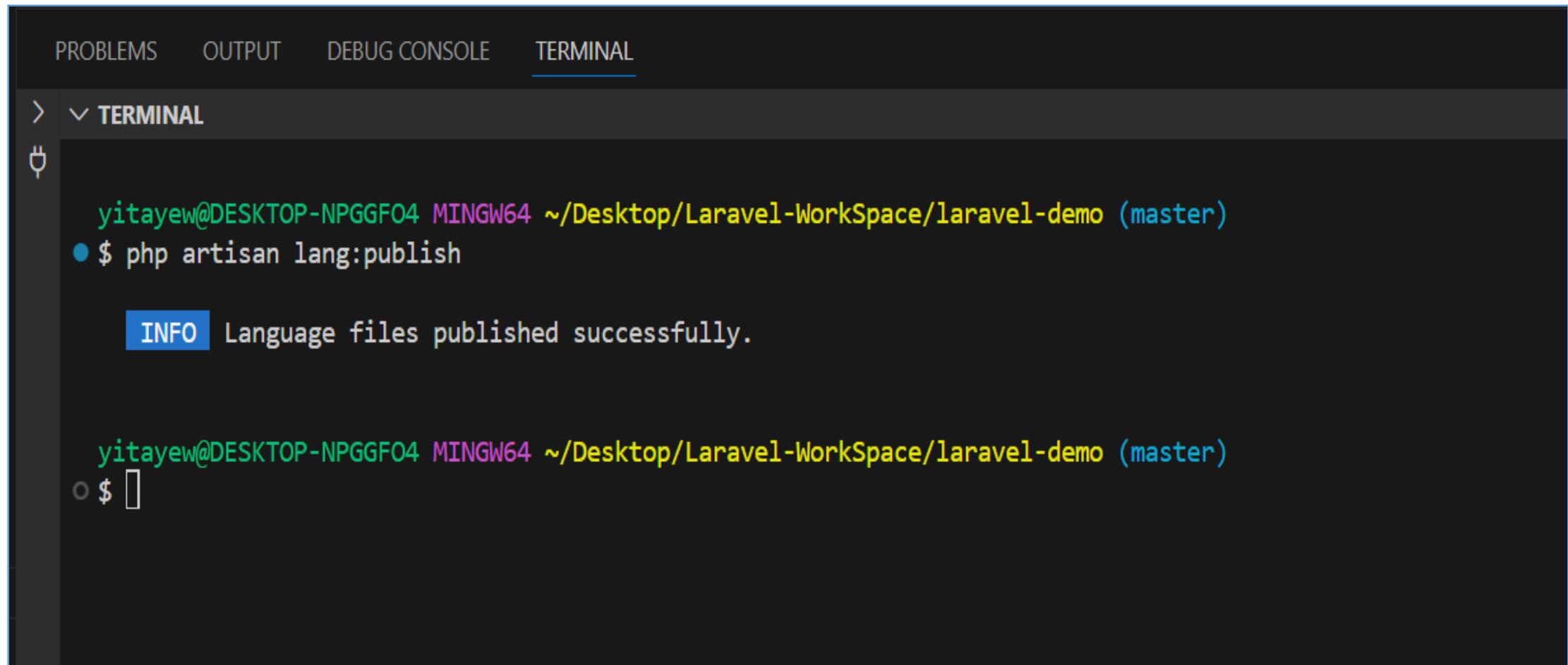
# Outlines of Discussion

- ❖ Localization In Laravel
- ❖ CRUD Operation using HTML Form
- ❖ Pagination in Laravel
- ❖ Stub in Laravel
- ❖ Migration In Laravel
- ❖ Seeding in laravel

# Localization In Laravel

- Localization in Laravel makes it easy to support multiple languages in your application. This is useful for applications that need to serve users in different regions or for those who prefer to interact in their native languages.
- Laravel provides a convenient way to manage language files and handle translations throughout your application.

# Publishing Lang folder



The screenshot shows a terminal window with the following interface elements:

- Top bar with tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (underlined).
- Section header: > ▾ TERMINAL.
- Icon: A small terminal icon.
- User information: yitayew@DESKTOP-NPGGF04 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo (master)
- Command: \$ php artisan lang:publish
- Output message: INFO Language files published successfully.
- User information: yitayew@DESKTOP-NPGGF04 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo (master)
- Command prompt: \$ [ ]

The terminal window has a dark background with light-colored text. The output messages are color-coded: user information in green, the command in blue, and the output message in cyan.

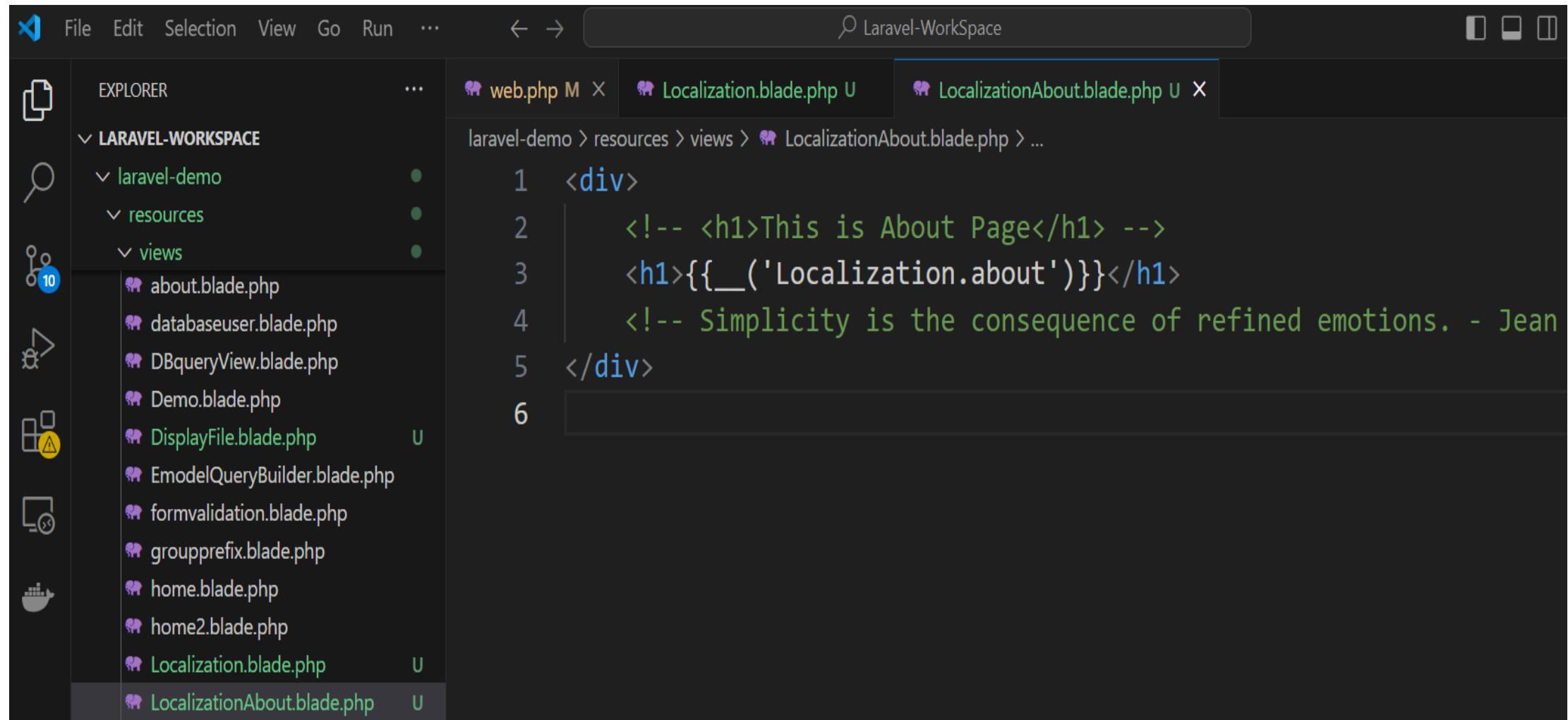
# Example (View-Localization)

The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists a 'Laravel-Workspace' folder containing a 'laravel-demo' project. Inside 'laravel-demo', there are 'resources' and 'views' folders. The 'views' folder contains several Blade files: about.blade.php, databaseuser.blade.php, DBqueryView.blade.php, Demo.blade.php, DisplayFile.blade.php (marked with a 'U' icon), EmodelQueryBuilder.blade.php, formvalidation.blade.php, groupprefix.blade.php, home.blade.php, home2.blade.php, Localization.blade.php (marked with a 'U' icon and highlighted in blue), LocalizationAbout.blade.php (marked with a 'U' icon), middleware.blade.php, middleware2.blade.php, middleware3.blade.php, middleware4.blade.php, middleware5.blade.php, and namedroute.blade.php.

The main editor area shows the content of the 'Localization.blade.php' file. The code uses PHP-like syntax within HTML tags to perform localization. It includes comments for sections like 'How Are You' and 'Well Come', and a quote from Thich Nhat Hanh. The file ends with links to 'LocalizationAbout', 'Contactus', and 'Home' pages.

```
<div>
    <!-- &lt;h1&gt;How Are You&lt;/h1&gt; --&gt;
    &lt;h1&gt;{{__('Localization.localization')}}&lt;/h1&gt;
    &lt;br&gt;
    <!-- &lt;h1&gt;Well Come&lt;/h1&gt; --&gt;
    &lt;h1&gt;{{__('Localization.heading')}}&lt;/h1&gt;
    &lt;!-- Walk as if you are kissing the Earth with your feet. - Thich Nhat Hanh --&gt;
&lt;/div&gt;
&lt;a href="/LocalizationAbout"&gt;{{__('Localization.about')}}&lt;/a&gt;
&lt;br&gt;
&lt;a href="Contactus"&gt;{{__('Localization.contactus')}}&lt;/a&gt;
&lt;br&gt;
&lt;a href="Home"&gt;{{__('Localization.home')}}&lt;/a&gt;</pre>
```

# View (LocalizationAbout)



The screenshot shows a code editor interface with a dark theme. The top bar includes icons for File, Edit, Selection, View, Go, Run, and a three-dot menu. A search bar displays "Laravel-WorkSpace". The left sidebar has icons for Explorer, Search (with 10 results), Issues, and a warning sign. The Explorer panel shows a project structure under "LARAVEL-WORKSPACE": "laravel-demo" > "resources" > "views". Inside "views", there are several blade files: "about.blade.php", "databaseuser.blade.php", "DBqueryView.blade.php", "Demo.blade.php", "DisplayFile.blade.php", "EmodelQueryBuilder.blade.php", "formvalidation.blade.php", "groupprefix.blade.php", "home.blade.php", "home2.blade.php", "Localization.blade.php", and "LocalizationAbout.blade.php". The "LocalizationAbout.blade.php" file is currently selected and open in the main editor area. The code in the editor is:

```
1 <div>
2     <!-- <h1>This is About Page</h1> -->
3     <h1>{{__('Localization.about')}}</h1>
4     <!-- Simplicity is the consequence of refined emotions. - Jean
5 </div>
6 
```

# Localization(Eng)

The screenshot shows a dark-themed interface of Visual Studio Code (VS Code) with an orange header bar. The title bar reads "Laravel-WorkSpace". The left sidebar is the "EXPLORER" view, showing the project structure under "LARAVEL-WORKSPACE". The "laravel-demo" folder contains "app", "bootstrap", "config", "database", "lang" (with subfolders "Amh" and "en"), and several PHP files like "auth.php", "Localization.php" (which is currently selected), "pagination.php", "passwords.php", and "validation.php". The "en" folder under "lang" contains "Localization.php". The main editor area shows the content of the "Localization.php" file in the "en" directory:

```
1 <?php
2 return [
3     'localization'=>'How are You...',
4     'heading'=>'Well Come ...',
5     'about'=>'About',
6     'contactus'=>'Contactus',
7     'home'=>'Home'
8 ]
9 ?>
10
```

# Localization (Ahm)

The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists the project structure under 'LARAVEL-WORKSPACE'. The 'lang' directory contains 'Amh' and 'en' sub-directories. 'Localization.php' is selected in the 'Amh' directory. The main editor area displays the contents of 'Localization.php' for the Amharic language (Amh). The file contains PHP code defining an array of localized strings:

```
1 <?php
2 return [
3     'localization'=>'ሰላም!!!',
4     'heading'=>'እንከን ደህን መጠበቅ!!!',
5     'about'=>'አለ',
6     'contactus'=>'ፖ.ስተኞች',
7     'home'=>'መነሻ ገጽ'
8 ]
9 ?>
10
11
```

# Localization (Oro)

The screenshot shows a code editor interface with a dark theme. The top bar includes icons for File, Edit, Selection, View, Go, Run, and a three-dot menu. The title bar says "Laravel-WorkSpace". The left sidebar has icons for Explorer, Search, Problems (with 10 items), and Diff. The Explorer panel shows a file tree under "LARAVEL-WORKSPACE" named "laravel-demo". It contains "app", "bootstrap", "config", "database", "lang" (which has "Amh" and "en" subfolders), and "Oro" (which has "Localization.php" files). The main editor area displays the contents of "Localization.php" for the "en" locale. The code is as follows:

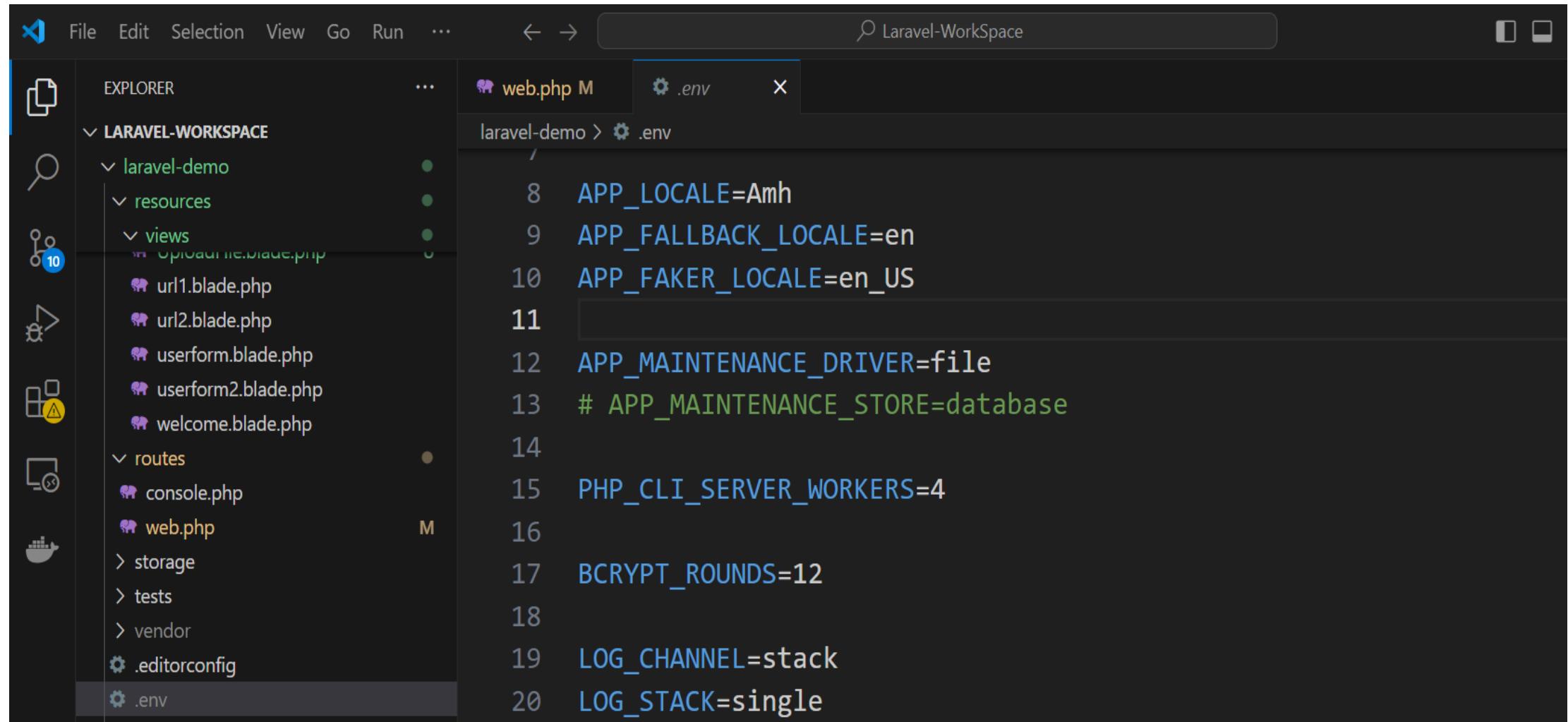
```
1 <?php
2 return [
3     'localization'=>'Akkam jirta',
4     'heading'=>"Anaa dhufuu",
5     'about'=>"Waa 'ee",
6     'contactus'=>"Contactus",
7     'home'=>'fuula jalqabaa'
8 ]
9 ?>
```

