

MVC

MODEL - VIEW – CONTROLLER

Pola arsitektur pada perancangan perangkat lunak berorientasi dengan object.

Memisahkan antara tampilan data dan proses.

Ini contoh dari code **program Prosedural**

```
<?php
$conn = mysqli_connect("localhost", "root", "") or die("Koneksi Gagal");
mysqli_select_db($conn, "043040023") or die("database salah");
?>
<!DOCTYPE html>
<html>
<head>
    <title>Daftar Mahasiswa</title>
</head>
<body>
<h2>Daftar Mahasiswa</h2>

<?php $result = mysqli_query($conn, "SELECT * FROM mahasiswa"); ?>

<?php while( $row = mysqli_fetch_assoc($result) ) { ?>
<ul>
    <li>"></li>
    <li><?php echo $row["nama"]; ?></li>
    <li><?php echo $row["email"]; ?></li>
    <li><?php echo $row["jurusan"]; ?></li>
    <li><?php echo $row["universitas"]; ?></li>
</ul>
<?php } ?>
```

Koneksi DB

Presentasi

SQL Query

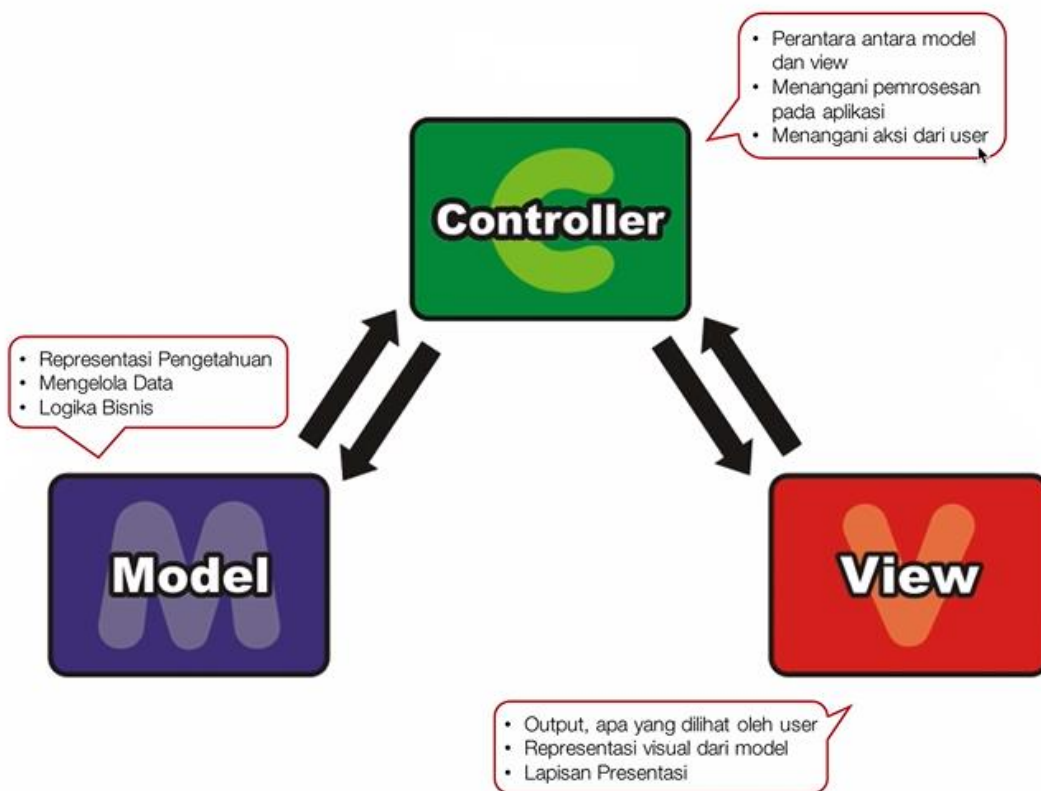
Presentasi

Kenapa MVC

- Organisasi dan Struktur Kode
- Pemisahan logic dan tampilan
- Memudahkan maintenance untuk kodenya
- Digunakan pada Framework

Framework MVC

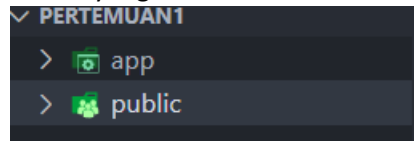
Bahasa	Framework
PHP	CodeIgniter, Laravel, Yii, dll
Java	Spring MVC, JSF, Struts, dll
Python	Django, CherryPy, dll
Ruby	Ruby on Rails, Sinatra ,dll
Javascript	AngularJS, React, Backbone.js, dll



Persiapan untuk membuat mvc dengan membuat folder :

1. App – Digunakan untuk menyimpan model ,dan controllernya.

Public – Digunakan untuk menyimpan , css , js, index , photo beserta tampilanya,dan alamat utama yang akan diakses dari user.



