**MVC**

**MVC (Model-View-Controller)** adalah sebuah pola arsitektur perangkat lunak yang membagi aplikasi menjadi tiga komponen utama: Model, View, dan Controller. Tujuan utama dari MVC adalah untuk memisahkan logika bisnis, presentasi, serta pengolahan permintaan dan respon, sehingga aplikasi menjadi lebih terstruktur, mudah dikembangkan, dan diuji. MVC sering digunakan dalam pengembangan web, desktop, dan aplikasi mobile, dan telah menjadi salah satu pola desain yang paling populer dalam pengembangan perangkat lunak.



Gambar 1 gambar model MVC

Berikut adalah penjelasan mengenai tiga komponen utama dalam arsitektur MVC:

**Model,** adalah bagian yang mengatur data dari proyek dan menyediakan fungsi untuk mengolah data yang ada. Dengan kata lain, bagian ini berfokus untuk data dan bukan untuk tugas lain seperti tampilan, dll.

**View,** bertanggung jawab untuk menampilkan data kepada pengguna dan menampilkan UI untuk berinteraksi. View akan menampilkan data yang diberikan oleh model. Fokus utama view adalah frontend, maka view tidak memiliki kode tentang logic data maupun kontrol untuk aplikasi.

**Controller,** merupakan bagian yang bertanggung jawab untuk menangani permintaan pengguna seperti menerima input, memproses permintaan, dan mengirimkan respon. Controller juga berfungsi untuk menghubungkan view dan model, controller berinteraksi dengan model untuk mendapatkan, memperbarui dan mengirim data. Output yang diberikan oleh model akan disampaikan kepada view untuk ditampilkan.

**Mengapa harus menggunakan MVC?**

Dengan memisahkan aplikasi menjadi tiga komponen, arsitektur MVC memungkinkan pengembangan yang lebih terstruktur, fleksibel, dan mudah dirawat. Setiap komponen memiliki tanggung jawab yang jelas, sehingga perubahan pada satu komponen tidak akan berdampak pada komponen lainnya.

**Manfaat lain menggunakan MVC**

- Pemisahan Kepentingan (Separation of Concerns):

MVC memisahkan aplikasi menjadi tiga komponen (Model, View, dan Controller) yang memiliki tanggung jawab yang jelas. Ini membantu mempertahankan kode yang bersih, terorganisir, dan mudah dipahami.

- Pengembangan yang Lebih Mudah:

Karena komponen-komponen dalam MVC saling terpisah, pengembang dapat bekerja pada setiap komponen secara independen. Perubahan pada satu komponen tidak akan mempengaruhi komponen lainnya, sehingga mempercepat proses pengembangan.

- Maintainability dan Fleksibilitas:

Struktur MVC membuat kode lebih mudah untuk dipelihara dan diperbarui di masa mendatang dan memuudahkan untuk mengganti atau memperbarui komponen tertentu (misalnya, mengubah tampilan tanpa mempengaruhi logika bisnis).

- Reusability:

Komponen-komponen dalam MVC dapat digunakan kembali di proyek lain dengan sedikit atau tanpa modifikasi, hal ini meningkatkan efisiensi dan mengurangi waktu pengembangan.

- Testability:

Karena komponen-komponen dalam MVC terpisah, unit testing dan pengujian terintegrasi menjadi lebih mudah dilakukan.

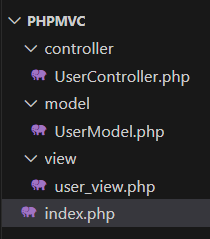
- Scalability:

Arsitektur MVC memungkinkan aplikasi untuk diskalakan secara horizontal dengan menambahkan instance tambahan dari komponen tertentu (misalnya, menambahkan server aplikasi atau server database).

- Pemisahan Tugas dan Tanggung Jawab:

Dalam MVC, setiap komponen memiliki tugas dan tanggung jawab yang jelas, membantu mengatur kompleksitas aplikasi, hal ini memudahkan pengaturan dan koordinasi tim pengembangan yang terlibat dalam proyek.

**Contoh Proyek MVC PHP Dasar**



Gambar 2 Susunan folder dan file PHP

1) Buatlah sebuah proyek baru bernama “PHPMVC” kemudian tambahkan 3 folder yang bernama “controller”, “model”, dan “view” serta masing-masing file php dengan susunan seperti gambar diatas.

