ARYATECHNO

Introduction to Models in CodeIgniter

Topics : Codeigniter
Written on February 29, 2024

In CodeIgniter, models play a crucial role in managing the application's data logic, interacting with the database, and performing various data-related operations. Models represent the M (Model) component in the MVC (Model-View-Controller) architecture, responsible for handling data manipulation, retrieval, and validation. Here's an introduction to models in CodeIgniter:

Purpose of Models:

- Models in CodeIgniter encapsulate the business logic and data manipulation logic of the application.
- They interact with the database to perform CRUD (Create, Read, Update, Delete) operations on data.
- Models can also contain validation rules to ensure data integrity and enforce business rules.
- Models facilitate the separation of concerns, allowing controllers to focus on application flow and views to focus on presentation.

Creating Models:

- Models in CodeIgniter are typically stored in the `application/models` directory.
- Each model is defined as a PHP class that extends the `CI_Model` class provided by CodeIgniter.
- You can create multiple models to handle different aspects of your application's data logic.

Example model file (`User_model.php`):

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

class User_model extends CI_Model {
    public function get_user($user_id) {
        // Database query to retrieve user data
        return $query->row();
    }
    public function create_user($data) {
        // Database query to insert new user data
        $this->db->insert('users', $data);
        return $this->db->insert_id();
    }
    // Add more methods for other data operations
```

```
}
?>
```

Loading Models:

- Models are loaded from within controller methods or other models using CodeIgniter's built-in Loader class.
- You can load models in controllers, libraries, or other models using the `$this->load->model()` method.

Example controller method:

```
public function index() {
    // Load model
    $this->load->model('user_model');
    // Call model method to retrieve user data
    $user = $this->user_model->get_user(123);
    // Use retrieved user data in controller
}
```

Using Models:

- Once loaded, you can call methods defined in the model to perform data operations such as fetching user data, creating new records, updating existing records, or deleting records.
- Models typically encapsulate database queries and other data-related operations, abstracting them away from the controller or view layer.

Best Practices:

- Keep models focused on data-related operations and business logic. Avoid including presentation logic or view-related code in models.
- Use models to encapsulate complex data operations and database interactions, keeping controllers lean and focused on application flow.
- Validate data and enforce business rules within models to ensure data integrity and security.