# Route

The screenshot shows the Visual Studio Code interface with a dark theme. The left sidebar is the Explorer, displaying the project structure under 'Laravel-Workspace'. The 'routes' folder contains 'console.php' and 'web.php'. The 'web.php' file is currently open in the main editor area. The code in 'web.php' is as follows:

```
// Localization Route
Route::view('localization','localization');
Route::view('LocalizationAbout','LocalizationAbout');
```

# Config on (.env folder)



The screenshot shows the Visual Studio Code interface with the title bar "Laravel-WorkSpace". The left sidebar displays a file tree for a Laravel workspace named "laravel-demo". The "resources/views" folder contains several Blade files: "uploadfile.blade.php", "url1.blade.php", "url2.blade.php", "userform.blade.php", "userform2.blade.php", and "welcome.blade.php". The "routes" folder contains "console.php" and "web.php". The ".env" file is open in the main editor area, showing environment variable configurations:

```
APP_LOCALE=Amh
APP_FALLBACK_LOCALE=en
APP_FAKE_LOCALE=en_US
APP_MAINTENANCE_DRIVER=file
# APP_MAINTENANCE_STORE=database
PHP_CLI_SERVER_WORKERS=4
BCRYPT_ROUNDS=12
LOG_CHANNEL=stack
LOG_STACK=single
```

# Output

A screenshot of a web browser window titled "127.0.0.1:8000/localization". The page content is in Amharic:

**ሰላም!!!**  
**እንኩን ደህን መጣሁ!!!**

Navigation links at the bottom:

- [አሉ](#)
- [ዶግታን](#)
- [መነሻ ገጽ](#)

A screenshot of a web browser window titled "127.0.0.1:8000/localization". The page content is in English:

**Akkam jirta**  
**Anaa dhufuu**

Navigation links at the bottom:

- [Waa'ee](#)
- [Contactus](#)
- [fuula jalqabaa](#)

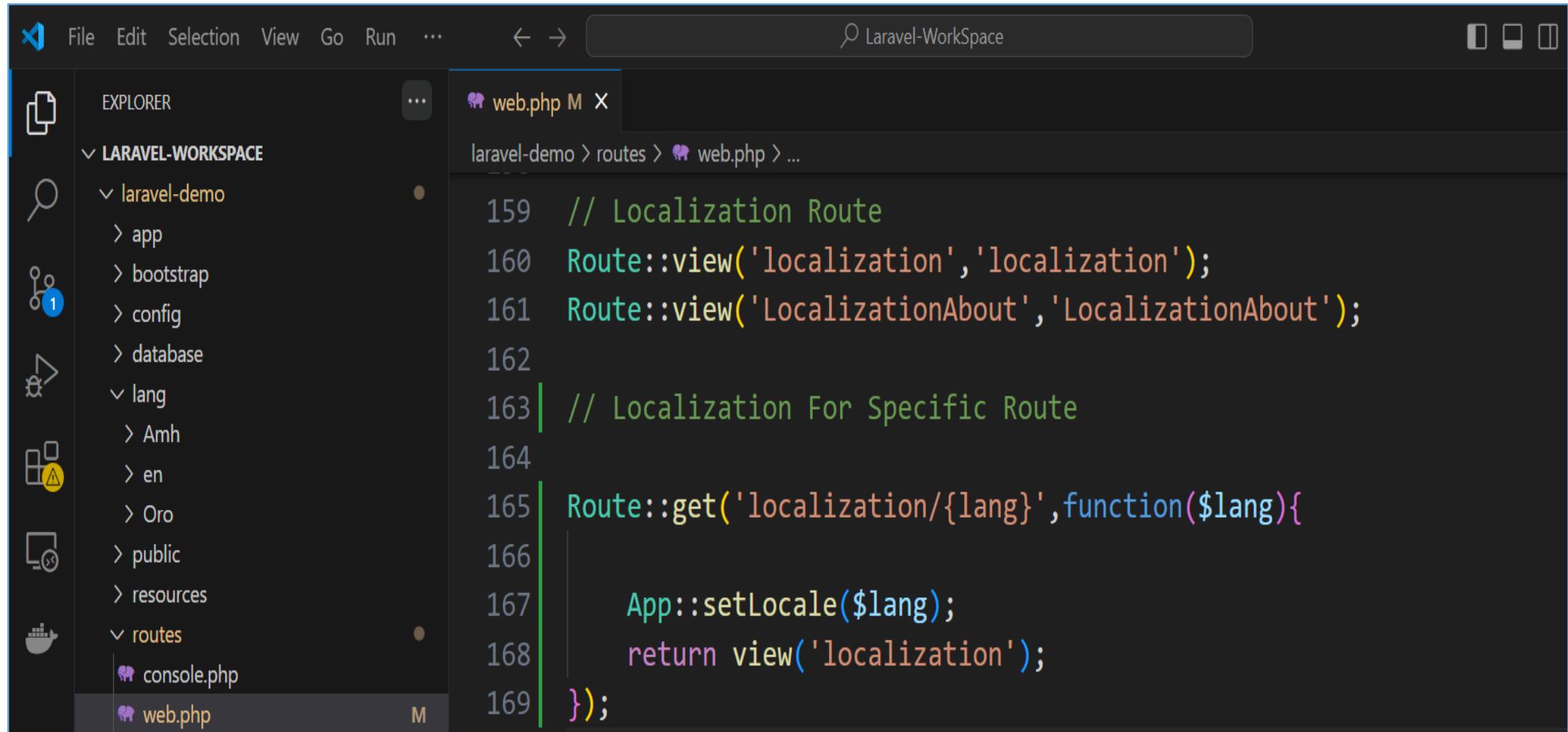
A screenshot of a web browser window titled "127.0.0.1:8000/localization". The page content is in English:

**How are You...**  
**Well Come ...**

Navigation links at the bottom:

- [About](#)
- [Contactus](#)
- [Home](#)

# Localization For Specific Route

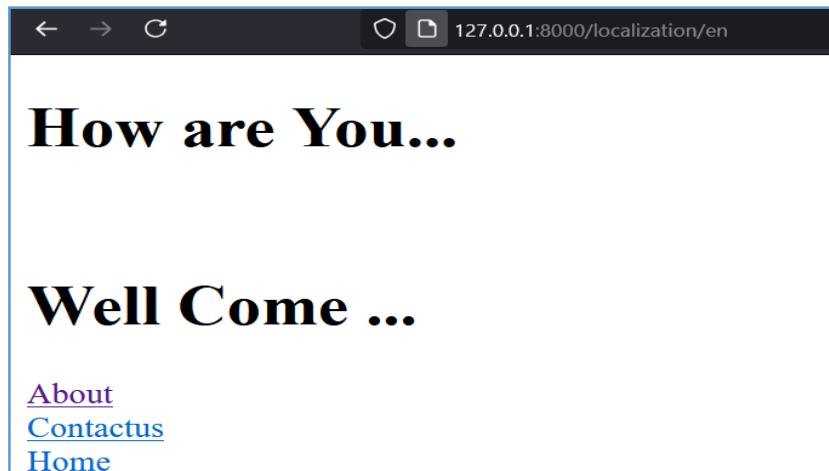
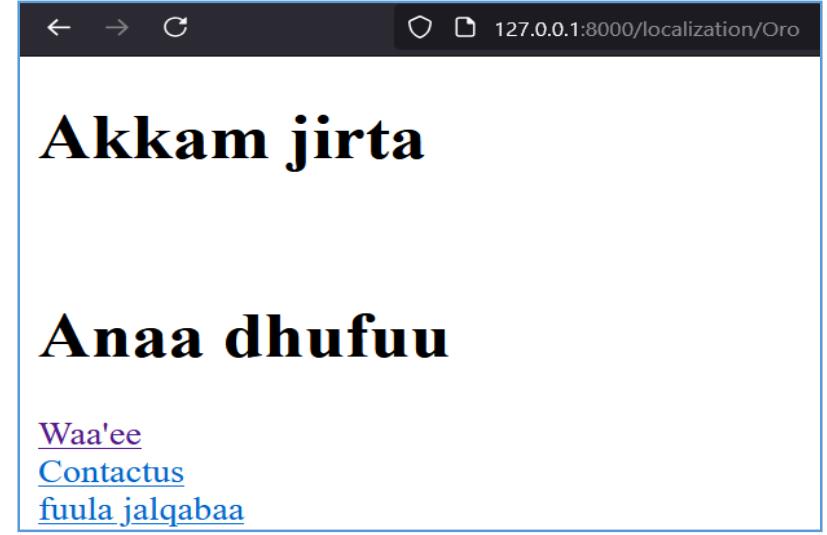
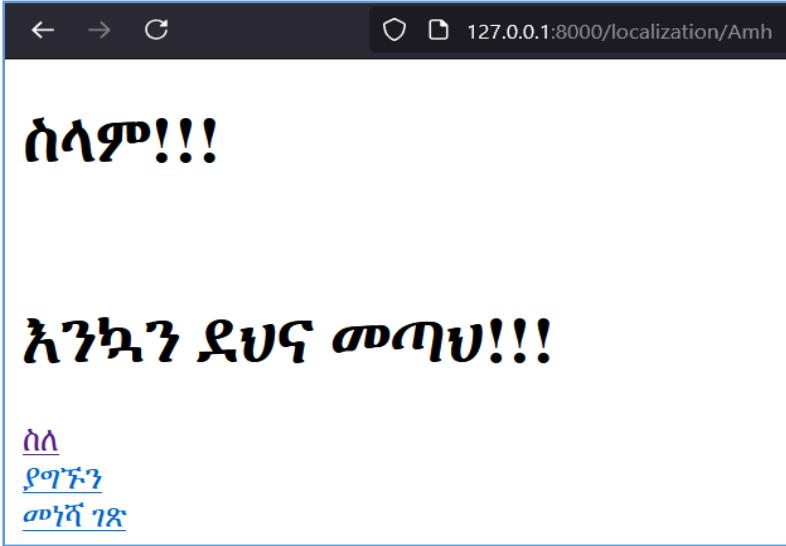


The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists the project structure under 'Laravel-Workspace'. The 'routes' folder contains 'console.php' and 'web.php'. The 'web.php' file is currently open in the main editor area. The code in 'web.php' is as follows:

```
159 // Localization Route
160 Route::view('localization','localization');
161 Route::view('LocalizationAbout','LocalizationAbout');
162
163 // Localization For Specific Route
164
165 Route::get('localization/{lang}',function($lang){
166
167     App::setLocale($lang);
168
169 });

```

# Cont. ...



# **CRUD Operation (Insert Data in MySQL Table With HTML form)**

- Inserting data into a MySQL table through an HTML form in Laravel 11 involves several steps, including setting up the route, creating a controller method, defining the form view, and creating the database model. Here's a guide on how to accomplish this.

# Creating (Model, View, Controller)

```
> ▾ TERMINAL
yitayew@DESKTOP-NPGGF04 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo (master)
● $ php artisan make:model employee

[INFO] Model [C:\Users\yitayew\Desktop\Laravel-WorkSpace\laravel-demo\app\Models\employee.php] created successfully.

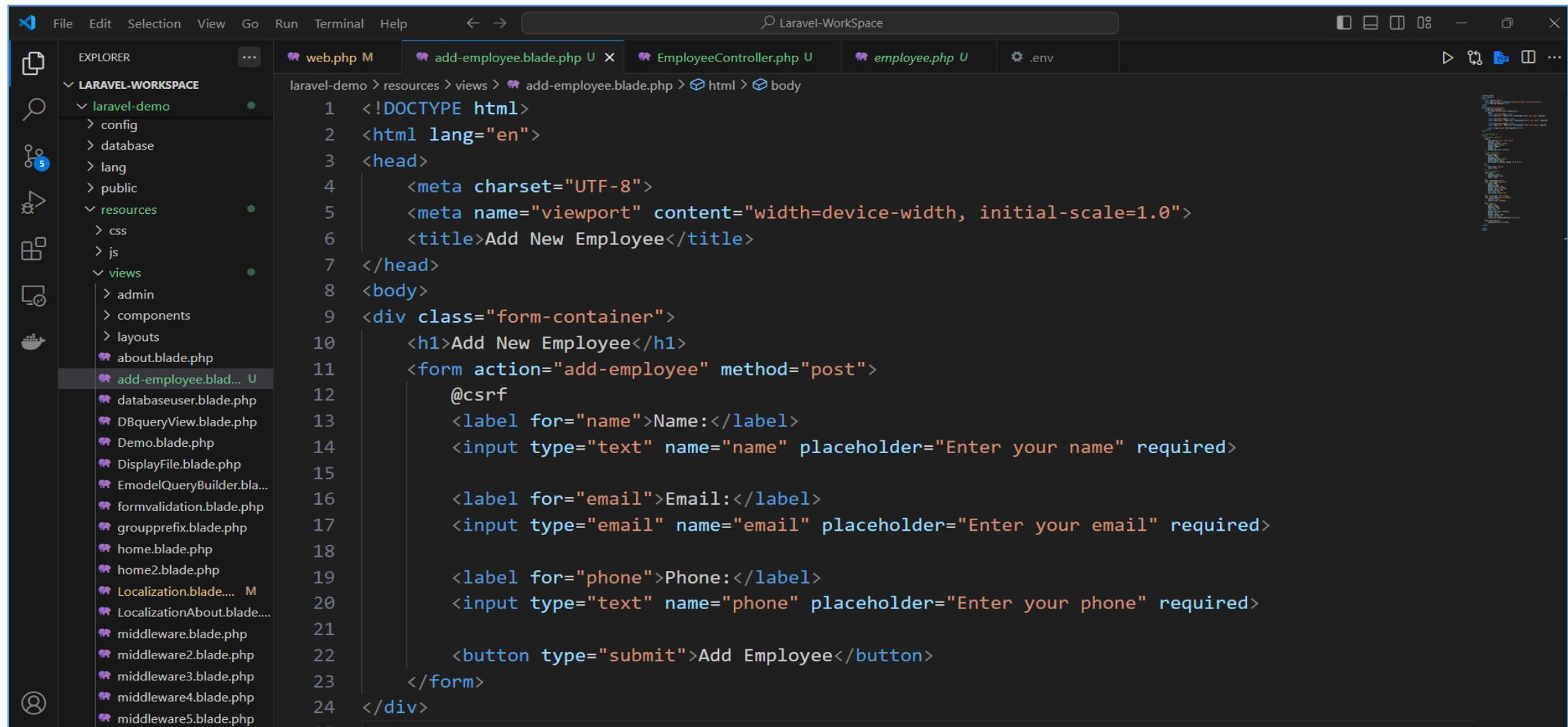
yitayew@DESKTOP-NPGGF04 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo (master)
● $ php artisan make:view add-employee

[INFO] View [C:\Users\yitayew\Desktop\Laravel-WorkSpace\laravel-demo\resources\views\add-employee.blade.php] created successfully.

yitayew@DESKTOP-NPGGF04 MINGW64 ~/Desktop/Laravel-WorkSpace/laravel-demo (master)
● $ php artisan make:controller EmployeeController

[INFO] Controller [C:\Users\yitayew\Desktop\Laravel-WorkSpace\laravel-demo\app\Http\Controllers\EmployeeController.php] created successfully.
```

# View (add-employee)



The screenshot shows a code editor interface with the title "Laravel-WorkSpace". The left sidebar displays the "EXPLORER" view of the "LARAVEL-WORKSPACE" named "laravel-demo". The "views" folder contains several files, including "add-employee.blade.php", which is currently selected and shown in the main editor area.