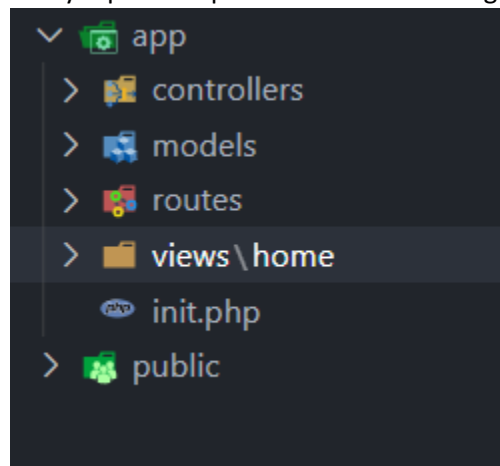
2. Didalm Folder app

Controllers - Digunakan untuk menyimpan file controllers.

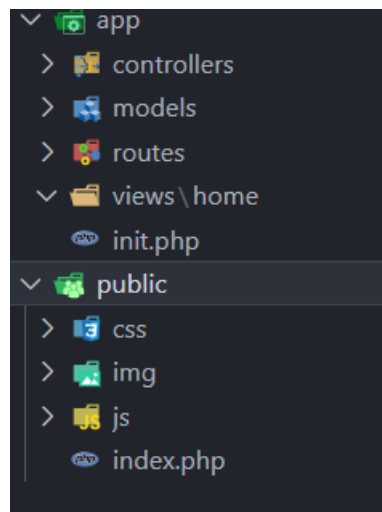
Models - Digunakan untuk menyimpan file model .

Routes – Digunakan untuk menyimpan inti mvcnya routing yang akan didirect .

Views – Digunakan untuk antar muka yang disajikan dari folder controllers ini , untuk menyimpan kumpulan views sesuai dengan controller yang memanggilnya nanti .



