



April 28, 2023 2 min read views 994 [Sasha Bondar](#) replies: 1 [Engineering](#) [PHP](#)  49

# Building a PHP MVC Framework from Scratch

In this tutorial, you will learn how to build a PHP Model-View-Controller (MVC) framework from scratch. MVC is a design pattern commonly used in web development to separate the application logic, data, and presentation. By following this tutorial, you will enhance your understanding of the MVC pattern and improve your skills as a PHP developer. If you are looking to [hire PHP developers](#), having a solid understanding of MVC frameworks will be beneficial.

## Prerequisites

To follow this tutorial, you should have a basic understanding of PHP and object-oriented programming. Familiarity with Composer, the PHP package manager, is also helpful.

## Step 1: Setting up the Project

Create a new directory for your project and navigate to it using the command line:

```
mkdir php-mvc-framework
cd php-mvc-framework
```

Initialize a new Composer project:

```
composer init
```

Follow the prompts to set up your project. When asked for dependencies, leave it empty for now.

## Step 2: Creating the Directory Structure

Create the following directory structure for your project:

```
src/
  Controllers/
  Models/
  Views/
```

## Step 3: Building the Core Components

### Router

Create a new file named `Router.php` in the `src/` directory. This file will contain the main routing logic for your framework.

```
<?php

namespace MVC;

class Router {
    protected $routes = [];

    public function addRoute($route, $controller, $action)
    {
        $this->routes[$route] = ['controller' => $controller, 'action' => $action];
    }

    public function dispatch($uri) {
        if (array_key_exists($uri, $this->routes)) {
            $controller = $this->routes[$uri]['controller'];
            $action = $this->routes[$uri]['action'];

            $controller = new $controller();
            $controller->$action();
        } else {
            throw new \Exception("No route found for URI: $uri");
        }
    }
}
```

## Base Controller

Create a new file named `Controller.php` in the `src/` directory. This file will contain the base controller class that all other controllers will extend.

```
<?php

namespace MVC;

class Controller {
    protected function render($view, $data = []) {
        extract($data);

        include "Views/$view.php";
    }
}
```

## Step 4: Implementing an Example Application

### Creating a Model

Create a new file named `User.php` in the `src/Models/` directory. This model will represent a user in your application.

```
<?php

namespace MVC\Models;

class User {
    public $name;
    public $email;

    public function __construct($name, $email) {
        $this->name = $name;
        $this->email = $email;
    }
}
```

## Creating a Controller

Create a new file named `UserController.php` in the `src/Controllers/` directory. This controller will handle user-related actions.

```
<?php

namespace MVC\Controllers;

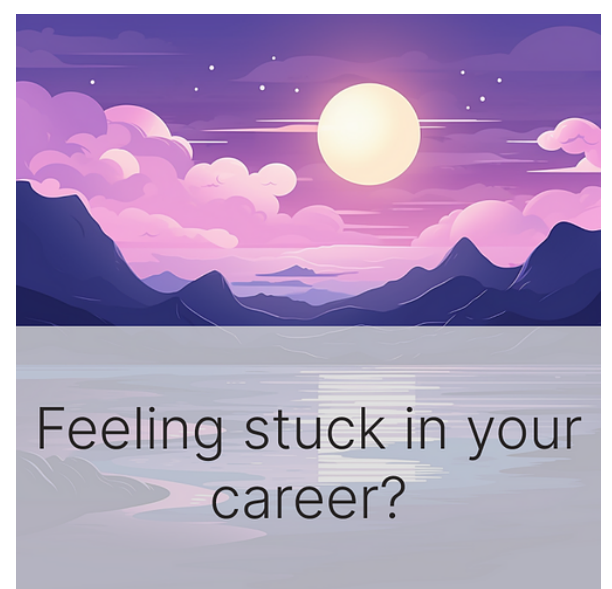
use MVC\Controller;
use MVC\Models\User;

class UserController extends Controller {
    public function index() {
        $users = [
            new User('John Doe', 'john@example.com'),
            new User('Jane Doe', 'jane@example.com')
        ];

        $this->render('user/index', ['users' => $users]);
    }
}
```

## Creating a View

Create a new file named `index.php` in the `src/Views/user/` directory. This view will display a list of users.



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>User List</title>
</head>
<body>
  <h1>User List</h1>
  <ul>
    <?php foreach ($users as $user): ?>
      <li><?= $user->name ?> (<?= $user->email ?>)</li>
    <?php endforeach; ?>
  </ul>
</body>
</html>
```

## Setting up Routes

Create a new file named `routes.php` in the `src/` directory. This file will define the routes for your application.

```
<?php

use MVC\Router;
use MVC\Controllers\UserController;

$route = new Router();

$route->addRoute('/', UserController::class, 'index');
```

## Testing the Application

Create a new file named `index.php` in the root directory of your project. This file will serve as the entry point for your application.

