Documentation for Using the ORM to Create a Table and Manipulate Data

Introduction

This document outlines how to use the ORM (Object-Relational Mapping) in PHP to create a table in a MySQL database, as well as perform CRUD (Create, Read, Update, Delete) operations on this table. The example provided is the creation and manipulation of a Users table.

Initial Configuration

Step 1: Database Configuration

Create a file Config.php containing the database connection information.

Config.php

```php
<?php
// Config.php
return [
    'host' => 'localhost',
    'dbname' => 'DBmaster',
    'user' => 'root',
    'password' => '',
];
?>
```

Step 2: Database Connection

Create a file Database.php to manage the database connection.

Database.php

```php
<?php
// Database.php
class Database {
    private static $pdo;

    public static function getConnection() {
        if (!self::$pdo) {
            $config = require 'Config.php';
            $dsn = "mysql:host={$config['host']};dbname={$config['dbname']}";
            self::$pdo = new PDO($dsn, $config['user'], $config['password']);
            self::$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        }
        return self::$pdo;
    }
}
?>
```

Creating and Manipulating the Users Table

Step 3: Define the Base ORM Class

Create a file ORM.php that implements the base ORM class.

Step 4: Define the User Class

Create a file User.php for the User class, inheriting from the ORM class.

User.php

```php
<?php
// User.php
require_once 'ORM.php';

class User extends ORM {
    protected static $table = 'Users';
    protected static $primaryKey = 'id';


    public function getUsername() {
        return $this->attributes['username'];
    }

    public function setUsername($username) {
        $this->attributes['username'] = $username;
    }

    public function getEmail() {
        return $this->attributes['email'];
    }

    public function setEmail($email) {
        $this->attributes['email'] = $email;
    }

    public function getPassword() {
        return $this->attributes['password'];
    }

    public function setPassword($password) {
        $this->attributes['password'] = sha1($password) ;
    }
    public static function addUserColumn($name, $type) {
        static::$columns[$name] = $type;
    }
```

Step 5: Create and Test the Users Table

Create a script file test-setup.php to set up and test the Users table.

```php
<?php
require_once 'Config.php';
require_once 'Database.php';
require_once 'ORMInterface.php';
require_once 'ORM.php';
require_once 'User.php';
// Add columns to the User table
User::addUserColumn('username', 'VARCHAR(255) NOT NULL');
User::addUserColumn('email', 'VARCHAR(255) NOT NULL');
User::addUserColumn('password', 'VARCHAR(255) NOT NULL');
User::addUserColumn('created_at', 'TIMESTAMP DEFAULT CURRENT_TIMESTAMP');

// Create the User table
echo "Creating User table...<br>";
if (User::setupTable()) {
    echo "User table created successfully.<br>";
} else {
    echo "Failed to create User table.<br>";
}
// Test adding a user
$user = new User([
    'username' => 'testuser',
    'email' => 'testuser@example.com',
    'password' => 'password123'
]);
if ($user->save()) {
    echo "User created successfully.<br>";
} else {
    echo "Failed to create user.<br>";
```

```php
}

// Test reading a user
$fetchedUser = User::find($user->attributes['id']);
if ($fetchedUser) {
    echo "User found: " . $fetchedUser->getUsername() . "<br>";
} else {
    echo "User not found.<br>";
}

// Test updating a user
$fetchedUser->setEmail('newemail@example.com');
if ($fetchedUser->save()) {
    echo "User updated successfully.<br>";
} else {
    echo "Failed to update user.<br>";
}

// Test deleting a user
if ($fetchedUser->delete()) {
    echo "User deleted successfully.<br>";
} else {
    echo "Failed to delete user.<br>";
}
```