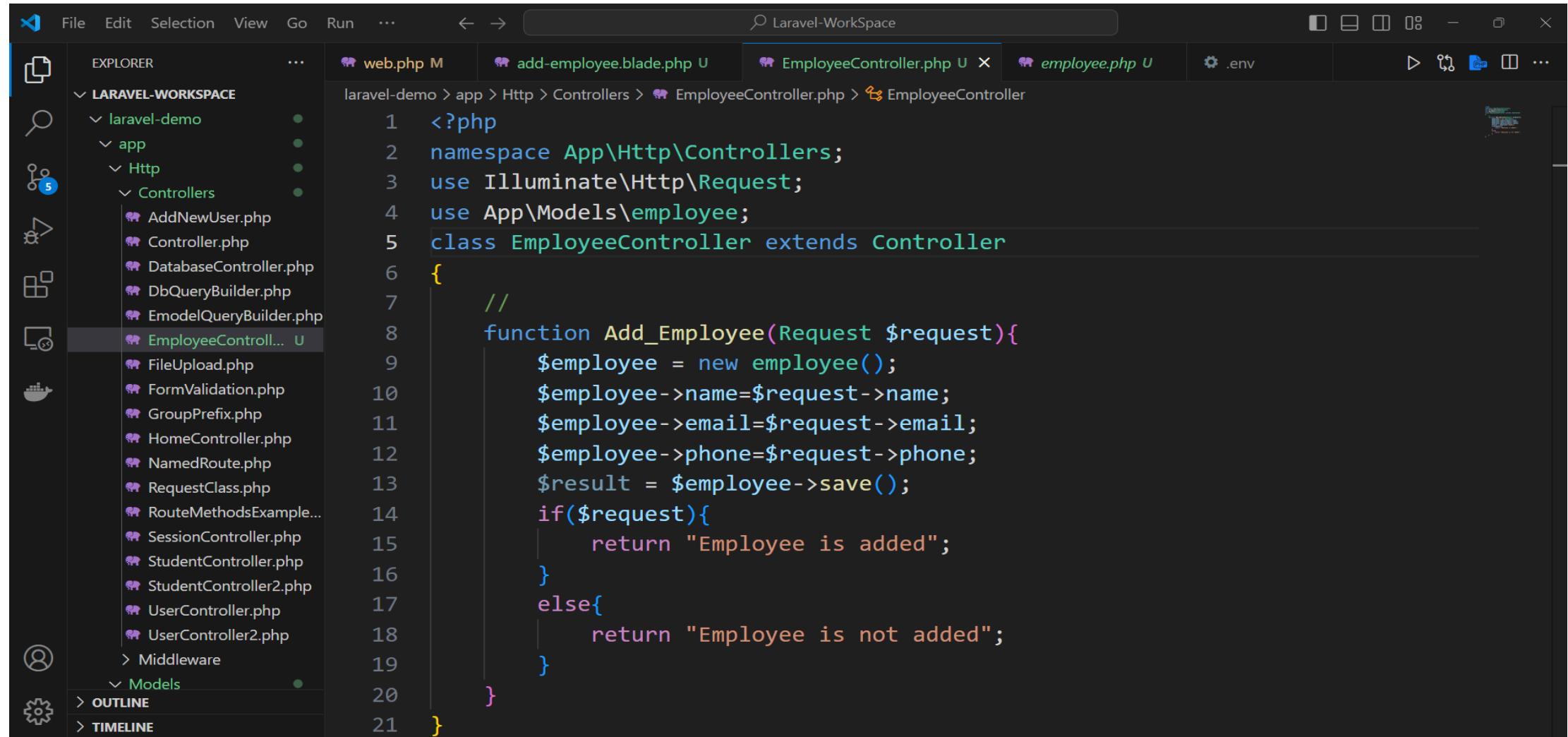
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Add New Employee</title>
</head>
<body>
<div class="form-container">
    <h1>Add New Employee</h1>
    <form action="add-employee" method="post">
        @csrf
        <label for="name">Name:</label>
        <input type="text" name="name" placeholder="Enter your name" required>

        <label for="email">Email:</label>
        <input type="email" name="email" placeholder="Enter your email" required>

        <label for="phone">Phone:</label>
        <input type="text" name="phone" placeholder="Enter your phone" required>

        <button type="submit">Add Employee</button>
    </form>
</div>
```

# Controller (EmployeeController)



The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists the project structure under 'LARAVEL-WORKSPACE' and 'laravel-demo'. The 'Controllers' folder contains several files, with 'EmployeeController.php' currently selected and highlighted in blue. The main editor area displays the PHP code for the EmployeeController:

```
1 <?php
2 namespace App\Http\Controllers;
3 use Illuminate\Http\Request;
4 use App\Models\employee;
5 class EmployeeController extends Controller
6 {
7     //
8     function Add_Employee(Request $request){
9         $employee = new employee();
10        $employee->name=$request->name;
11        $employee->email=$request->email;
12        $employee->phone=$request->phone;
13        $result = $employee->save();
14        if($result){
15            return "Employee is added";
16        }
17        else{
18            return "Employee is not added";
19        }
20    }
21 }
```

# Model (employee)

The screenshot shows the Visual Studio Code interface with the title bar "Laravel-WorkSpace". The left sidebar is the Explorer view, showing the project structure under "LARAVEL-WORKSPACE". The "Models" folder contains three files: "employee.php", "Student.php", and "user.php", with "employee.php" currently selected. The main editor area displays the code for the "employee.php" model:

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Model;
6
7 class employee extends Model
8 {
9     //mention your table name for modification
10    // protected $table = "your table name";
11 }
12
```

# Route

The screenshot shows a code editor interface for a Laravel application named "laravel-demo". The left sidebar displays the project structure under "LARAVEL-WORKSPACE". The main editor area shows the content of the "web.php" file.

```
167 // Inserting Data Using Form
168 Route::view('add-employee', 'add-employee');
169
170 // Importing Controller
171 use App\Http\Controllers\EmployeeController;
172 Route::post('add-employee', [EmployeeController::class, 'Add_Employee']);
173
```

The "routes" folder in the project structure contains the "web.php" file, which is currently selected in the sidebar.

# Output

A screenshot of a web browser window displaying a form titled "Add New Employee". The form includes fields for Name, Email, and Phone, each with a placeholder text "Enter your name", "Enter your email", and "Enter your phone" respectively. A blue button at the bottom is labeled "Add Employee". The browser's address bar shows the URL "127.0.0.1:8000/add-employee".

← → ⌛ 127.0.0.1:8000/add-employee 140% ☆ ⌂ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌊ ⌋ ⌃

## Add New Employee

Name:

Email:

Phone:

Add Employee

# Cont. ...

Showing rows 0 - 3 (4 total, Query took 0.0002 seconds.)

```
SELECT * FROM `employees`
```

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

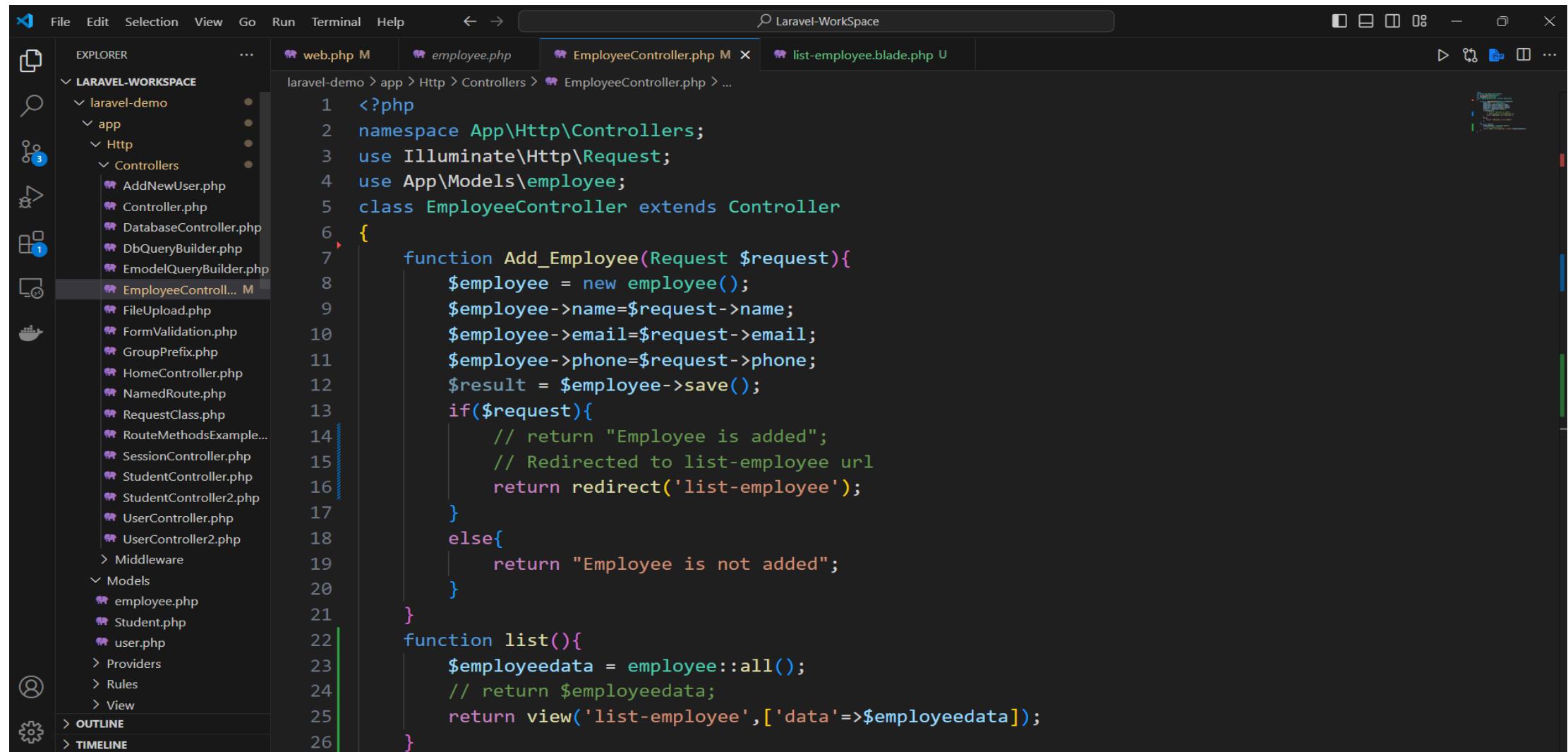
Show all | Number of rows: 25 ▾ Filter rows: Search this table Sort by key: None ▾

Extra options

← T →		id	name	email	phone	created_at	updated_at	
<input type="checkbox"/>	 Edit	 Copy	 Delete	1 Yitayew Solomon	yitayewsolomon3@gmail.com	0937559718	2024-11-12	2024-11-12
<input type="checkbox"/>	 Edit	 Copy	 Delete	2 Haileab Solomon	haileabsolomon@gmail.com	0937559718	2024-11-12	2024-11-12
<input type="checkbox"/>	 Edit	 Copy	 Delete	3 Yared Solomon	yaredsolomon@gmail.com	0937559718	2024-11-12	2024-11-12
<input type="checkbox"/>	 Edit	 Copy	 Delete	6 Natanim Yitayew	natan@gmail.com	0937559720	2024-11-12	2024-11-12

↑  Check all With selected:  Edit  Copy  Delete  Export

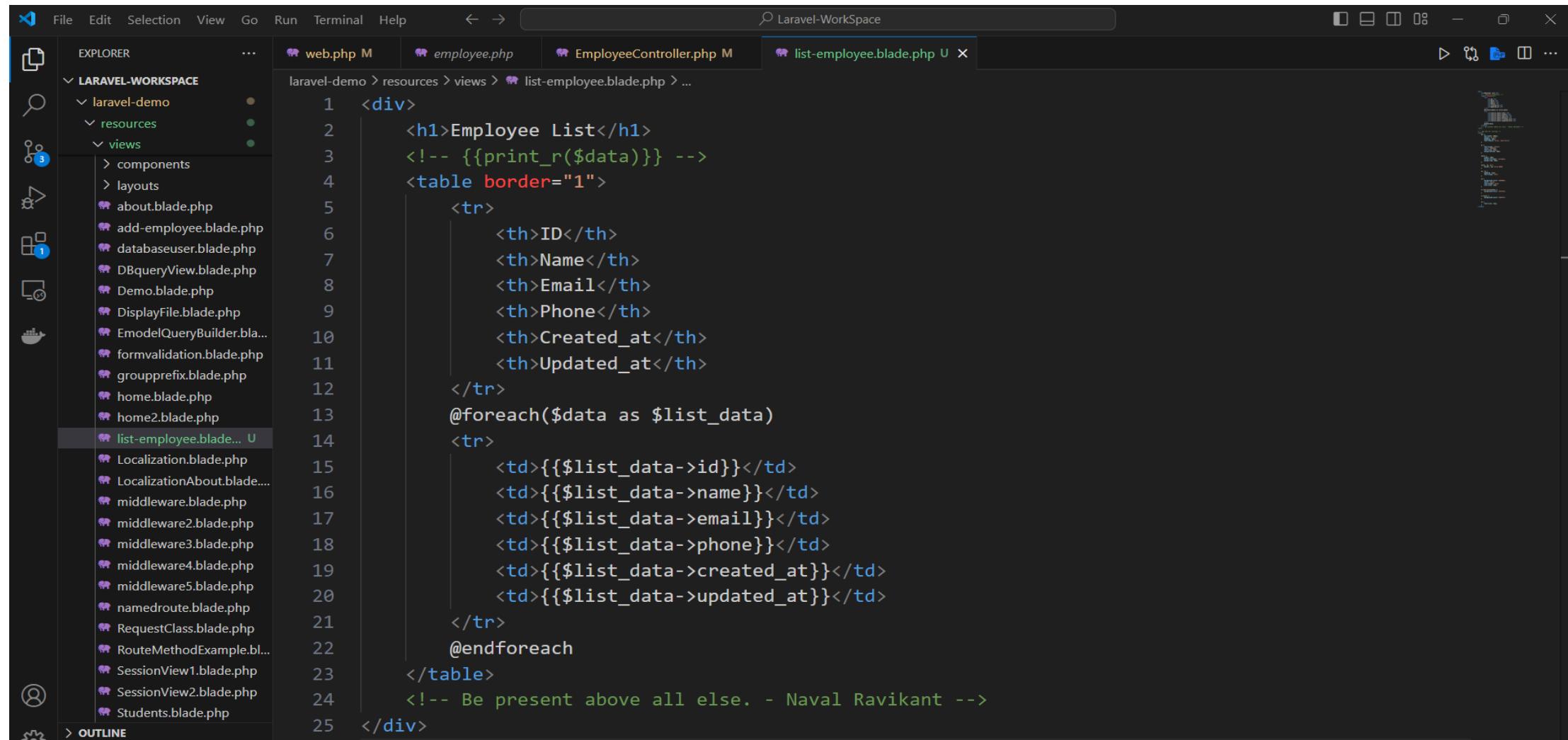
# Get and Display Data From MySql Table



The screenshot shows a code editor interface with a dark theme. The title bar reads "Laravel-WorkSpace". The left sidebar has icons for File, Edit, Selection, View, Go, Run, Terminal, Help, Explorer, Search, Problems (with 3 items), Files (with 1 item), Outline, and Timeline. The Explorer panel shows a project structure under "LARAVEL-WORKSPACE": laravel-demo > app > Http > Controllers > EmployeeController.php. The main editor area displays the following PHP code:

