Creating RESTful APIs in CodeIgniter

Topics : Codeigniter
Written on March 01, 2024

To create RESTful APIs in CodeIgniter, you can follow these steps:

1. Set Up CodeIgniter:

- Install and set up CodeIgniter framework on your server or local environment if you haven't already.

2. Configure Routes:

- Define routes in `application/config/routes.php` to map API endpoints to controller methods.

```
$route['api/users'] = 'api/UsersController';
```

3. Create Controllers:

- Create a controller for each resource you want to expose via the API.
- Implement methods for handling CRUD operations (e.g., index, show, store, update, delete).

```
class UsersController extends CI_Controller {
    public function index() {
        // Get all users from the database
    }

    public function show($id) {
        // Get a single user by ID
    }

    public function store() {
        // Create a new user
    }

    public function update($id) {
        // Update an existing user
    }

    public function delete($id) {
        // Delete a user
```

```
    }
}
```

4. Implement CRUD Operations:

- Use CodeIgniter's database library or any ORM library (e.g., Eloquent, Doctrine) to perform CRUD operations on your database.

5. Format Responses:

- Format responses as JSON or XML using CodeIgniter's output library or by directly echoing JSON-encoded data.

```
 $this->output
    ->set_content_type('application/json')
    ->set_output(json_encode($data));
```

6. Handle Request Methods:

- Use CodeIgniter's input class to access request data and determine the HTTP method (GET, POST, PUT, DELETE).
- Route requests to appropriate controller methods based on the HTTP method.

```
$http_method = $this->input->method();
switch ($http_method) {
    case 'get':
        // Handle GET request
        break;
    case 'post':
        // Handle POST request
        break;
    case 'put':
        // Handle PUT request
        break;
    case 'delete':
        // Handle DELETE request
        break;
    default:
        // Handle unsupported request
        break;
}
```

7. Input Validation:

- Use CodeIgniter's form validation library or manual validation to validate input data and ensure data integrity.

8. Secure Your API:

- Implement authentication and authorization mechanisms to secure your API endpoints.
- Use HTTPS to encrypt data transmitted between clients and the server.