Connecting CodeIgniter with databases

Topics : Codeigniter
Written on February 29, 2024

Connecting CodeIgniter with databases is a fundamental aspect of building dynamic web applications. CodeIgniter provides a convenient database abstraction layer that allows you to interact with various database systems using a unified interface. Here's how you can connect CodeIgniter with databases:

1. Configure Database Settings:

- Open the `application/config/database.php` file in your CodeIgniter project.
- Set up the database connection parameters such as hostname, username, password, database name, and other options based on your database server configuration.

Example configuration for MySQL:

```
$db['default'] = array(
    'dsn'   => '',
    'hostname' => 'localhost',
    'username' => 'your_username',
    'password' => 'your_password',
    'database' => 'your_database_name',
    'dbdriver' => 'mysqli',
    'dbprefix' => '',
    'pconnect' => FALSE,
    'db_debug' => (ENVIRONMENT !== 'production'),
    'cache_on' => FALSE,
    'cachedir' => '',
    'char_set' => 'utf8',
    'dbcollat' => 'utf8_general_ci',
    'swap_pre' => '',
    'encrypt' => FALSE,
    'compress' => FALSE,
    'stricton' => FALSE,
    'failover' => array(),
    'save_queries' => TRUE
);
```

2. Loading the Database Library:

- CodeIgniter provides a database library that simplifies database operations.

- The database library is loaded automatically when you autoload libraries in your `application/config/autoload.php` file.
- Alternatively, you can load the database library manually in your controller or model using the `$this->load->database()` method.

Example of manual loading in a controller:

```
$this->load->database();
```

3. Performing Database Queries:

- Once the database library is loaded, you can use its methods to perform database queries such as selecting, inserting, updating, and deleting data.
- CodeIgniter provides a query builder class that allows you to build SQL queries using a fluent interface or execute raw SQL queries.

Example of using the query builder:

```
$query = $this->db->get('users');
foreach ($query->result() as $row) {
    echo $row->username;
}
```

Example of executing a raw SQL query:

```
$query = $this->db->query('SELECT * FROM users');
foreach ($query->result() as $row) {
    echo $row->username;
}
```

4. Using Active Record:

- CodeIgniter's Active Record class provides a convenient way to perform database operations using object-oriented syntax.
- Active Record allows you to chain methods to build complex queries and helps prevent SQL injection attacks.

Example of using Active Record:

```
$this->db->select('username, email');
$this->db->from('users');
$this->db->where('status', 'active');
$query = $this->db->get();
```

5. Handling Database Errors:

- CodeIgniter provides error handling mechanisms for database operations, allowing you to handle errors gracefully.
- You can enable or disable database debugging in the `database.php` configuration file based on your environment.

6. Important Notes:

- Always validate and sanitize user input to prevent SQL injection and other security vulnerabilities.
- Use transactions for atomic operations that involve multiple database queries.
- Optimize database queries for performance by indexing frequently queried columns and minimizing unnecessary queries.