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Models\employee;
class EmployeeController extends Controller
{
    function Add_Employee(Request $request){
        $employee = new employee();
        $employee->name=$request->name;
        $employee->email=$request->email;
        $employee->phone=$request->phone;
        $result = $employee->save();
        if($result){
            // return "Employee is added";
            // Redirected to list-employee url
            return redirect('list-employee');
        }
        else{
            return "Employee is not added";
        }
    }
    function list(){
        $employeedata = employee::all();
        // return $employeedata;
        return view('list-employee',[ 'data'=>$employeedata]);
    }
}
```

# View (List-employee)



The screenshot shows a dark-themed interface of the Visual Studio Code code editor. The title bar reads "Laravel-WorkSpace". The left sidebar has icons for Explorer, Search, Problems, Files, and Outline. The Explorer view shows a project structure under "LARAVEL-WORKSPACE": "laravel-demo" (with "resources" and "views" expanded), "components", "layouts", "about.blade.php", "add-employee.blade.php", "databaseuser.blade.php", "DBqueryView.blade.php", "Demo.blade.php", "DisplayFile.blade.php", "EmodelQueryBuilder.blade.php", "formvalidation.blade.php", "groupprefix.blade.php", "home.blade.php", "home2.blade.php", and "list-employee.blade.php" (which is currently selected and highlighted in green). The main editor area displays the contents of "list-employee.blade.php".

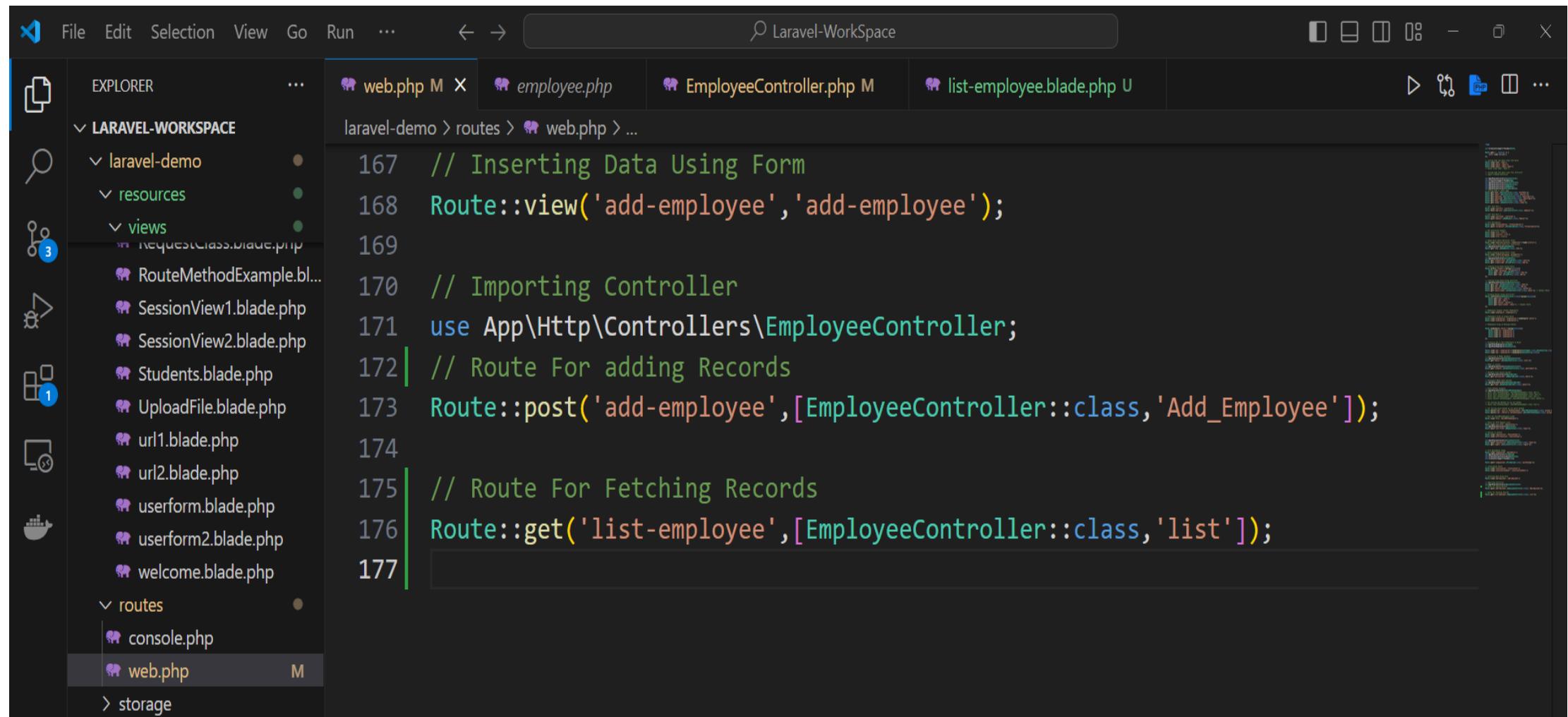
```
<div>
    <h1>Employee List</h1>
    <!-- {{print_r($data)}} -->
    <table border="1">
        <tr>
            <th>ID</th>
            <th>Name</th>
            <th>Email</th>
            <th>Phone</th>
            <th>Created_at</th>
            <th>Updated_at</th>
        </tr>
        @foreach($data as $list_data)
        <tr>
            <td>$list_data->id</td>
            <td>$list_data->name</td>
            <td>$list_data->email</td>
            <td>$list_data->phone</td>
            <td>$list_data->created_at</td>
            <td>$list_data->updated_at</td>
        </tr>
        @endforeach
    </table>
    <!-- Be present above all else. - Naval Ravikant -->
</div>
```

# Model (Employee)

The screenshot shows a code editor interface with the title bar "Laravel-WorkSpace". The left sidebar is the "EXPLORER" view, showing the project structure under "LARAVEL-WORKSPACE". The "Models" folder contains three files: employee.php (selected), Student.php, and user.php. The main editor area displays the contents of employee.php:

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Model;
6
7 class employee extends Model
8 {
9     //mention your table name for modification
10    // protected $table = "your table name";
11 }
12
```

# Route



The screenshot shows a code editor interface for a Laravel workspace named "Laravel-WorkSpace". The left sidebar displays the project structure under "Laravel-WORKSPACE". The "routes" folder contains two files: "console.php" and "web.php". The "web.php" file is currently open and shows the following PHP code:

```
167 // Inserting Data Using Form
168 Route::view('add-employee','add-employee');
169
170 // Importing Controller
171 use App\Http\Controllers\EmployeeController;
172 // Route For adding Records
173 Route::post('add-employee',[EmployeeController::class,'Add_Employee']);
174
175 // Route For Fetching Records
176 Route::get('list-employee',[EmployeeController::class,'list']);
177
```

The code defines routes for inserting data using a form, importing the EmployeeController, and defining routes for adding and fetching employee records.

# Output (<http://localhost:8000/add-employee>)

A screenshot of a web browser window displaying a form titled "Add New Employee". The form contains fields for Name, Email, and Phone, each with a placeholder value. A blue "Add Employee" button is at the bottom.

The browser's address bar shows the URL [localhost:8000/add-employee](http://localhost:8000/add-employee). The page title is "Add New Employee".

Field	Value
Name:	Yitayew Solomon
Email:	yitayewsolomon3@gmail.com
Phone:	0961014202

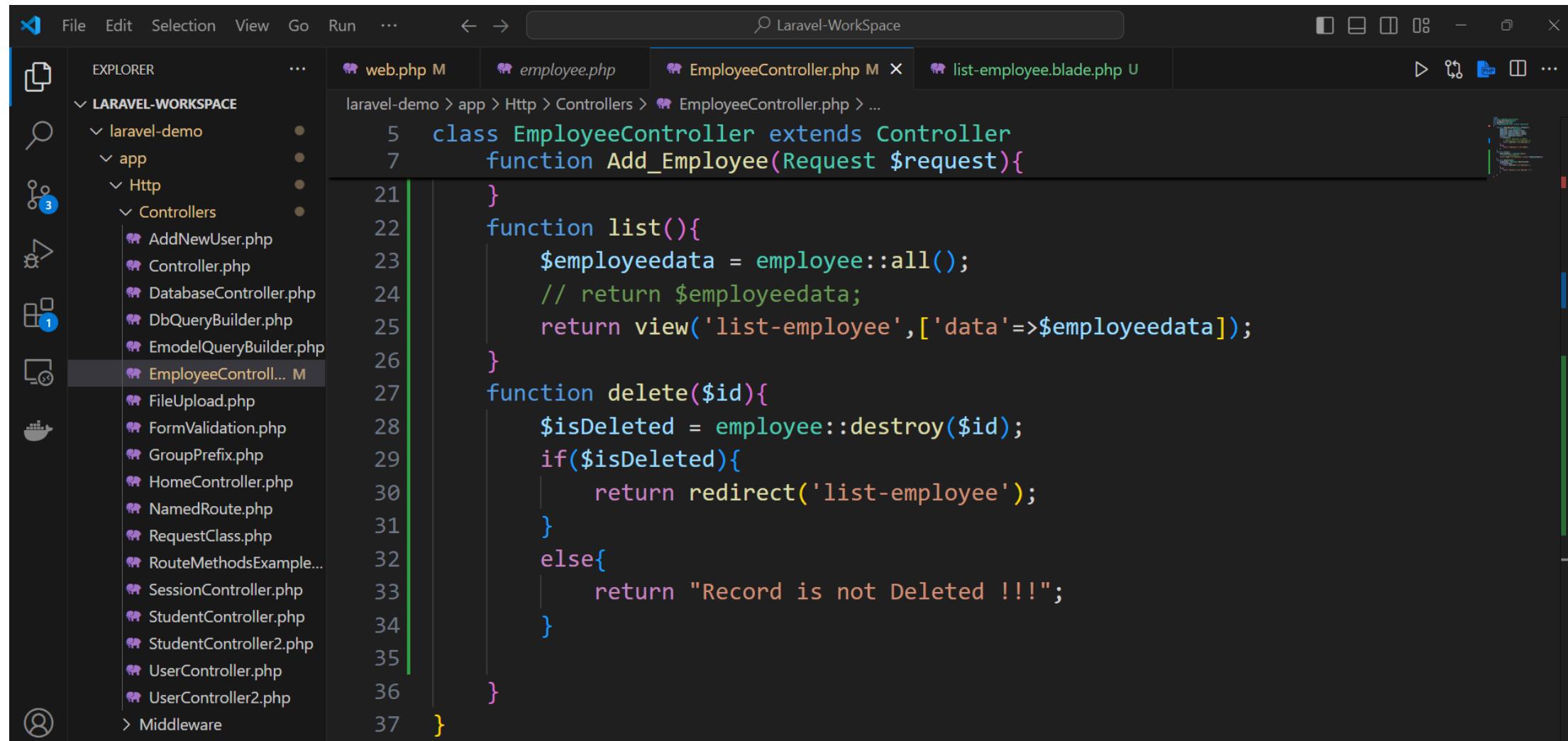
**Add Employee**

# Cont. ... (<http://localhost:8000/list-employee>)

Employee List

ID	Name	Email	Phone	Created_at	Updated_at
1	Yitayew Solomon	yitayewsolomon3@gmail.com	0937559718	2024-11-12 00:00:00	2024-11-12 00:00:00
2	Haileab Solomon	haileabsolomon@gmail.com	0937559718	2024-11-12 00:00:00	2024-11-12 00:00:00
3	Yared Solomon	yaredsolomon@gmail.com	0937559718	2024-11-12 00:00:00	2024-11-12 00:00:00
6	Natanim Yitayew	natan@gmail.com	0937559720	2024-11-12 00:00:00	2024-11-12 00:00:00
7	Dureti Guye	dure@gmail.com	0910112244	2024-11-13 00:00:00	2024-11-13 00:00:00
8	Yitayew Solomon	yitayewsolomon3@gmail.com	0961014202	2024-11-13 00:00:00	2024-11-13 00:00:00

# Delete Data From MySql Table



The screenshot shows a dark-themed interface of Visual Studio Code (VS Code) with the title bar "Laravel-WorkSpace". The left sidebar is the Explorer, showing a file tree for a "laravel-demo" workspace under "LARAVEL-WORKSPACE". The tree includes "app", "Http", and "Controllers" folders containing various PHP files like "EmployeeController.php", "AddNewUser.php", etc. The main editor area displays the "EmployeeController.php" file. The code is written in PHP and defines a class "EmployeeController" extending "Controller". It contains three methods: "Add\_Employee", "list", and "delete". The "list" method retrieves all employee data from the database and returns it to a view. The "delete" method attempts to delete an employee by ID; if successful, it redirects to the list page; if not, it returns a message indicating the record was not deleted.

```
class EmployeeController extends Controller
{
    function Add_Employee(Request $request){
    }

    function list(){
        $employeedata = employee::all();
        // return $employeedata;
        return view('list-employee',[ 'data'=>$employeedata]);
    }

    function delete($id){
        $isDeleted = employee::destroy($id);
        if($isDeleted){
            return redirect('list-employee');
        }
        else{
            return "Record is not Deleted !!!";
        }
    }
}
```

# View (List-employee)

The screenshot shows a dark-themed interface of the Visual Studio Code (VS Code) code editor. At the top, the title bar displays "Laravel-WorkSpace". The left sidebar features a "File Explorer" with a tree view of the "laravel-demo" project structure under "LARAVEL-WORKSPACE". The "views" folder contains several Blade files, including "add-employee.blade.php", "databaseuser.blade.php", "DBqueryView.blade.php", "Demo.blade.php", "DisplayFile.blade.php", "EmodelQueryBuilder.blade.php", "formvalidation.blade.php", "groupprefix.blade.php", "home.blade.php", "home2.blade.php", and "list-employee.blade.php", which is currently selected and highlighted in green.

The main editor area shows the content of "list-employee.blade.php". The code is a Blade template for displaying a list of employees. It starts with an `<div>` tag containing an `<h1>Employee List</h1>` heading. It includes a comment block `<!-- {{print_r($data)}} -->`. A `<table border="1">` is used to structure the data. The table has a header row (`<tr>`) with columns for ID, Name, Email, Phone, Created\_at, Updated\_at, and Operations. The body of the table (`@foreach($data as $list_data)`) contains rows for each employee, with columns for id, name, email, phone, created\_at, updated\_at, and a "Delete" link generated by a Blade href helper. The template ends with a closing `</table>` tag and a final comment block `<!-- Be present above all else. - Naval Ravikant -->`.

