MVC
(Model View Controller)

Definisi MVC

Pola MVC ini sudah dirumuskan sejak akhir tahun 1970. Sebuah pola arsitektur perangkat lunak yang memisahkan antara presentasi data dengan metode yang berinteraksi dengan data tersebut.

Penjelasan Komponen-komponen MVC

MVC, singkatan dari Model-View-Controller, memisahkan aplikasi kedalam tiga bagian yaitu bagian Model, View, dan Controller.

Model

Komponen ini berinteraksi dengan database, bertugas untuk menerima, menyimpan dan mengambil data dari database untuk pengguna. Komponen ini menanggapi permintaan data dan mengembalikan data yang diminta.

Aspek penting dari komponen model adalah secara teknis komponen ini ‘buta’, artinya komponen ini tidak tahu apa yang terjadi dengan data ketika telah diberikan ke komponen View atau Controller. Tugasnya murni memproses data ke penyimpanan atau mencari dan menyiapkan data untuk disampaikan ke komponen yang lain.

Komponen model tidak bisa disimpulkan sebagai sebuah database begitu saja. Model berperan sebagai penjaga data dan menerima semua permintaan tanpa bertanya.

View

Menampilkan informasi kepada pengguna dan mengintegrasikan data dari controller. bertanggung jawab untuk memformat dan menampilkan data.

View menampilkan data yang diminta dari model. Biasanya dalam aplikasi web yang dibangun dengan MVC, View adalah bagian dari aplikasi yang menghasilkan dan menampilkan HTML. Komponen view menerima aksi dari pengguna lalu berinteraksi dengan Controller. Contoh sederhana adalah sebuah tombol yang dihasilkan oleh View yang ketika diklik akan memicu aksi di dalam Controller.

Ada beberapa pemahaman yang keliru tentang komponen ini, khususnya oleh para pengembang aplikasi web yang menggunakan MVC ini. Sebagai contoh, banyak kesalahan karena komponen View dibuat tidak memiliki koneksi apa pun dengan Model dan bahwa semua data yang ditampilkan oleh komponen View dihasilkan dari Controller. Bentuk aliran proses ini mengabaikan teori di balik pola MVC sepenuhnya, bahwa:

“Untuk menerapkan MVC secara benar, tidak boleh ada interaksi antara komponen View dengan Model, semua interaksi harus ditangani Controller”

Penting juga untuk diingat bahwa komponen View tidak pernah memperoleh data dari Controller. Seperti disebut saat membahas Model, tidak ada hubungan langsung antara View dan Controller tanpa Model di antaranya.

Controller

Bertanggung jawan mengirim dan menerima data dari Model dan menyampaikannya ke komponen View. Komponen ini menanggapi permintaan (request) URL dan mengembalikan halaman yang diminta.

Controller bertugas menangani data yang diinput pengguna lalu mengupdate model berdasarkan input tersebut. Fungsi Controller tergantung pada pengguna, tanpa interaksi dari pengguna, Controller tidak dapat bekerja.

Secara ringkas, Controller dapat disimpulkan sebagai pengumpul informasi, yang disampaikannya ke Model untuk dikelola di tempat penyimpanan. Controller tidak berisi kode apapun kecuali kode untuk mengumpulkan masukan pengguna. Controller terhubung ke sebuah view dan Model tunggal yang membentuk sistem aliran data satu arah.

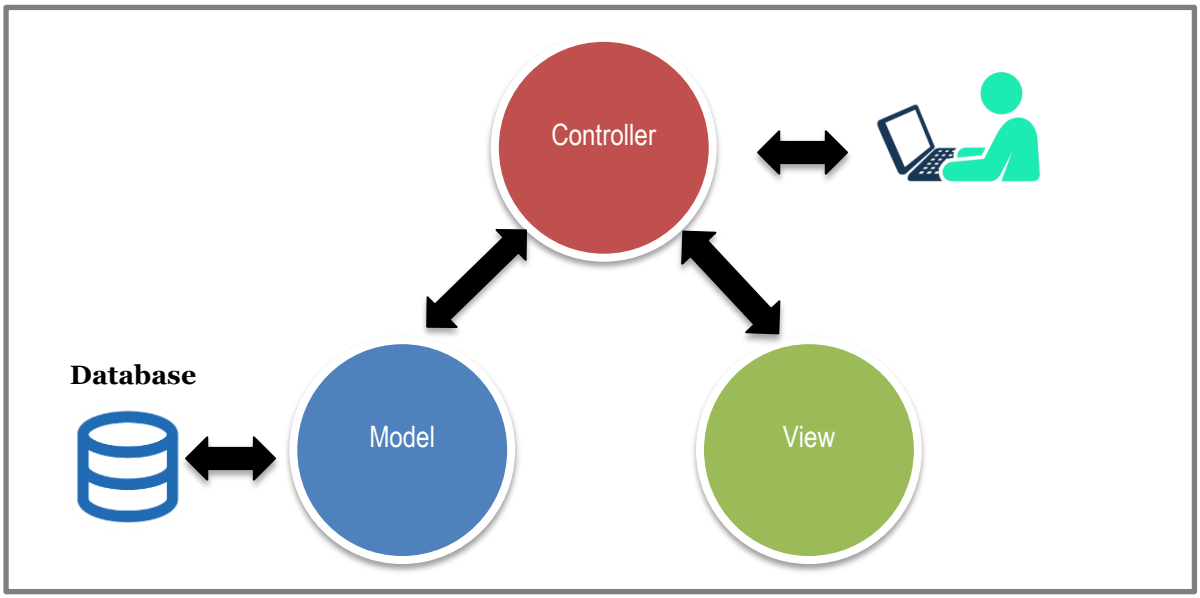
Penting untuk diingat bahwa Controller hanya melakukan tugasnya setelah pengguna berinteraksi dengan komponen View. Controller akan bergerak