3. Kita membuat file index.html didalam file public

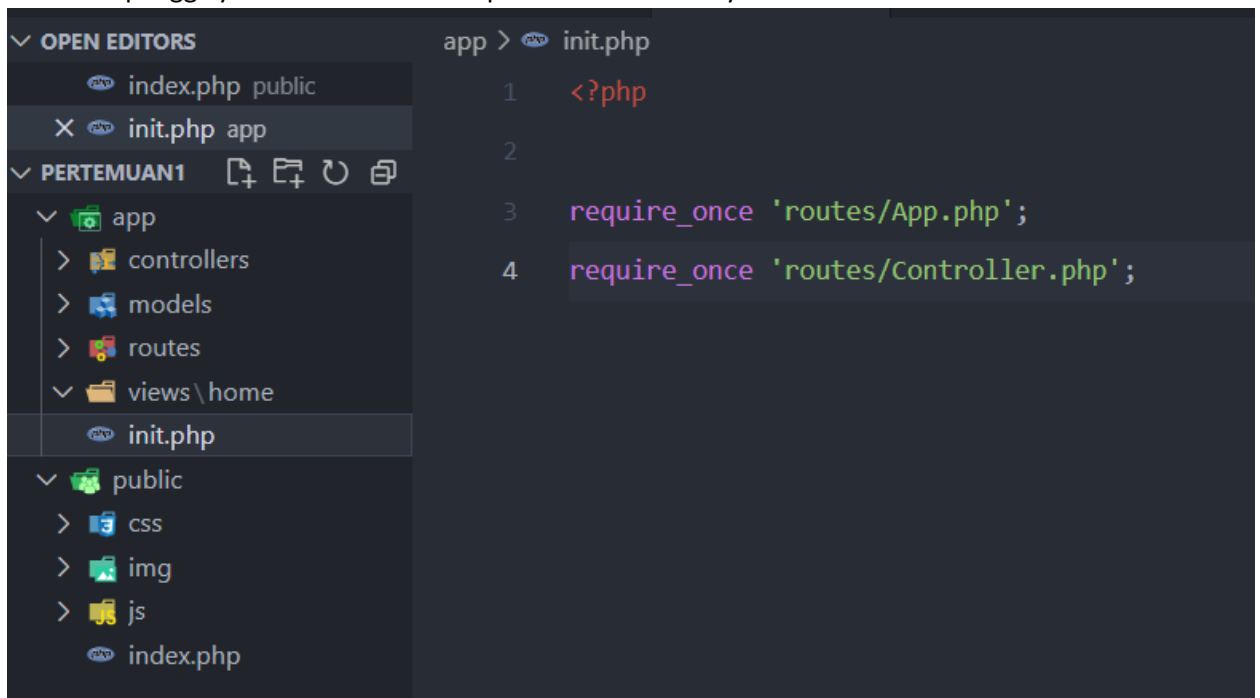


- Dan isi dari index.php ini kita akan panggil file yang ada di app dengan file init.php , dengan require once .

- File init.php ini nanti dia yang akan memanggil seluruh mvcnya yang kita butuhin didalam folder app , teknik ini disebut dengan bootstrapping.

```
public > index.php > ...  
  
1  <?php  
  
2  
  
3  require_once '../app/init.php';
```

- Didalam file init ini kita akan panggil class yang akan digunakan . 2 class ini yang dipanggil yaitu file utama untuk pembentukan mvcnya.



The screenshot shows an IDE with two panels. The left panel displays the 'OPEN EDITORS' and 'PERTEMUAN1' views. The 'OPEN EDITORS' view shows 'index.php public' and 'init.php app'. The 'PERTEMUAN1' view shows a file tree with 'app' (containing 'controllers', 'models', 'routes', and 'views\home') and 'public' (containing 'css', 'img', 'js', and 'index.php'). The 'init.php' file is selected. The right panel shows the content of 'init.php' in the 'app' directory, which includes the PHP opening tag and two 'require_once' statements for 'routes/App.php' and 'routes/Controller.php'.

```
app > init.php  
  
1  <?php  
  
2  
  
3  require_once 'routes/App.php';  
  
4  require_once 'routes/Controller.php';
```

- Didalam App.php dan Controller.php ini yaitu terdapat class ,

```

app > routes > App.php > PHP Intelephense > App
1  <?php
2
3  class App{
4      public function __construct()
5      {
6          echo "OK!";
7      }
8  }

```

- Class Controller ini yaitu controller utama yang nanti akan diextends pada folder Controller.

```

app > routes > Controller.php > PHP Intelephense > Controller
1  <?php
2
3  class Controller{
4
5  }

```

- Kita coba buat function construct agar langsung dapat dieksekusi dengan hanya memanggil class aja .

```

app > routes > App.php > PHP Intelephense > App
1  <?php
2
3  class App{
4      public function __construct()
5      {
6          echo "OK!";
7      }
8  }

```

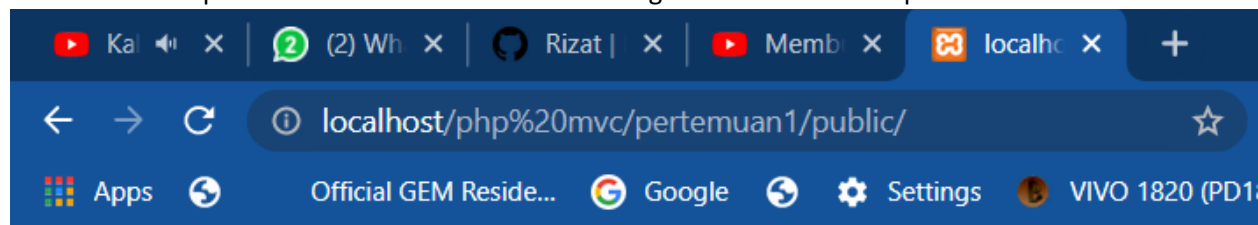
- Kita coba jalankan mvcnya dengan memanggil class App

```

public > index.php > ...
1  <?php
2
3  require_once '../app/init.php';
4
5  $app = new App;

```

- Lalu coba lihat apakah sudah berhasil atau belum dengan membuka folder public.