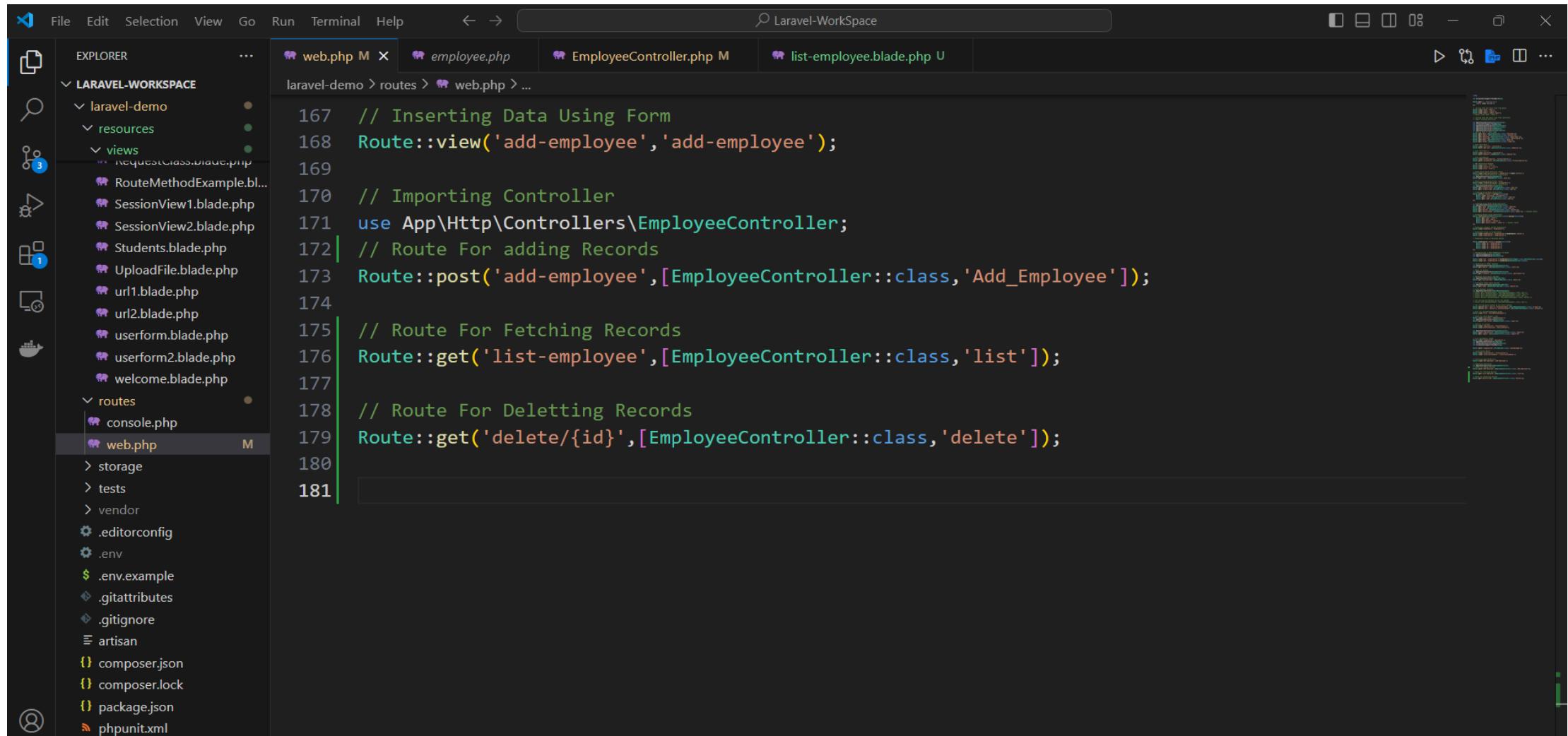
```
1 <div>
2   <h1>Employee List</h1>
3   <!-- {{print_r($data)}} -->
4   <table border="1">
5     <tr>
6       <th>ID</th>
7       <th>Name</th>
8       <th>Email</th>
9       <th>Phone</th>
10      <th>Created_at</th>
11      <th>Updated_at</th>
12      <th>Operations</th>
13    </tr>
14    @foreach($data as $list_data)
15      <tr>
16        <td>$list_data->id</td>
17        <td>$list_data->name</td>
18        <td>$list_data->email</td>
19        <td>$list_data->phone</td>
20        <td>$list_data->created_at</td>
21        <td>$list_data->updated_at</td>
22        <td><a href="{{'delete/'.$list_data->id}}>Delete</a></td>
23      </tr>
24    @endforeach
25  </table>
26  <!-- Be present above all else. - Naval Ravikant -->
```

# Model (Employee)

The screenshot shows a code editor interface with the title bar "Laravel-WorkSpace". The left sidebar is the "EXPLORER" view, showing the project structure under "LARAVEL-WORKSPACE". The "Models" folder contains three files: employee.php, Student.php, and user.php. The "employee.php" file is currently selected and shown in the main editor area.

```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Model;
6
7 class employee extends Model
8 {
9     //mention your table name for modification
10    // protected $table = "your table name";
11 }
12
```

# Route

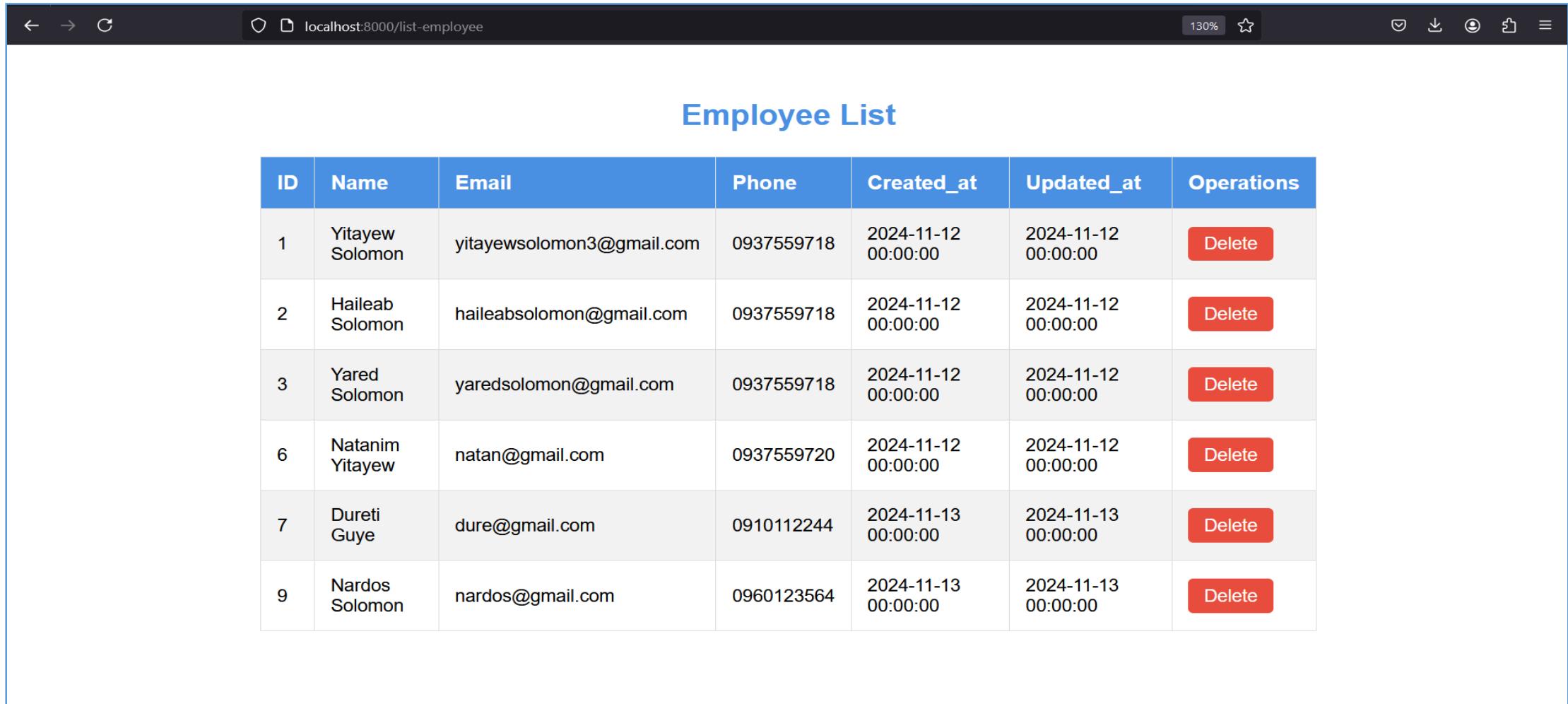


The screenshot shows a code editor interface with a dark theme, displaying PHP code for Laravel routes. The file is named `web.php` and is located in the `routes` directory of a Laravel workspace. The code defines several routes:

```
167 // Inserting Data Using Form
168 Route::view('add-employee', 'add-employee');
169
170 // Importing Controller
171 use App\Http\Controllers\EmployeeController;
172 // Route For adding Records
173 Route::post('add-employee',[EmployeeController::class,'Add_Employee']);
174
175 // Route For Fetching Records
176 Route::get('list-employee',[EmployeeController::class,'list']);
177
178 // Route For Deleting Records
179 Route::get('delete/{id}',[EmployeeController::class,'delete']);
180
181
```

The code editor also shows other files in the workspace like `employee.php`, `EmployeeController.php`, and `list-employee.blade.php`. The sidebar on the left shows the project structure with a tree view of files and folders.

# Output (<http://localhost:8000/list-employee>)



A screenshot of a web browser displaying a table titled "Employee List". The table has columns for ID, Name, Email, Phone, Created\_at, Updated\_at, and Operations. Each row contains data for an employee, including their ID, name, email, phone number, and creation and update timestamps. A red "Delete" button is present in the Operations column for each row.

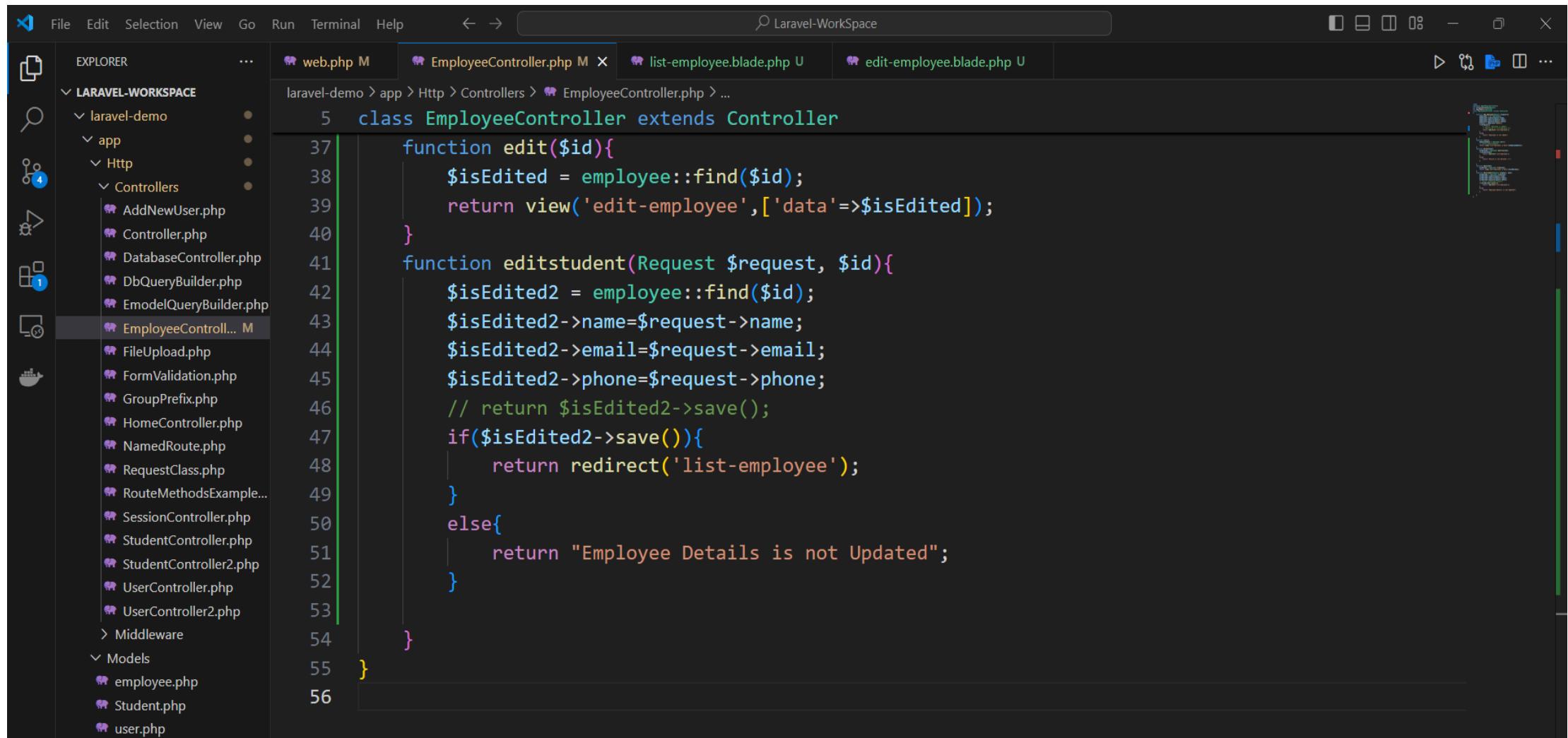
ID	Name	Email	Phone	Created_at	Updated_at	Operations
1	Yitayew Solomon	yitayewsolomon3@gmail.com	0937559718	2024-11-12 00:00:00	2024-11-12 00:00:00	<button>Delete</button>
2	Haileab Solomon	haileabsolomon@gmail.com	0937559718	2024-11-12 00:00:00	2024-11-12 00:00:00	<button>Delete</button>
3	Yared Solomon	yaredsolomon@gmail.com	0937559718	2024-11-12 00:00:00	2024-11-12 00:00:00	<button>Delete</button>
6	Natanim Yitayew	natan@gmail.com	0937559720	2024-11-12 00:00:00	2024-11-12 00:00:00	<button>Delete</button>
7	Dureti Guye	dure@gmail.com	0910112244	2024-11-13 00:00:00	2024-11-13 00:00:00	<button>Delete</button>
9	Nardos Solomon	nardos@gmail.com	0960123564	2024-11-13 00:00:00	2024-11-13 00:00:00	<button>Delete</button>

# Output (After Deleting)

Employee List

ID	Name	Email	Phone	Created_at	Updated_at	Operations
1	Yitayew Solomon	yitayewsolomon3@gmail.com	0937559718	2024-11-12 00:00:00	2024-11-12 00:00:00	<button>Delete</button>
2	Haileab Solomon	haileabsolomon@gmail.com	0937559718	2024-11-12 00:00:00	2024-11-12 00:00:00	<button>Delete</button>
3	Yared Solomon	yaredsolomon@gmail.com	0937559718	2024-11-12 00:00:00	2024-11-12 00:00:00	<button>Delete</button>
6	Natanim Yitayew	natan@gmail.com	0937559720	2024-11-12 00:00:00	2024-11-12 00:00:00	<button>Delete</button>
7	Dureti Guye	dure@gmail.com	0910112244	2024-11-13 00:00:00	2024-11-13 00:00:00	<button>Delete</button>