2) Saat ini kita akan membuat sebuah contoh proyek dimana output dari proyek ini adalah data user (meliputi ID, Nama, dan Email). Masukan kode berikut pada file UserController.php:

<?php

class UserPresenter

{

    private $model;

    private $view;

    public function \_\_construct(UserModel $model, UserView $view)

    {

        $this->model = $model;

        $this->view = $view;

    }

    public function showUserInfo()

    {

        $this->view->displayUserInfo(

            $this->model->getId(),

            $this->model->getName(),

            $this->model->getEmail()

        );

    }

}

?>

3) Masukan kode berikut pada UserModel.php:

<?php

class UserModel

{

    private $id;

    private $name;

    private $email;

    public function \_\_construct($id, $name, $email)

    {

        $this->id = $id;

        $this->name = $name;

        $this->email = $email;

    }

    public function getId()

    {

        return $this->id;

    }

    public function getName()

    {

        return $this->name;

    }

    public function getEmail()

    {

        return $this->email;

    }

}

?>

4) Masukan kode berikut pada user\_view.php:

<?php

    class UserView

    {

        public function displayUserInfo($id, $name, $email)

        {

            echo "User ID: " . $id . "<br>";

            echo "Name: " . $name . "<br>";

            echo "Email: " . $email . "<br>";

        }

    }

?>

5) Masukan kode berikut pada index.php:

<?php

require\_once './model/UserModel.php';

require\_once './controller/UserController.php';

require\_once './view/user\_view.php';

$user = new UserModel(1, 'John Doe', 'john.doe@example.com');

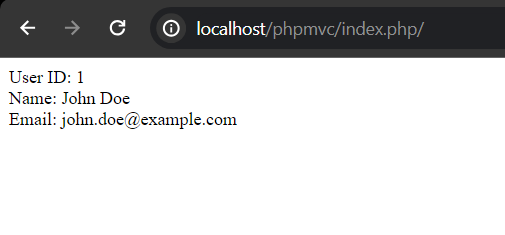
$userView = new UserView();

$userPresenter = new UserPresenter($user, $userView);

$userPresenter->showUserInfo();

?>

6) Selanjutnya tahap pengetesan, pastikan untuk menyalakan “Apache” pada XAMPP dan memasukan url dengan benar, maka output nya akan menjadi seperti ini:



**Penjelasan**

- UserModel.php adalah kelas yang mewakili model data pengguna, dengan properti id, name, dan email.

- UserController.php adalah kelas presenter yang menerima model dan view, dan bertanggung jawab untuk menghubungkan keduanya.

- user\_view.php adalah kelas yang bertanggung jawab untuk menampilkan informasi pengguna.

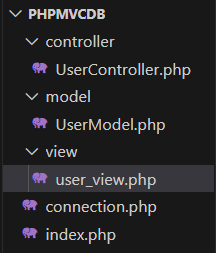
- index.php adalah file utama yang menginisialisasi model, view, dan presenter, serta memanggil metode showUserInfo() dari presenter.

File-file tersebut berkolabolasi untuk menghasilkan output yang diharapkan. Kalian dapat mengatur output yang dikeluarkan dengan mengedit bagian index.php berikut:



**Contoh Proyek MVC PHP Database**

Setelah kita mencoba membuat proyek MVC sederhana, sekarang kita akan mencoba membuat sebuah proyek MVC PHP dengan database. Untuk proyek ini, kita akan mencoba meng-implementasikan ‘create’ dan ‘read’ pada mySQL database.



1) Mirip seperti sebelumnya, untuk membuat proyek ini kita akan membuat sebuah folder dan file seperti berikut. Jangan lupa untuk membuat file “connection.php” untuk menyimpan kode penghubung kedalam database.

2) Step kedua adalah mempersiapkan koneksi antara database dan proyek MVC PHP kita. masukan kode berikut kedalam connection.php:

<?php

    $host = "localhost";

    $port = 3306;

    $db = "phpmvc";

    $user = "root";

    $pwd = "";

    $pdo = new PDO("mysql:host=$host:$port;dbname=$db",$user,$pwd);

Pastikan kamu memasukan port, user, dan database dengan tepat.

3) Masukan kode berikut pada user\_view.php:

<?php

    // File: View/UserView.php

    class UserView {

        public function displayUsers($users) {

            echo "<h1>All Users</h1>";

            echo "<ul>";

            foreach ($users as $user) {

                echo "<li>ID: {$user['id']}, Name: {$user['name']}, Email: {$user['email']}</li>";

            }

            echo "</ul>";

        }

        public function displayUser($user) {

            echo "<h1>User Details</h1>";

            echo "ID: {$user['id']}<br>";

            echo "Name: {$user['name']}<br>";

            echo "Email: {$user['email']}<br>";

        }

        public function showCreateUserForm() {

            echo "<h1>Create User</h1>";

            echo "<form method='post' action='index.php?action=create'>";

            echo "Name: <input type='text' name='name'><br>";

            echo "Email: <input type='email' name='email'><br>";

            echo "<input type='submit' value='Create'>";

            echo "</form>";