OK!

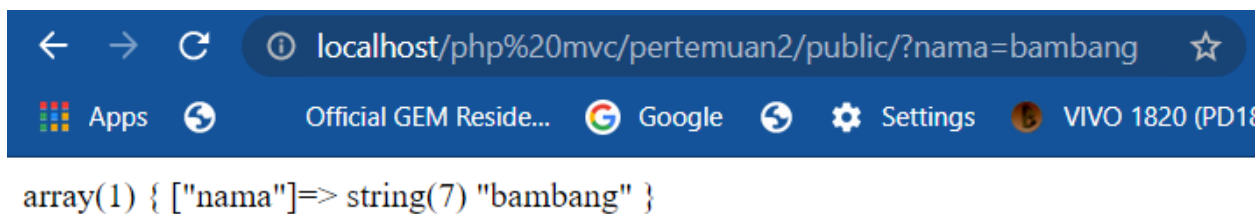
Routing

Yaitu membuat url agar redirect sesuai yang kita inginkan ,

Kita buka file App.php di folder routes

`$_GET` digunakan untuk mengambil apa saja yang ada pada url.

```
3  class App{
4      public function __construct()
5      {
6          var_dump($_GET);
7      }
8  }
```



array(1) { [\"nama\"]=> string(7) \"bambang\" }

Membuat function untuk mengambil url dan dipecah dan dirapihkan

`.htaccess` file untuk memodifikasi server apache kita, dan kita bisa lakukan perFolder

Htaccess Indexes .Kita bikin file `.htaccess`, disini kita membuat options di folder app selama di folder app tidak ada file index , maupun index.html atau index.php tidak akan ditampilkan,dan blocks aksesnya .

```
app > .htaccess
1  Options -indexes
```

Kita membuat file `.htaccess` pada folder public

Htaccess multiviews, digunakan untuk menghindari kesalahan ketika kita memanggil file/folder didalam folder tersebut.

```
public > .htaccess
1  Options -Multiviews
```

RewriteEngine On digunakan untuk menulis ulang link.

```
Options -Multiviews

//digunakan untuk membuat menulis ulang url
RewriteEngine On

// kondisi request fileName jika url yang ditulis nama folder diabaikan
RewriteCond %{REQUEST_FILENAME} !-d

//untuk menghindari nama folder/nama fil
RewriteCond %{REQUEST_FILENAME} !-f

//aturan untuk menulis ulang urlnya
// ^ membaca apapun yang ditulis di url setelah menulis folder public
// . ambil apapun
// * ambil caracter 1 , 1 kecuali enter
// $ sampai selesai ketika pencet enter
// arahkan ke file index yang mengirimkan url
// $1 placeholder apapun yang ditulis di url dimasukan ke $1
// [L] aturan jika ada rule lain yang terpenuhi, jangan jalankan rule
lain setelah ini .
RewriteRule ^(.*)$ index.php?url=$1 [L]
```

Rtrim digunakan untuk menghapus apapun yang ada , disini kita ingin menghapus '/'


```
public function parseUrl(){  
    // jika ada data url dikirimkan  
    if( isset($_GET['url']) ){  
        $url = rtrim($_GET['url'], '/') ;  
        return $url;  
    }  
}
```

Filter_var digunakan untuk membuat filter dari urlnya , disini kita bersihkan dari character character aneh dari url kita .

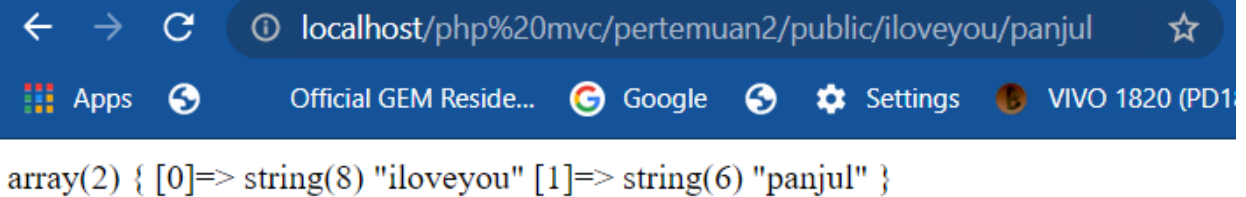
```
public function parseUrl(){  
    // jika ada data url dikirimkan  
    if( isset($_GET['url']) ){  
        $url = rtrim($_GET['url'], '/') ;  
        $url = filter_var($url, FILTER_SANITIZE_URL);  
        return $url;  
    }  
}
```