```
<?php

require 'vendor/autoload.php';

$uri = $_SERVER['REQUEST_URI'];

$route = require 'src/routes.php';
$route->dispatch($uri);
```

Run the built-in PHP web server to test your application:

```
php -S localhost:8000
```

Visit <http://localhost:8000> in your web browser to see the user list.

# Conclusion

By following this tutorial, you have successfully built a simple PHP MVC framework from scratch. This knowledge will serve as a strong foundation for building more complex applications or even contributing to existing PHP MVC frameworks. If you are looking to [hire PHP developers](#), having a solid understanding of MVC frameworks is essential for success in the field of web development.

Share: 

FacebookTwitterLinkedIn

49

## Are you an engineering manager tired of recruiting?

Checkout out how we can help you to focus on delivering technical excellence and growing your product by hiring remote developers and creating high-performing teams.

Learn more


Comments 1

Name:

Email:

Write a comment..

Create comment



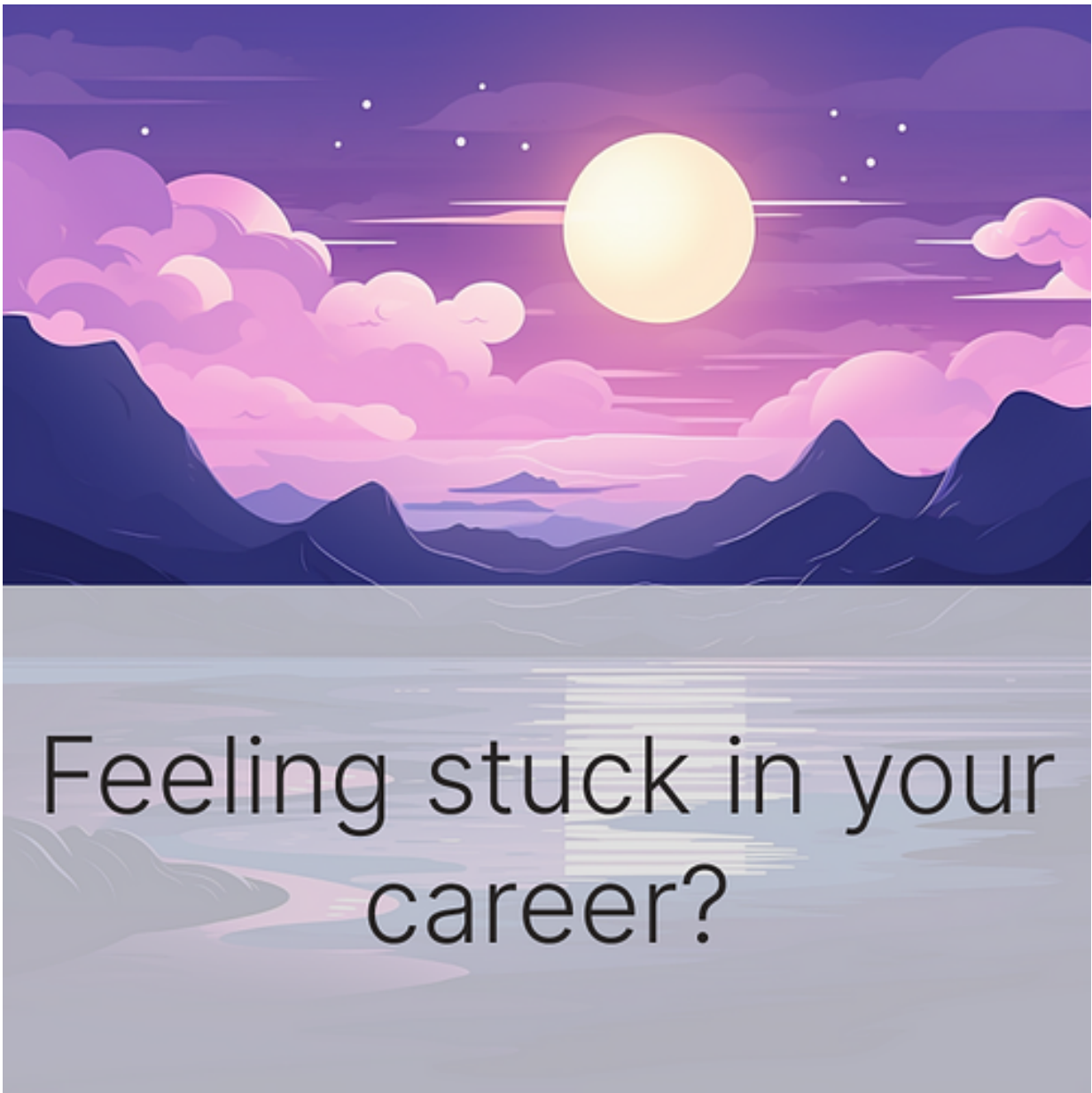
Shola Omofola 8 days ago

Very insightful.

A couple of observations.

1. After running composer init , set your namespace to "MVC"

2. Movee "\$router->dispatch(\$uri);" from index.php to routes.php to prevent Error: Call to a member function dispatch() on integer in



---

Media

Categories

- [Recruiting](#)
- [Engineering](#)
- [Career](#)
- [Managing](#)
- [Soft Skills](#)
- [Success stories](#)

Social Media

- [Instagram](#)
- [Facebook](#)
- [LinkedIn](#)
- [Twitter](#)

Apply as Developer

[Apply](#)

Contact us

[Send us a message](#)