        }

        public function showSuccessMessage($message) {

            echo "<div class='success'>{$message}</div>";

        }

        public function showErrorMessage($message) {

            echo "<div class='error'>{$message}</div>";

        }

    }

?>

4) Masukan kode berikut pada UserController.php:

<?php

    // File: Presenter/UserPresenter.php

    class UserPresenter {

        private $model;

        private $view;

        public function \_\_construct(UserModel $model, UserView $view) {

            $this->model = $model;

            $this->view = $view;

        }

        public function getAllUsers() {

            $users = $this->model->getAllUsers();

            $this->view->displayUsers($users);

        }

        public function getUserById($id) {

            $user = $this->model->getUserById($id);

            $this->view->displayUser($user);

        }

        public function createUser($name, $email) {

            $success = $this->model->createUser($name, $email);

            if ($success) {

                $this->view->showSuccessMessage("User created successfully.");

            } else {

                $this->view->showErrorMessage("Failed to create user.");

            }

        }

    }

?>

5) Masukan kode berikut pada UserModel.php:

<?php

    // File: Model/UserModel.php

    class UserModel {

        private $pdo;

        public function \_\_construct($pdo) {

            $this->pdo = $pdo;

        }

        public function getAllusers() {

            $stmt = $this->pdo->prepare("SELECT \* FROM user");

            $stmt->execute();

            return $stmt->fetchAll(PDO::FETCH\_ASSOC);

        }

        public function getUserById($id) {

            $stmt = $this->pdo->prepare("SELECT \* FROM user WHERE id = :id");

            $stmt->bindParam(':id', $id, PDO::PARAM\_INT);

            $stmt->execute();

            return $stmt->fetch(PDO::FETCH\_ASSOC);

        }

        public function createUser($name, $email) {

            $stmt = $this->pdo->prepare("INSERT INTO user (name, email) VALUES (:name, :email)");

            $stmt->bindParam(':name', $name, PDO::PARAM\_STR);

            $stmt->bindParam(':email', $email, PDO::PARAM\_STR);

            return $stmt->execute();

        }

    }

?>

6) Masukan kode berikut pada index.php:

<?php

    require\_once 'connection.php';

    require\_once './model/UserModel.php';

    require\_once './controller/UserController.php';

    require\_once './view/user\_view.php';

    $model = new UserModel($pdo);

    $view = new UserView();

    $presenter = new UserPresenter($model, $view);

    if ($\_SERVER['REQUEST\_METHOD'] === 'POST' && isset($\_GET['action']) && $\_GET['action'] === 'create') {

        $name = $\_POST['name'];

        $email = $\_POST['email'];

        $presenter->createUser($name, $email);

    } else {

        // Get all users

        $presenter->getAllUsers();

        // Display the create user form

        $view->showCreateUserForm();

    }

?>

7) Untuk step terakhir, kita akan membuat sebuah database pada mySQL. Gunakan perintah berikut untuk dieksekusi pada Query:

CREATE TABLE user (

id int(11) NOT NULL,

name varchar(50) NOT NULL,

email varchar(50) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_general\_ci;

INSERT INTO user (id, name, email) VALUES

(1, 'johan', 'johanbaru@gmail.com'),

(2, 'Hylos', 'hylos@gmail.com'),

(3, 'Kipli', 'peashooter@gmail.com'),

(4, 'Bobi', 'bobibola@gmail.com'),

(5, 'Ihza', 'Captain@gmail.com');

ALTER TABLE user

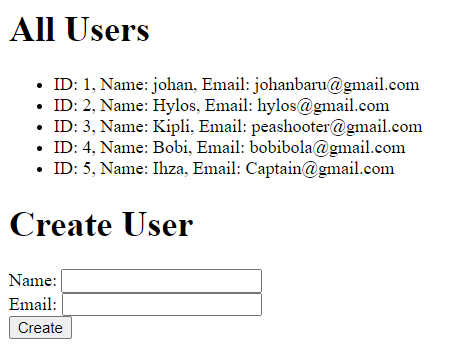
ADD PRIMARY KEY (id);

ALTER TABLE user

MODIFY id int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=6;

COMMIT;

8) Berikut adalah hasilnya. Kamu dapat menambahkan nama dan email baru untuk dimasukan kedalam database.



**Penjelasan**

- File connection.php berisi kode untuk menghubungkan proyek dengan database.

- UserModel.php berisikan kode untuk mengambil dan membuat data untuk database. function ‘getAllusers()’ digunakan untuk mengambil data dan ‘createUser()’ digunakan untuk membuat data.

- Data yang diambil akan ditampilkan pada user\_view.php.