# Populate Data in HTML form from MySql Table

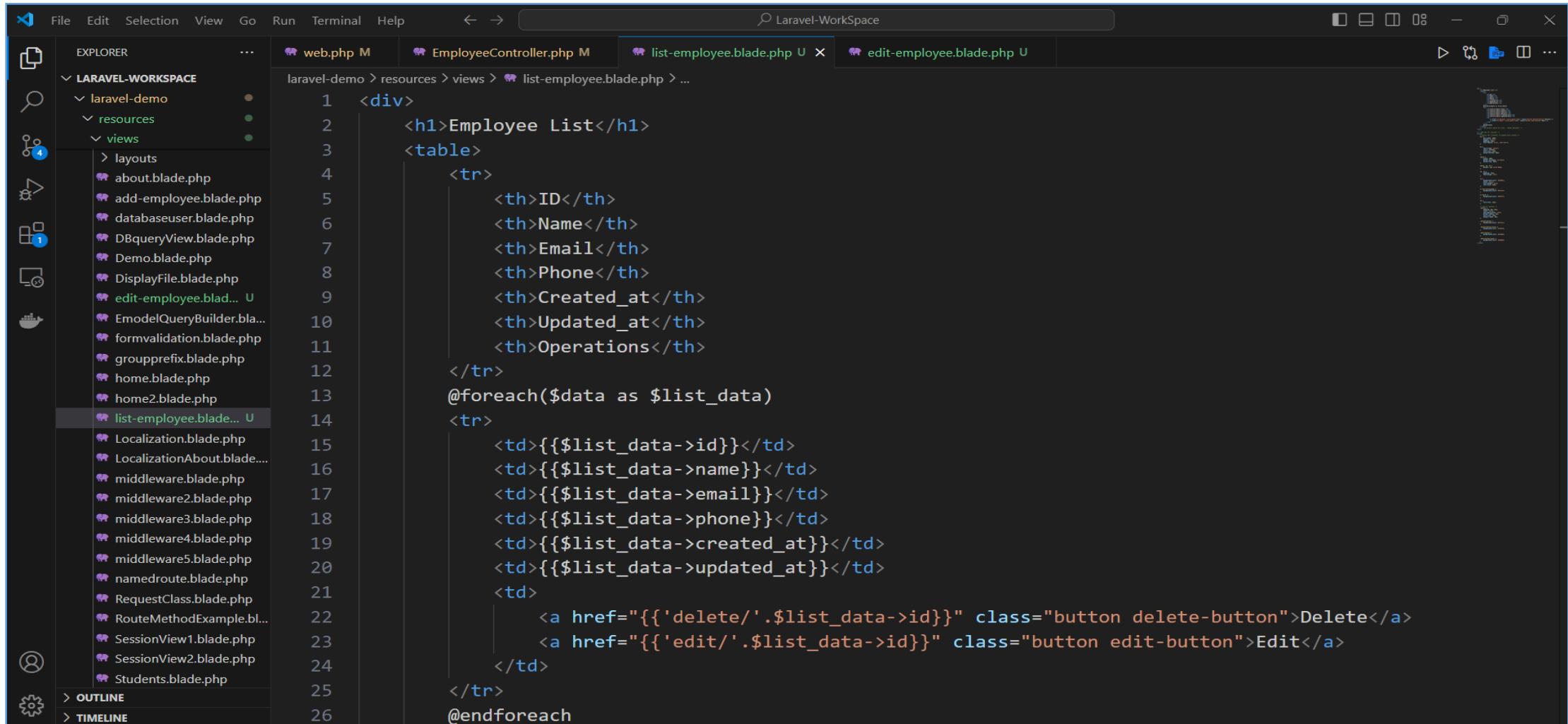


The screenshot shows a dark-themed interface of Visual Studio Code (VS Code) with an orange header bar. The title bar says "Laravel-WorkSpace". The left sidebar has icons for Explorer, Search, Problems, and others. The "EXPLORER" section shows a tree view of a Laravel workspace named "laravel-demo". Under "app/Http/Controllers", several files are listed, including "EmployeeController.php" which is currently selected and shown in the main editor area. The "EmployeeController.php" file contains PHP code for a controller:

```
class EmployeeController extends Controller
{
    function edit($id){
        $isEdited = employee::find($id);
        return view('edit-employee',[ 'data'=>$isEdited]);
    }

    function editstudent(Request $request, $id){
        $isEdited2 = employee::find($id);
        $isEdited2->name=$request->name;
        $isEdited2->email=$request->email;
        $isEdited2->phone=$request->phone;
        // return $isEdited2->save();
        if($isEdited2->save()){
            return redirect('list-employee');
        }
        else{
            return "Employee Details is not Updated";
        }
    }
}
```

# View (List-Employee)



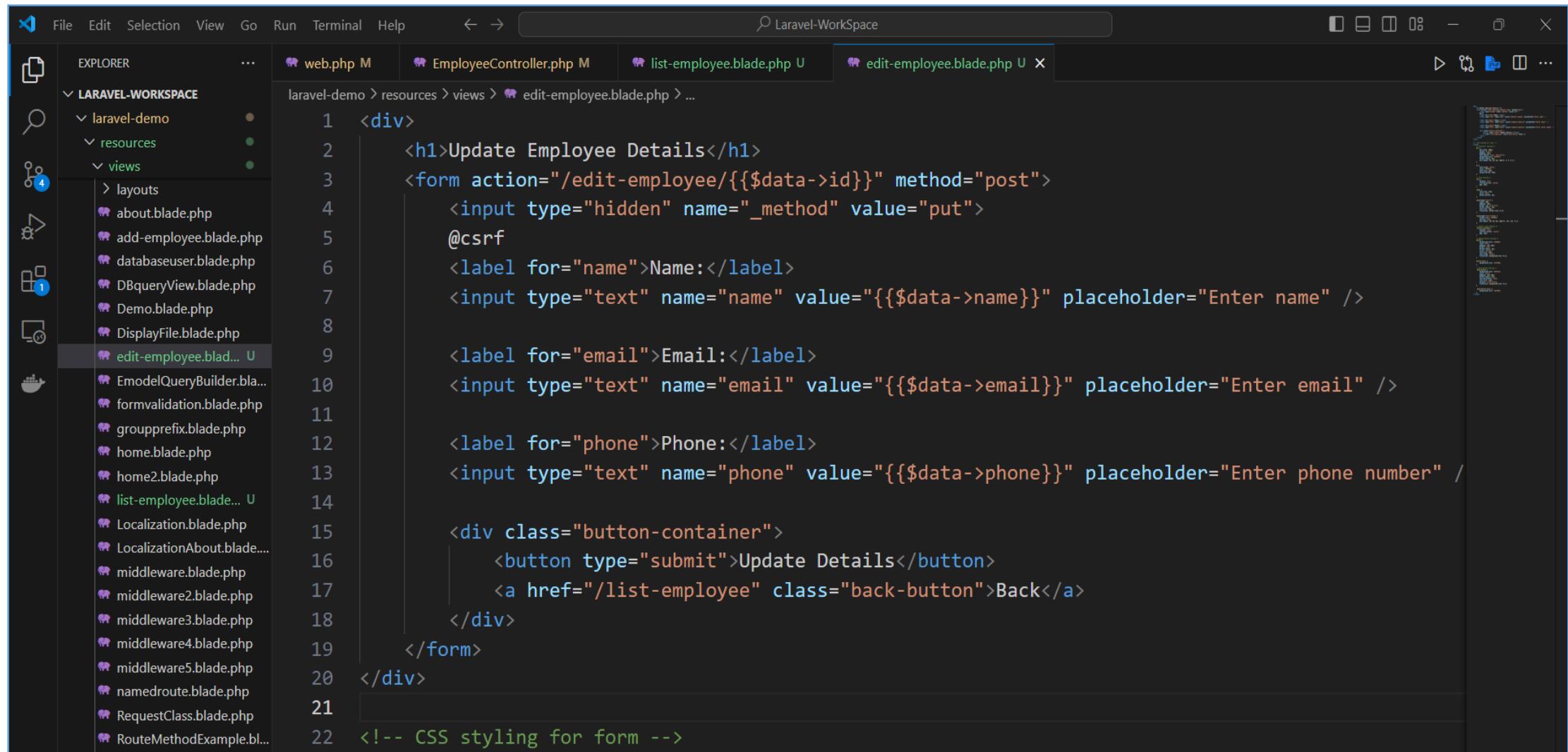
The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists the project structure under 'LARAVEL-WORKSPACE'. The 'views' folder contains several blade files, including 'list-employee.blade.php' which is currently selected and shown in the main editor area.

The main editor area displays the following code:

```
<div>
    <h1>Employee List</h1>
    <table>
        <thead>
            <tr>
                <th>ID</th>
                <th>Name</th>
                <th>Email</th>
                <th>Phone</th>
                <th>Created_at</th>
                <th>Updated_at</th>
                <th>Operations</th>
            </tr>
        </thead>
        <tbody>
            @foreach($data as $list_data)
            <tr>
                <td>{{$list_data->id}}</td>
                <td>{{$list_data->name}}</td>
                <td>{{$list_data->email}}</td>
                <td>{{$list_data->phone}}</td>
                <td>{{$list_data->created_at}}</td>
                <td>{{$list_data->updated_at}}</td>
                <td>
                    <a href="{{'delete/'.$list_data->id}}" class="button delete-button">Delete</a>
                    <a href="{{'edit/'.$list_data->id}}" class="button edit-button">Edit</a>
                </td>
            </tr>
            @endforeach
        </tbody>
    </table>

```

# View (edit-employee)

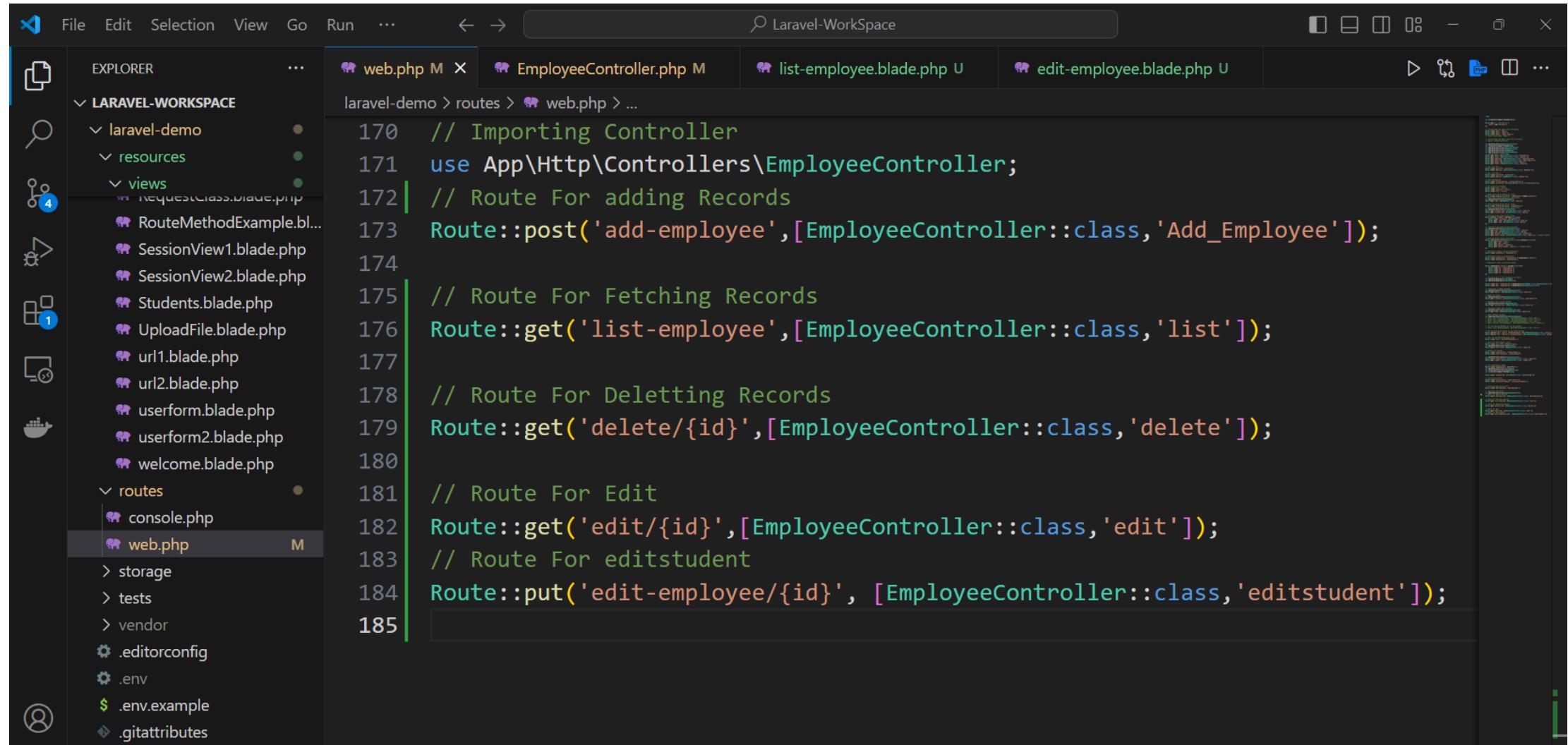


The screenshot shows a code editor interface with a dark theme. The title bar says "Laravel-WorkSpace". The left sidebar has icons for File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar. The Explorer sidebar shows a project structure under "LARAVEL-WORKSPACE": "laravel-demo" > "resources" > "views" > "edit-employee.blade.php". The main editor area displays the following Blade template code:

```
1 <div>
2   <h1>Update Employee Details</h1>
3   <form action="/edit-employee/{{$data->id}}" method="post">
4     <input type="hidden" name="_method" value="put">
5     @csrf
6     <label for="name">Name:</label>
7     <input type="text" name="name" value="{{$data->name}}" placeholder="Enter name" />
8
9     <label for="email">Email:</label>
10    <input type="text" name="email" value="{{$data->email}}" placeholder="Enter email" />
11
12    <label for="phone">Phone:</label>
13    <input type="text" name="phone" value="{{$data->phone}}" placeholder="Enter phone number" />
14
15    <div class="button-container">
16      <button type="submit">Update Details</button>
17      <a href="/list-employee" class="back-button">Back</a>
18    </div>
19  </form>
20 </div>
21
22 <!-- CSS styling for form -->
```

The code includes a form for updating employee details, with fields for Name, Email, and Phone, and buttons for Update Details and Back.

# Route



The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists the project structure under 'LARAVEL-WORKSPACE'. The 'routes' folder contains 'console.php' and 'web.php', with 'web.php' currently selected. The main editor area displays the 'web.php' file, which contains the following PHP code:

```
170 // Importing Controller
171 use App\Http\Controllers\EmployeeController;
172 // Route For adding Records
173 Route::post('add-employee',[EmployeeController::class,'Add_Employee']);
174
175 // Route For Fetching Records
176 Route::get('list-employee',[EmployeeController::class,'list']);
177
178 // Route For Deleting Records
179 Route::get('delete/{id}',[EmployeeController::class,'delete']);
180
181 // Route For Edit
182 Route::get('edit/{id}',[EmployeeController::class,'edit']);
183 // Route For editstudent
184 Route::put('edit-employee/{id}', [EmployeeController::class,'editstudent']);
185
```

The code editor has tabs for 'EmployeeController.php', 'list-employee.blade.php', and 'edit-employee.blade.php'. The status bar at the bottom right shows a file count of 4.

# Output (<http://localhost:8000/list-employee>)

Employee List

ID	Name	Email	Phone	Created_at	Updated_at	Operations
1	Yitayew Solomon	yitayewsolomon3@gmail.com	0937559718	2024-11-12 00:00:00	2024-11-12 00:00:00	<button>Delete</button> <button>Edit</button>
2	Haileab Solomon	haileabsolomon@gmail.com	0937559718	2024-11-12 00:00:00	2024-11-12 00:00:00	<button>Delete</button> <button>Edit</button>
3	Yared Solomon	yaredsolomon@gmail.com	0937559718	2024-11-12 00:00:00	2024-11-12 00:00:00	<button>Delete</button> <button>Edit</button>
6	Natanim Yitayew	natan@gmail.com	0937559720	2024-11-12 00:00:00	2024-11-12 00:00:00	<button>Delete</button> <button>Edit</button>
7	Dureti Guye	dure@gmail.com	0912787576	2024-11-13 00:00:00	2024-11-13 00:00:00	<button>Delete</button> <button>Edit</button>

# Output(<http://localhost:8000/edit/7>)

A screenshot of a web browser window displaying an 'Update Employee Details' form. The browser's address bar shows the URL [localhost:8000/edit/7](http://localhost:8000/edit/7). The page title is 'Update Employee Details'. The form contains three input fields: 'Name' with value 'Dureti Guye', 'Email' with value 'dure@gmail.com', and 'Phone' with value '0910010203'. At the bottom are two buttons: a blue 'Update Details' button and an orange 'Back' button.

