ARYATECHNO

Creating and managing Controllers

Topics : Codeigniter
Written on February 29, 2024

1. Create a New Controller:

1. Navigate to the `application/controllers` directory in your CodeIgniter project.
2. Create a new PHP file for your controller. The file name should match the controller class name.
3. Define your controller class, which should extend `CI_Controller`.
4. Implement methods within the controller class to handle different actions or pages in your application.

Example:

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class Welcome extends CI_Controller {

    public function index() {
        // Controller method logic for the index page
    }

    public function about() {
        // Controller method logic for the about page
    }

    // Add more methods as needed
}
?>
```

2. Routing Requests to Controllers:

- By default, CodeIgniter routes requests to controllers based on the URI segment after the base URL.
- For example, if your base URL is `http://localhost/myapp/`, a request to `http://localhost/myapp/welcome/about` will route to the `about()` method in the `Welcome` controller.
- You can customize routing in the `application/config/routes.php` file if needed.

3. Loading Models and Views:

- Within your controller methods, you can load models and views as needed to interact with the database and render HTML output.
- Use `$this->load->model()` to load models and `$this->load->view()` to load views.
- Pass data to views using the second parameter of the `load->view()` method as an associative array.

Example:

```
<?
public function index() {
    // Load model
    $this->load->model('welcome_model');
    // Call model method to retrieve data
    $data['message'] = $this->welcome_model->get_message();
    // Load view with data
    $this->load->view('welcome_message', $data);
}
?>
```

4. Managing Controllers:

- Organize your controllers logically based on the functionality they provide.
- Avoid creating overly large controllers by separating related functionality into different controllers.
- Reuse code by creating base controllers or using CodeIgniter's modular structure.
- Ensure proper access control and security measures within your controllers to protect sensitive data and prevent unauthorized access.

5. Testing:

- Test your controllers by accessing the corresponding URLs in your browser or through API clients like Postman.
- Verify that the expected data is retrieved from models and passed to views correctly.
- Debug any errors or issues encountered during testing and refine your controller logic as needed.