Explode digunakan untuk memecah string yang ada , disini kita memecahnya melalui '/' dari data \$url menjadi array.

```

public function parseUrl(){
    // jika ada data url dikirimkan
    if( isset($_GET['url']) ){
        $url = rtrim($_GET['url'], '/') ;
        $url = filter_var($url, FILTER_SANITIZE_URL);
        $url = explode('/', $url);
        return $url;
    }
}

```



← → ↻ ⓘ localhost/php%20mvc/pertemuan2/public/iloveyou/panjul ☆

Apps ↻ Official GEM Reside... Google ↻ ⚙ Settings 📱 VIVO 1820 (PD1

array(2) { [0]=> string(8) "iloveyou" [1]=> string(6) "panjul" }

Menggabungkan String dengan implode()

jika **explode** adalah pemecah string, maka **implode** adalah penggabung kembali string yang telah di pecahkan oleh explode.

Controller

Kita edit lagi file App.php didalam folder routes

1. Kita membuat default datanya dahulu

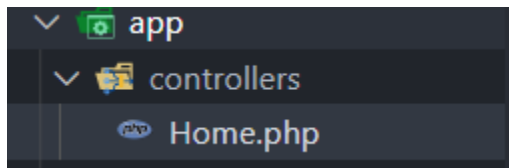
```

class App{
    protected    $controller = 'Home',
                 $method = 'index',
                 $params = [];

    public function __construct()
    {
        $url = $this->parseUrl();
        var_dump($url);
    }
}

```

2. Kita membuat controller dengan nama yang sama dengan controller yang kita buat.



3. Dan buat class sama seperti nama controllersnya , dan nama methodnya index .

```

class Home{
    public function index()
    {
        echo "home/index";
    }
}

```

4. mengecek apakah ada file yang ada didalam folder controllers yang dikirimkan di url , di folder routes - App.php .

```
// controller
// mengecek apakah ada file yang ada didalam folder controllers yang dikirimkan di url
if( file_exists('../app/controllers/'. $url[0] . '.php') ){
    $this->controller = $url[0];
    // kita hilangkan controllernya dari element array
    unset($url[0]);
}

require_once '../app/controllers/'.$this->controller.'.php';
// diinstansiasi
$this->controller = new $this->controller;
```

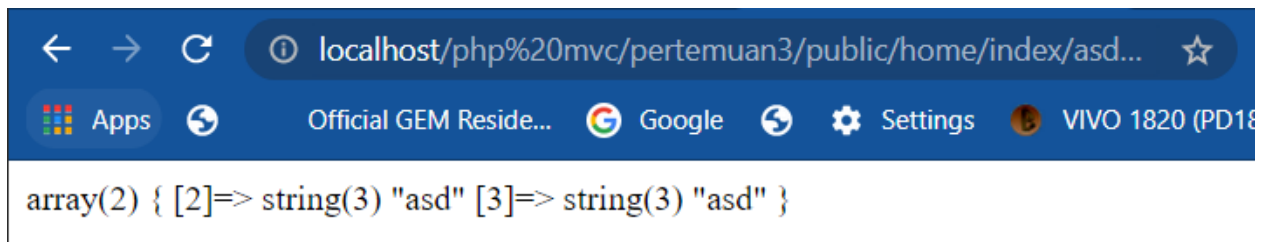
5. Lalu kita buat untuk methodnya

```
// method
if( isset($url[1]) ){
    if( method_exists($this->controller, $url[1]) ){
        $this->method = $url[1];
        unset($url[1]);
    }
}
```

Lalu kita membuat paramsnya .

```
// params
if( !empty($url) ){
    var_dump($url);
}
```

6. Jika dijalankan kita kirimkan parameternya jika ada filenya dan ada methodnya maka url index 0, dan 1 akan hilang akan tersisa index 2,3 itu dijadikan paramsnya .