← → C    130% ☆

localhost:8000/edit/7

Update Employee Details

Name:

Dureti Guye

Email:

dure@gmail.com

Phone:

0910010203

Update Details    Back

# Output (<http://localhost:8000/list-employee>)

Employee List

ID	Name	Email	Phone	Created_at	Updated_at	Operations
1	Yitayew Solomon	yitayewsolomon3@gmail.com	0937559718	2024-11-12 00:00:00	2024-11-12 00:00:00	<button>Delete</button> <button>Edit</button>
2	Haileab Solomon	haileabsolomon@gmail.com	0937559718	2024-11-12 00:00:00	2024-11-12 00:00:00	<button>Delete</button> <button>Edit</button>
3	Yared Solomon	yaredsolomon@gmail.com	0937559718	2024-11-12 00:00:00	2024-11-12 00:00:00	<button>Delete</button> <button>Edit</button>
6	Natanim Yitayew	natan@gmail.com	0937559720	2024-11-12 00:00:00	2024-11-12 00:00:00	<button>Delete</button> <button>Edit</button>
7	Dureti Guye	dure@gmail.com	0910010203	2024-11-13 00:00:00	2024-11-13 00:00:00	<button>Delete</button> <button>Edit</button>

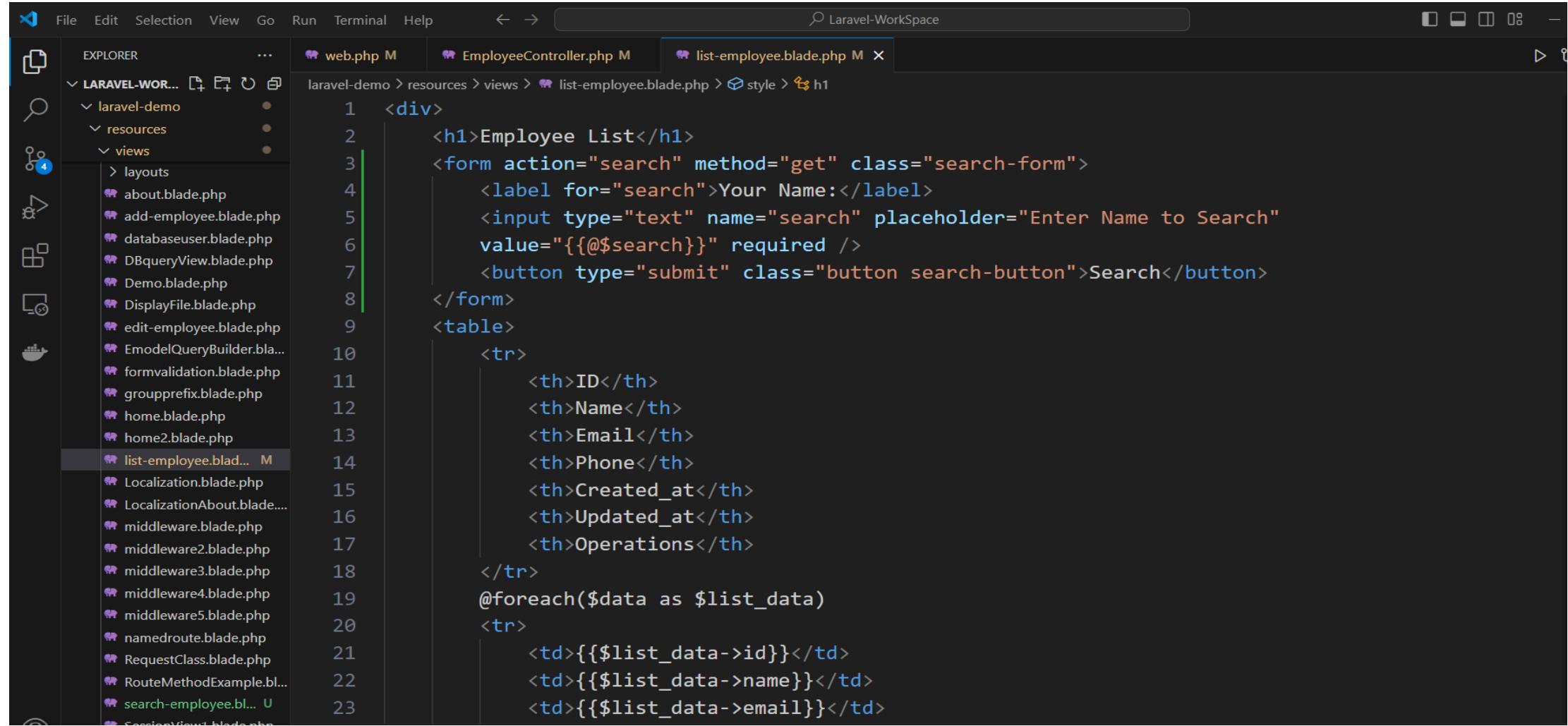
# Search Data From MySql Database Table

The screenshot shows a code editor interface with a dark theme. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. A search bar at the top right contains the text "Laravel-WorkSpace". The left sidebar, titled "EXPLORER", shows a file tree under "LARAVEL-WORKSPACE" named "laravel-demo". Inside "laravel-demo", there are "app", "Http", and "Controllers" folders. The "Controllers" folder contains several files: AddNewUser.php, Controller.php, DatabaseController.php, DbQueryBuilder.php, EmodelQueryBuilder.php, EmployeeController.php (which is selected and highlighted in grey), FileUpload.php, FormValidation.php, GroupPrefix.php, HomeController.php, NamedRoute.php, RequestClass.php, RouteMethodsExample.php, SessionController.php, StudentController.php, StudentController2.php, UserController.php, and UserController2.php. Below these are "Middleware" and "Models" sections. The main editor area displays PHP code for the "EmployeeController.php" file. The code defines two functions: "editstudent" and "search". The "editstudent" function takes a Request \$request and an id, updates an employee record, and returns a redirect to the employee list. The "search" function takes a Request \$request, performs a database search based on the name field, and returns a view with the search results.

```
class EmployeeController extends Controller
{
    public function editstudent(Request $request, $id){
        $isEdited2->email=$request->email;
        $isEdited2->phone=$request->phone;
        // return $isEdited2->save();
        if($isEdited2->save()){
            return redirect('list-employee');
        }
        else{
            return "Employee Details is not Updated";
        }
    }

    public function search(Request $request){
        // return $request->search;
        $EmployeeDataSearch = employee::where("name","like","%$request->search%")->get();
        return view('list-employee',[ 'data'=>$EmployeeDataSearch,'search'=>$request->search]);
    }
}
```

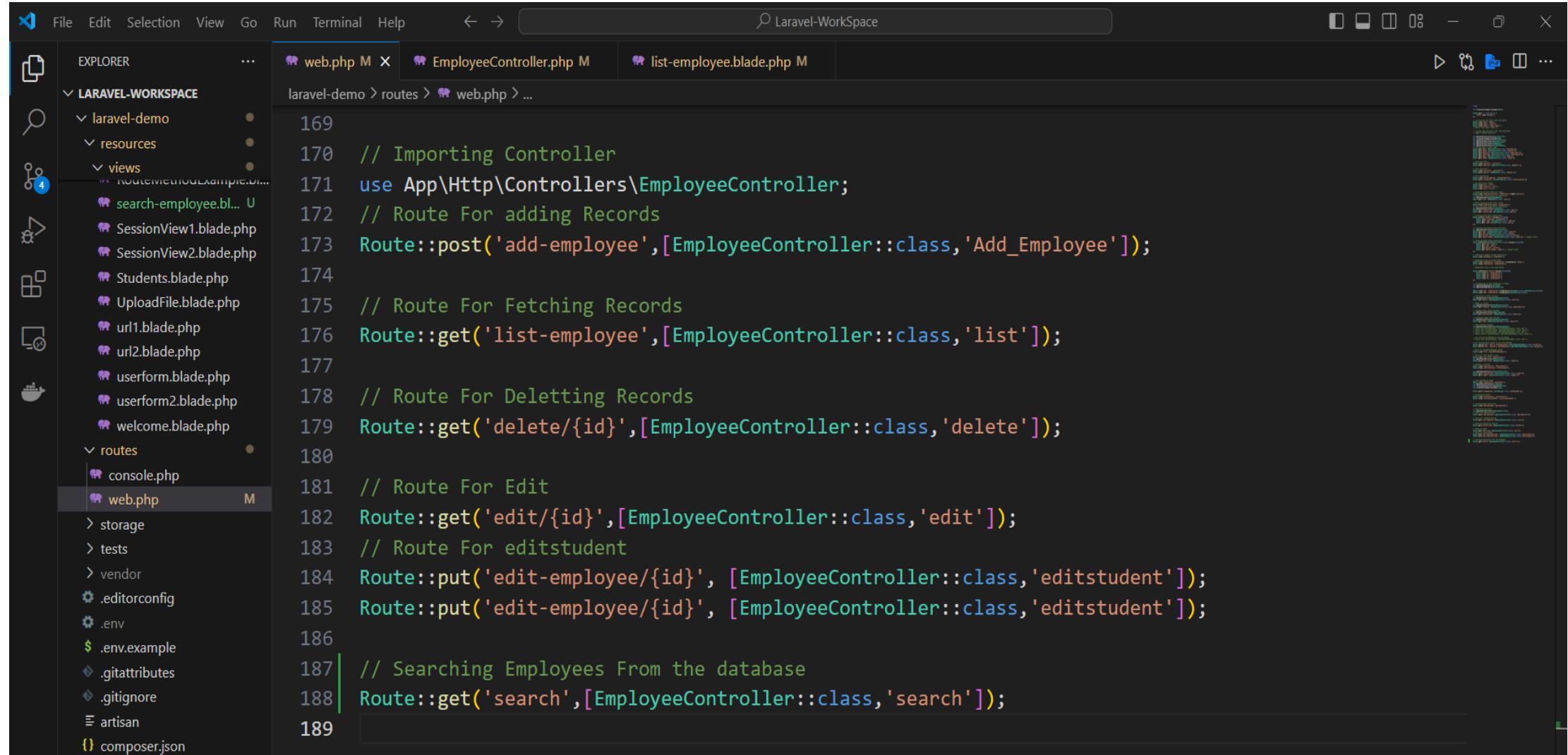
# View (list-employee)



The screenshot shows a code editor interface with a dark theme. The left sidebar contains icons for File, Edit, Selection, View, Go, Run, Terminal, and Help. The Explorer panel on the left shows a project structure under 'LARAVEL-WOR...' with a folder 'laravel-demo' containing 'resources' and 'views'. Inside 'views', there are several blade files like 'about.blade.php', 'add-employee.blade.php', etc., and the file 'list-employee.blade.php' which is currently selected. The top navigation bar includes a search bar for 'Laravel-WorkSpace' and various window control icons. The main editor area displays the following PHP Blade code:

```
1 <div>
2   <h1>Employee List</h1>
3   <form action="search" method="get" class="search-form">
4     <label for="search">Your Name:</label>
5     <input type="text" name="search" placeholder="Enter Name to Search"
6           value="{{@$search}}" required />
7     <button type="submit" class="button search-button">Search</button>
8   </form>
9   <table>
10    <tr>
11      <th>ID</th>
12      <th>Name</th>
13      <th>Email</th>
14      <th>Phone</th>
15      <th>Created_at</th>
16      <th>Updated_at</th>
17      <th>Operations</th>
18    </tr>
19    @foreach($data as $list_data)
20    <tr>
21      <td>{{$list_data->id}}</td>
22      <td>{{$list_data->name}}</td>
23      <td>{{$list_data->email}}</td>
```

# Route



The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with the title bar "Laravel-WorkSpace". The left sidebar displays the "EXPLORER" view, which lists the project structure under "LARAVEL-WORKSPACE". The "routes" folder contains "console.php" and "web.php", where "web.php" is currently selected. The main editor area shows the "web.php" file content:

```
169
170 // Importing Controller
171 use App\Http\Controllers\EmployeeController;
172 // Route For adding Records
173 Route::post('add-employee',[EmployeeController::class,'Add_Employee']);
174
175 // Route For Fetching Records
176 Route::get('list-employee',[EmployeeController::class,'list']);
177
178 // Route For Deleting Records
179 Route::get('delete/{id}',[EmployeeController::class,'delete']);
180
181 // Route For Edit
182 Route::get('edit/{id}',[EmployeeController::class,'edit']);
183 // Route For editstudent
184 Route::put('edit-employee/{id}', [EmployeeController::class,'editstudent']);
185 Route::put('edit-employee/{id}', [EmployeeController::class,'editstudent']);
186
187 // Searching Employees From the database
188 Route::get('search',[EmployeeController::class,'search']);
189
```

# Output (<http://127.0.0.1:8000/search?search=>)

The screenshot shows a web browser window with multiple tabs open. The active tab is titled "127.0.0.1:8000/search?search=". The page content is titled "Employee List". It features a search bar with the placeholder "Enter Name to Search" and a green "Search" button. Below the search bar is a table with columns: ID, Name, Email, Phone, Created\_at, Updated\_at, and Operations. The table contains five rows of employee data.

ID	Name	Email	Phone	Created_at	Updated_at	Operations
1	Yitayew Solomon	yitayewsolomon3@gmail.com	0937559718	2024-11-12 00:00:00	2024-11-12 00:00:00	<a href="#">Delete</a> <a href="#">Edit</a>
2	Haileab Solomon	haileabsolomon@gmail.com	0937559718	2024-11-12 00:00:00	2024-11-12 00:00:00	<a href="#">Delete</a> <a href="#">Edit</a>
3	Yared Solomon	yaredsolomon@gmail.com	0937559718	2024-11-12 00:00:00	2024-11-12 00:00:00	<a href="#">Delete</a> <a href="#">Edit</a>
6	Natanim Yitayew	natan@gmail.com	0937559720	2024-11-12 00:00:00	2024-11-12 00:00:00	<a href="#">Delete</a> <a href="#">Edit</a>
7	Dureti Guye	dure@gmail.com	0910010203	2024-11-13 00:00:00	2024-11-13 00:00:00	<a href="#">Delete</a> <a href="#">Edit</a>

# Output (<http://127.0.0.1:8000/search?search=Yitayew>)

The screenshot shows a web browser window with the URL <http://127.0.0.1:8000/search?search=Yitayew> in the address bar. The page title is "Employee List". A search bar contains the text "Yitayew". Below the search bar is a table with columns: ID, Name, Email, Phone, Created\_at, Updated\_at, and Operations. Two rows of data are visible.