setelah adanya interaksi pengguna dengan View. Kesalahan paling umum yang dibuat oleh pengembang adalah menganggap Controller sebagai pintu masuk (gateway), dan pada akhirnya menetapkan fungsi dan tanggung jawab pada Controller yang seharusnya menjadi tanggung jawab komponen View (ini biasanya merupakan hasil dari pengembang yang sama yang keliru memahami komponen View hanya sebagai template). Selain itu, merupakan kesalahan umum yang memberi Controller fungsi yang memberinya tanggung jawab penuh untuk mengolah, meneruskan, dan memproses data dari Model ke View, sedangkan dalam pola MVC hubungan ini harus dijaga antara Model dan View.

(Sumber: https://www.sitepoint.com/the-mvc-pattern-and-php-1/)

Cara Kerja MVC

Bagaimana MVC bekerja? Berikut penjelasannya:



- Pengguna berinteraksi dengan view, entah mengklik sebuah link atau mensubmit sebuah form
- Controller menangani masukan pengguna lalu meneruskan informasi tersebut ke model
- Model menerima informasi dan memperbarui statusnya (contoh: menambahkan data ke database atau menghitung rumus)

- View memeriksa status dari model dan menanggapinya (misalnya menampilkan pesan data telah tersimpan atau hasil perhitungan)
- View lalu menanti interaksi lain dari pengguna

Sumber: (https://www.htmlgoodies.com/beyond/php/article.php/3912211/Principles-Of-MVC-for-PHP-Developers.htm)

MVC Dalam Kehidupan

Agar lebih jelas tentang arsitektur ini, berikut analogi pola MVC yang ada dalam kehidupan sehari-hari.

- Kita pergi ke sebuah restoran untuk makan. Kita tentu tidak bisa langsung ke dapur restoran untuk mengambil masakan, tetapi memanggil pelayan restoran.
- Pelayan restoran lalu menghampiri kita dan kita memesan makanan. Dalam hal ini, pelayan restoran tidak tahu siapa kita dan apa yang kita inginkan. Pelayan hanya mencatat pesanan kita.
- Pelayan kemudian menuju ke dapur restoran. Di dapur pelayan tidak menyiapkan sendiri masakan yang kita pesan. Tetapi menyerahkan catatan pesanan dan nomor meja kita ke koki restoran yang akan menyiapkan pesanan kita.
- Koki restoran mengambil bahan baku dan bumbu dari mesin pendingin untuk membuat pesanan kita.

- Setelah pesanan kita jadi, koki lalu menyerahkannya kepada pelayan restoran yang lalu membawa pesanan tersebut kepada kita sesuai pesanan

Dalam cerita tersebut:

- *Kita = View*
- *Pelayan restoran = Controller*
- *Koki restoran = Model*
- *Mesin pendingin = Data*

Kelebihan dan Kekurangan MVC

Kelebihan MVC:

- Aplikasi menjadi mudah dipelihara dan dikembangkan
- Pengembangan berbagai jenis komponen aplikasi dapat dilakukan secara paralel. Artinya pengembangan suatu komponen tidak tergantung komponen lain
- Mengurangi kompleksitas aplikasi karena pemisahan menjadi tiga unit Model, View, dan Controller
- Bekerja dengan baik untuk aplikasi berjenis web
- Aplikasi ramah SEO
- Semua kelas dan obyek tidak saling tergantung sehingga dapat diuji secara terpisah

Kekurangan MVC:

- Alur navigasi didalam aplikasi terkadang dapat menjadi kompleks karena menggunakan lapisan-lapisan Model, View, Controller, yang menuntut pengguna MVC untuk beradaptasi dengan kriteria dekomposisi MVC ini
- Meningkatkan kompleksitas dan ketidakefisienan data
- Memerlukan multi pemrogram untuk melakukan pemrograman paralel
- Pengetahuan multi teknologi diperlukan
- Perlu memelihara banyak kode di dalam controller

(Sumber: https://www.guru99.com/mvc-tutorial.html)

Framework Pengguna MVC

Framework populer yang menggunakan pola MVC ini:

- Ruby on Rails
- CodeIgniter
- Laravel
- Django
- CakePHP
- Yii
- Zend Framework
- Symphony

MVC Dalam PHP

Berikut ini contoh sederhana aplikasi PHP yang menerapkan pola MVC untuk memberikan gambaran awal arsitektur aplikasi yang menggunakan MVC. Tuliskan kode-kode dibawah ini dalam sebuah file PHP. Beri nama dengan **mvc.php**:

```
<?php
class Model
{
    public $string;

    public function __construct(){
        $this->string = "MVC + PHP = Luar biasa, Klik disini!";
    }
}

class View
{
    private $model;
    private $controller;

    public function __construct($controller,$model) {
        $this->controller = $controller;
        $this->model = $model;
    }

    public function output(){
```

```
        return ‘<p><a href=”mvc.php?action=clicked”’ . $this->model->string .
"</a></p>";
    }
}


class Controller
{
    private $model;

    public function __construct($model) {
        $this->model = $model;
    }
    public function clicked() {
            $this->model->string = “Updated Data, terima kasih pada MVC dan
    PHP!”;
        }
}


$model = new Model();
$controller = new Controller($model);
$view = new View($controller, $model);

if (isset($_GET['action']) && !empty($_GET['action'])) {
    $controller->{$_GET['action']}();
}

echo $view->output();
```

Jalankan file **mvc.php** dan jika kita klik pada sebuah link yang ditampilkan pada browser maka teks akan berubah