A screenshot of a web browser window. The address bar shows the URL `localhost/php%20mvc/pertemuan3/public/home/index/asd...`. The browser's toolbar includes icons for Apps, Official GEM Reside..., Google, Settings, and a VIVO 1820 (PD18) device. The main content area displays a PHP array output: `array(2) { [2]=> string(3) "asd" [3]=> string(3) "asd" }`.

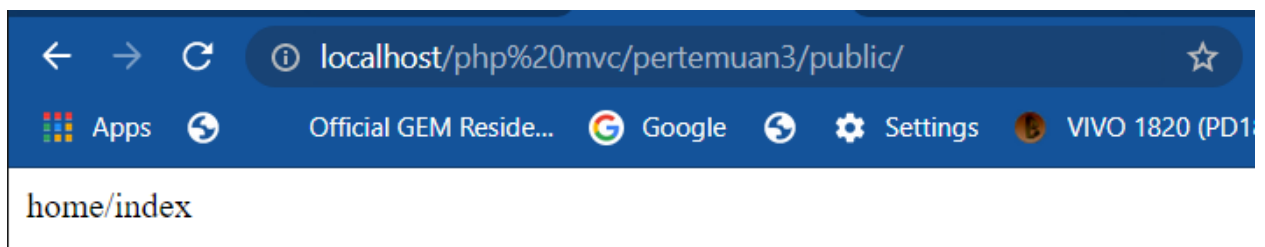
7. mengambil data array dari urlnya
jalankan contoller dan method, serta kirimkan params



```
// params
if( !empty($url) ){
    // mengambil data array dari urlnya
    $this->params = array_values($url);
    var_dump($url);
}

// jalankan contoller dan method, serta kirimkan params
// parameter pertama menajalankan function , parameter ke 2
// untuk mengirimkan params
call_user_func_array([$this->controller,$this->method],
    $this->params);
}
```

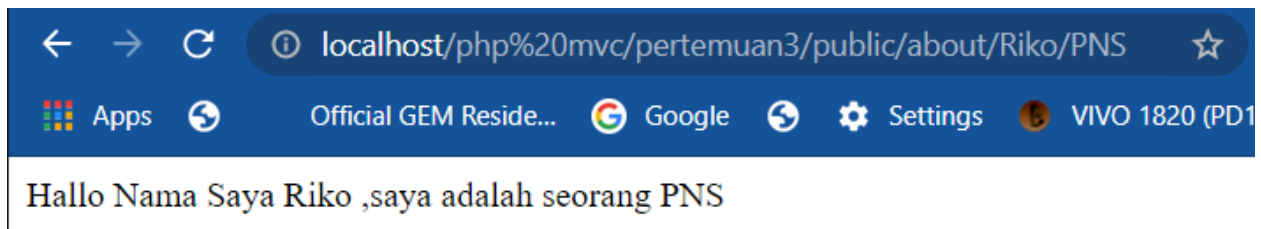
8. Maka jika kita hanya menuju public langsung menjalankan controllers defaultnya yaitu file home dan method index.



A screenshot of a web browser window. The address bar shows the URL `localhost/php%20mvc/pertemuan3/public/`. The browser's toolbar is the same as in the previous screenshot. The main content area displays the text `home/index`.

9. Jika kita memiliki file controllernya harus mempunyai method default index .Dan untuk menangkap data yang dikirimkan .

```
app > controllers > About.php > PHP Intelephense > About
1  <?php
2
3  class About {
4      public function index($nama = 'Rizat Sakmir',$pekerjaan = 'WEB
        Developer'){
5          echo "Hallo Nama Saya $nama ,saya adalah seorang $pekerjaan";
6      }
7      public function page()
8      {
9          echo 'About/page';
10     }
11 }
```



VIEW

Kita akan membuat function view untuk mengarahkan ke halaman web yang sesuai dengan routenya .