ID	Name	Email	Phone	Created_at	Updated_at	Operations
1	Yitayew Solomon	yitayewsolomon3@gmail.com	0937559718	2024-11-12 00:00:00	2024-11-12 00:00:00	<a href="#">Delete</a> <a href="#">Edit</a>
6	Natanim Yitayew	natan@gmail.com	0937559720	2024-11-12 00:00:00	2024-11-12 00:00:00	<a href="#">Delete</a> <a href="#">Edit</a>

# Pagination in Laravel

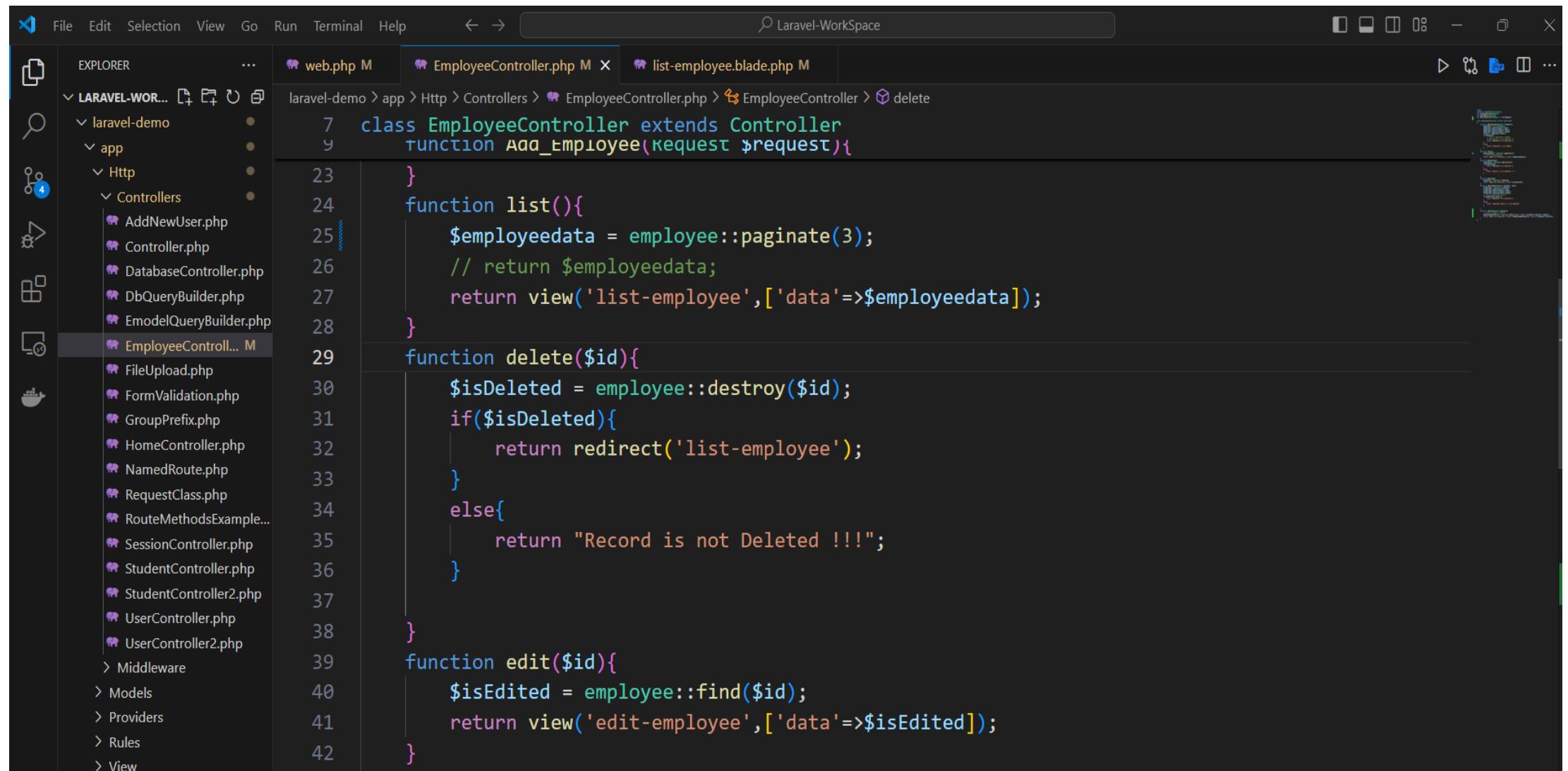
- Pagination in Laravel makes it easy to divide large datasets into manageable chunks, enhancing the performance and usability of your application. Laravel provides built-in methods for paginating database queries, making the process simple and efficient.

```
php
use App\Models\Employee;

public function listEmployees() {
    // Fetch employees with 10 records per page
    $employees = Employee::paginate(10);

    // Return the paginated data to the view
    return view('list-employees', compact('employees'));
}
```

# Example



The screenshot shows a dark-themed interface of the Visual Studio Code (VS Code) code editor. The title bar reads "Laravel-WorkSpace". The left sidebar contains icons for File, Edit, Selection, View, Go, Run, Terminal, and Help, along with a search bar and a tab labeled "Laravel-WorkSpace". The main area has tabs for "web.php M", "EmployeeController.php M", and "list-employee.blade.php M". The "EmployeeController.php M" tab is active, displaying PHP code for a controller named "EmployeeController". The code includes methods for adding a new employee, listing employees, deleting an employee, and editing an employee. The code editor shows line numbers from 7 to 42. The right side of the interface features a vertical panel with a preview of the "list-employee.blade.php" blade file.

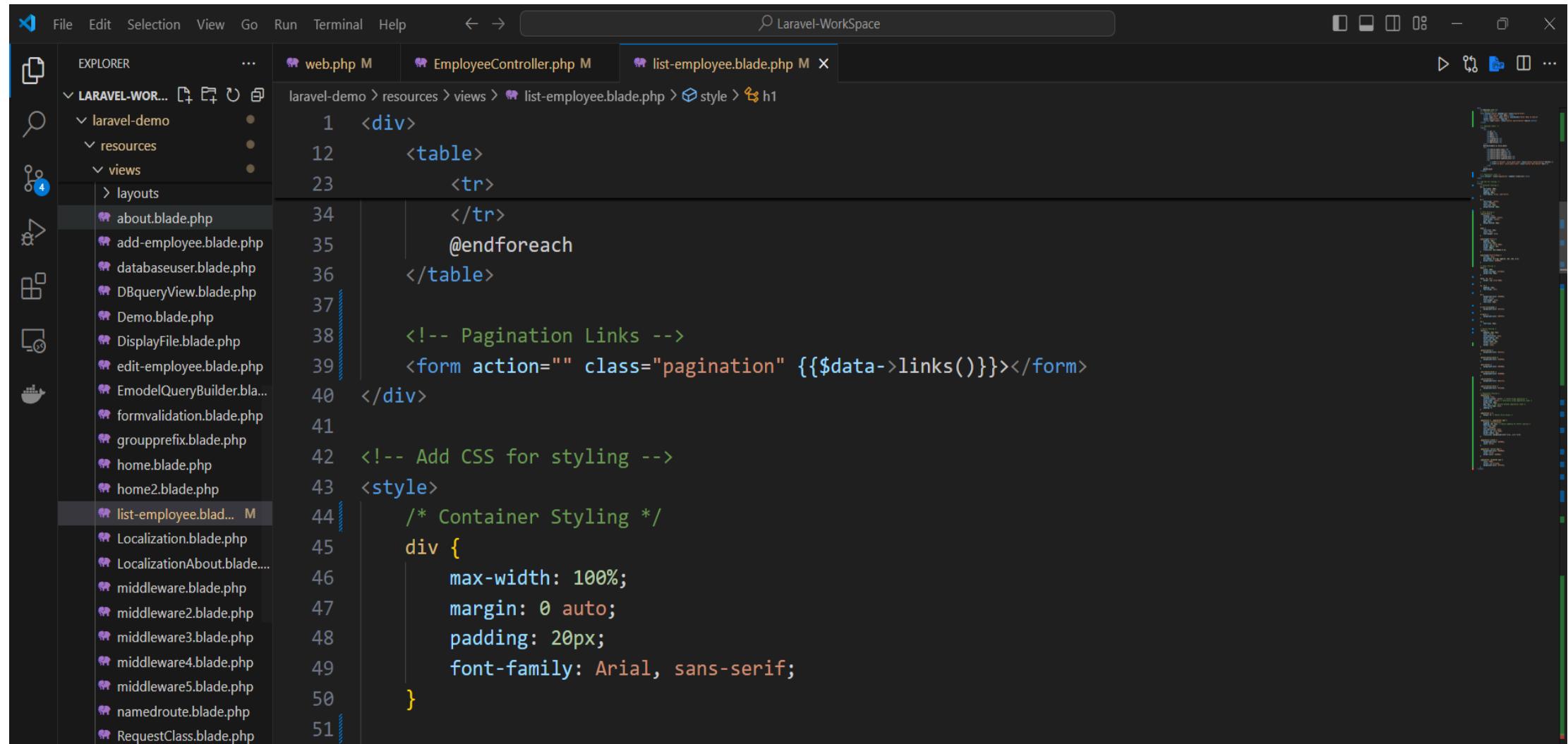
```
class EmployeeController extends Controller
{
    function Add_Employee(request $request){
    }

    function list(){
        $employeedata = employee::paginate(3);
        // return $employeedata;
        return view('list-employee',[ 'data'=>$employeedata]);
    }

    function delete($id){
        $isDeleted = employee::destroy($id);
        if($isDeleted){
            return redirect('list-employee');
        }
        else{
            return "Record is not Deleted !!!";
        }
    }

    function edit($id){
        $isEdited = employee::find($id);
        return view('edit-employee',[ 'data'=>$isEdited]);
    }
}
```

# Cont. ...



The screenshot shows the Visual Studio Code interface with a dark theme. The left sidebar contains icons for Explorer, Search, Problems (with 4 items), and others. The Explorer panel shows a file tree for a 'Laravel-Wor...' workspace, with 'laravel-demo' expanded to show 'resources' and 'views'. Inside 'views', 'list-employee.blade.php' is selected. The main editor area displays the following Blade template code:

```
1 <div>
2     <table>
3         <tr>
4             </tr>
5         @foreach
6             </table>
7
8         <!-- Pagination Links -->
9         <form action="" class="pagination" {{ $data->links() }}></form>
10    </div>
11
12    <!-- Add CSS for styling -->
13    <style>
14        /* Container Styling */
15        div {
16            max-width: 100%;
17            margin: 0 auto;
18            padding: 20px;
19            font-family: Arial, sans-serif;
20        }
21
```

The code editor has syntax highlighting for Blade tags (e.g., `<div>`, `<table>`, `<tr>`, `<form>`) and CSS. The status bar at the top indicates the workspace is 'Laravel-WorkSpace'. The bottom right corner shows the standard VS Code status icons.

# Output

← → ⌂ 127.0.0.1:8000/list-employee 120% ☆ ⌂ ⌂ ⌂

## Employee List

Your Name:  Search

ID	Name	Email	Phone	Created_at	Updated_at	Operations
1	Yitayew Solomon	yitayewsolomon3@gmail.com	0937559718	2024-11-12 00:00:00	2024-11-12 00:00:00	<button style="background-color: red; color: white; padding: 5px 10px; border: none;">Delete</button> <button style="background-color: #4f81bd; color: white; padding: 5px 10px; border: none;">Edit</button>
2	Haileab Solomon	haileabsolomon@gmail.com	0937559718	2024-11-12 00:00:00	2024-11-12 00:00:00	<button style="background-color: red; color: white; padding: 5px 10px; border: none;">Delete</button> <button style="background-color: #4f81bd; color: white; padding: 5px 10px; border: none;">Edit</button>
3	Yared Solomon	yaredsolomon@gmail.com	0937559718	2024-11-12 00:00:00	2024-11-12 00:00:00	<button style="background-color: red; color: white; padding: 5px 10px; border: none;">Delete</button> <button style="background-color: #4f81bd; color: white; padding: 5px 10px; border: none;">Edit</button>

Showing 1 to 3 of 5 results

« Previous Next »

1 2 3

# Exercises

- **Perform CRUD Operation for Other Multi Medea Documents**
  - 1. Image**
  - 2. Audio**
  - 3. Video**
  - 4. Others**

# Stub in Laravel

- A stub in Laravel is a pre-defined template or boilerplate code used to generate specific file types or components of the framework, such as controllers, models, or other classes.
- Stubs help speed up the development process by providing a base structure for these components, which can then be customized according to the application's needs.

# Customizing Stubs

Laravel allows you to customize stubs if you need specific changes to the generated files.

1. **Publish Stubs:** To customize stubs, you need to publish them first. Use the following command:

bash

 Copy code

```
php artisan stub:publish
```

This will copy the default stubs to the `stubs` directory at the root of your project.

2. **Edit the Stubs:** Navigate to the `stubs` directory in your project and edit the required stub file.

For example:

- `controller.stub` : For controllers
- `model.stub` : For models
- `migration.stub` : For migrations

# Migration In Laravel

- In Laravel, migration is a way to manage and version-control your database schema. Migrations allow developers to define and modify database structures in a programmatic and version-controlled manner, making it easy to synchronize the database schema across different environments.

# Benefits of Migrations

- ❖ **Version Control:** Track changes to the database schema over time.
- ❖ **Team Collaboration:** Makes it easy for teams to work on the same database structure.
- ❖ **Rollback Support:** Easily undo changes to the database.
- ❖ **Automated Deployment:** Define schema changes in code and execute them during deployment.

# Creating a Migration

## Creating a Migration

To create a migration file, use the Artisan command:

bash

 Copy code

```
php artisan make:migration create_table_name_table
```

This generates a migration file in the `database/migrations` directory with a timestamp in the filename, ensuring order.

For example:

bash

 Copy code

```
php artisan make:migration create_employees_table
```

# Cont. ...

```
class CreateEmployeesTable extends Migration
{
    public function up()
    {
        Schema::create('employees', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->string('phone');
            $table->timestamps(); // Adds `created_at` and `updated_at` columns
        });
    }

    public function down()
    {
        Schema::dropIfExists('employees');
    }
}
```

 Copy code

# Cont. ...

## Running Migrations

To execute all pending migrations and apply changes to the database:

bash

 Copy code

```
php artisan migrate
```

This command reads the migration files and updates the database schema.

---

## Rolling Back Migrations

To undo the last batch of migrations:

bash

 Copy code

```
php artisan migrate:rollback
```

# Cont. ...

To reset all migrations (undo all changes):

```
bash
```

 Copy code

```
php artisan migrate:reset
```

To rollback and reapply all migrations in a single command:

```
bash
```

 Copy code

```
php artisan migrate:refresh
```

# Migration Commands Summary

## Migration Commands Summary

Command	Description
<code>php artisan make:migration</code>	Create a new migration file.
<code>php artisan migrate</code>	Run all pending migrations.
<code>php artisan migrate:rollback</code>	Rollback the last batch of migrations.
<code>php artisan migrate:reset</code>	Rollback all migrations.
<code>php artisan migrate:refresh</code>	Rollback and reapply all migrations.
<code>php artisan migrate:fresh</code>	Drop all tables and <b>rerun all</b> migrations.

# Seeding in laravel

- In Laravel, seeding is the process of **populating your database** with sample or default data. Seeders are classes that allow you to insert records into your database tables.
- This is particularly useful when you need to populate the database with data during **development** or **testing**, or if you need to pre-populate certain fields in production.

# Benefits of Seeding

- ❖ **Quick Data Population:** Allows you to quickly insert sample data into your database for testing and development.
- ❖ **Reusable:** Seeders can be re-run without affecting your production data (with proper configuration).
- ❖ **Testing:** Ensures that your application works with real-like data during the testing phase.

# Cont. ...

## Creating a Seeder

To create a new seeder, use the Artisan command:

```
bash
```

 Copy code

```
php artisan make:seeder SeederName
```

For example, to create a seeder for employees:

```
bash
```

 Copy code

```
php artisan make:seeder EmployeesTableSeeder
```

# Cont. ...

This creates a seeder file inside the `database/seeders` directory:

```
php
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;

class EmployeesTableSeeder extends Seeder
{
    public function run()
    {
        DB::table('employees')->insert([
            'name' => 'John Doe',
            'email' => 'john.doe@example.com',
            'phone' => '1234567890',
        ]);
    }
}
```

 Copy code

# Cont. ...

## Summary of Seeder Commands

Command	Description
<code>php artisan make:seeder SeederName</code>	Create a new seeder class.
<code>php artisan db:seed</code>	Run all seeders defined in <code>DatabaseSeeder</code> .
<code>php artisan db:seed --class=SeederName</code>	Run a specific seeder.
<code>php artisan migrate --seed</code>	Run migrations and seed the database.
<code>php artisan make:factory FactoryName --model=ModelName</code>	Create a new factory.

# Thank you!

Appreciate your action.