1. Buka file Controller.php didalam folder App-Route .

```
pertemuan4 > app > routes > Controller.php > PHP Intelephense > Controller > view
1  <?php
2
3  class Controller{
4      // membuat function view dan menerima parameter yang dikirimkan yaitu berupa folder dan filenya
5      // dan kita siapkan parameter ke 2 untuk menerima parameter data dan dikasih array kosong.
6      public function view($view,$data = [])
7      {
8          // kita langsung aja panggil file yang ingin kita tampilkan
9          require_once '../app/views/'.$view.'.php';
10     }
11 }
```

2. Buka file home.php karena kita ingin menampilkan view berupa halaman web bukan lagi echo .

```
pertemuan4 > app > controllers > Home.php >
1  <?php
2
3  class Home{
4      public function index()
5      {
6          echo "home/index";
7      }
8  }
```

3. Meridirect ke halaman web yang ada di folder views

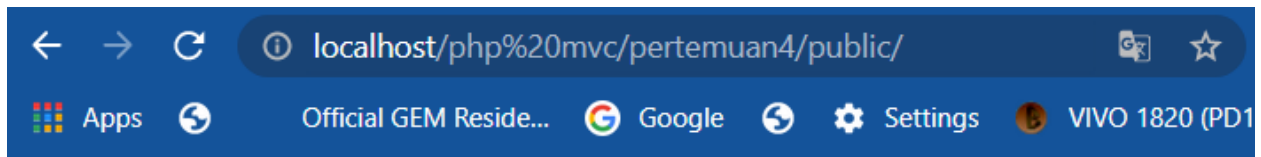
pertemuan4 > app > controllers > Home.php > PHP Intelephense > Home

```
1  <?php
2
3  // kita extends ke file controller agar dapat menggunakan function Controller
4  class Home extends Controller{
5      public function index()
6      {
7          // kita jalankan function view nya dan mengirimkan parameter file yang dituju .
8          $this->view('index');
9      }
10 }
```

4. Hasilnya kita dapat tampilkan isi dari file index.php didalam folder views .

pertemuan4 > app > views > index.php > html

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Belajar PHP MVC</title>
8  </head>
9  <body>
10     <h1>Hello Wolrd!</h1>
11 </body>
12 </html>
```

Hello Wolrd!

5. Kita mencoba sekalian mengirimkan data kita buka file about.php yang didalam folder controllers .

```
pertemuan4 > app > controllers > About.php > PHP Intelephense > About
1  <?php
2
3  class About extends Controller{
4      public function index($nama = 'Rizat Sakmir',$pekerjaan = 'WEB Developer',$umur = 21)
5      {
6          // dengan membuat array assoative
7          $data['nama'] = $nama;
8          $data['pekerjaan'] = $pekerjaan;
9          $data['umur'] = $umur;
10         // karena di function view kita sudah menyiapkan parameter untuk menangkap data
11         $this->view('about/index',$data);
12     }
13     public function page()
14     {
15         $this->view('about/page');
16     }
17 }
```

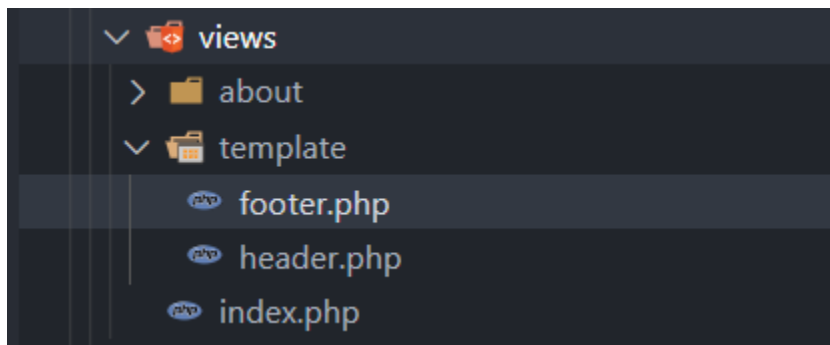
6. Kita akan gunakan datanya pada file about/index.php

```

pertemuan4 > app > views > about > index.php > html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Halaman About</title>
8  </head>
9  <body>
10     <h1>About Me</h1>
11     <p>Hello Saya <?=$data['nama']?> Saya adalah seorang <?=$data['pekerjaan']?> dan umur saya <?=$data['umur']?>.</p>
12 </body>
13 </html>

```

7. Kita akan membuat template Header dan footernya .
Membuat folder template didalam folder views.



8. Diisikan header untuk file header.php . sekalian kita tangkap datanya untuk membuat Titlenya.

```

pertemuan4 > app > views > template > header.php > html > body
1  <!doctype html>
2  <html lang="en">
3  <head>
4      <!-- Required meta tags -->
5      <meta charset="utf-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1">
7
8      <!-- Bootstrap CSS -->
9      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
10
11      <title><?=$data['title']?></title>
12 </head>
13 <body>

```

9. Dan diisikan Footer untuk file footer.php

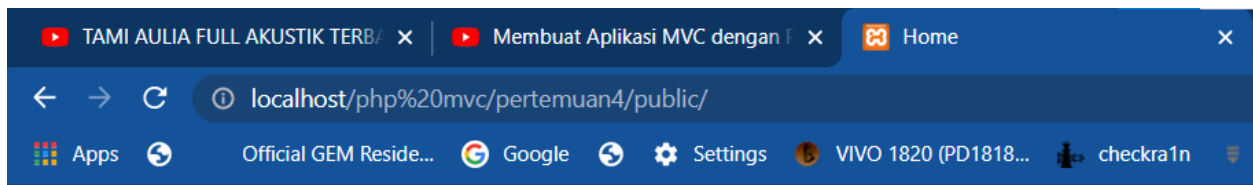
```
pertemuan4 > app > views > template > footer.php > ...
1
2 <!-- Bootstrap JS -->
3 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integrity="
4 </body>
5 </html>
```

10. Lalu kita akan gunakan template di tiap tiap halamannya seperti ini .

```
pertemuan4 > app > controllers > Home.php > PHP Intelephense > Home > index
1 <?php
2
3 // kita extends ke file controller agar dapat menggunakan function Controller
4 class Home extends Controller{
5     public function index()
6     {
7         // kita jalankan function view nya dan mengirimkan parameter file yang dituju .
8         // kirimkan data untuk titlenya
9         $data['title'] = 'Home';
10        $this->view('template/header',$data);
11        $this->view('index');
12        $this->view('template/footer');
13    }
14 }
```

11. Didalam file index.php didalam folder views hanya berisikan body saja.

```
pertemuan4 > app > views > index.php > ...
1
2 <h1>Hello, world!</h1>
3
```



Hello, world!

Dan lakukan berulang pada tiap tiap controllernya.

Assets

Menggunakan file assets yang ada didalam folder app, seperti css,js dan image.

Kita menggunakan absolute url.

Contoh dalam penggunaan css bootstrap .

```
<!doctype html>
<html lang="en">
<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <!-- Bootstrap CSS -->
  <link href="http://localhost/php mvc/pertemuan5/public/css/bootstrap.css" rel="stylesheet">

  <title><?=$data['title']?></title>
</head>
<body>
```

Kita akan membuat url agar kalau kita merubah foldernya hanya merubah 1 file dan 1 baris saja seperti .Env

1. Membuat file Env.php didalam folder routes
Define untuk membuat constanta.

```
pertemuan5 > app > routes > Env.php > ...
```

```
1  <?php
2
3  define('URL','http://localhost/php_mvc/pertemuan5/public');
```

2. Buka file init.php kita require file Env.php ini .

```
pertemuan5 > app > init.php
```

```
1  <?php
2
3  require_once 'routes/App.php';
4  require_once 'routes/Controller.php';
5  require_once 'routes/Env.php';
```

3. Kita tinggal gunakan untuk menggunakan asset di file public .

```
<!-- Bootstrap CSS -->
<link href="<?=URL;?>/css/bootstrap.css" rel="stylesheet">
```

MODEL

Kita memanggil data nama dari model .

1. Kita buka file controllers – home.php dan tambahkan codingan ini .

```
// memanggil nama dari model yang nama filenya User_model dan jalankan function getUser
$data['nama'] = $this->model('User_model')->getUser();
```

2. Kita membuat file User_model.php didalam folder models.

```
pertemuan5 > app > models > User_model.php > PHP I
1  <?php
2
3  class User_model
4  {
5      private $nama = 'Rizat Sakmir';
6
7      public function getUser()
8      {
9          return $this->nama;
10     }
11 }
```

3. Lalu kita membuat function model didalam file Controller.php didalam folder routes

```
public function model($model)
{
    require_once '../app/models/'.$model.'.php';
    // sekalian kita instansiasi/ jalankan function get user nya.
    return new $model;
}
```

4. Jangan lupa kirimkan parameter \$data ke dalam view homenya .

```
// memanggil nama dari model yang nama filenya User_model dan jalankan function getUser
$data['nama'] = $this->model('User_model')->getUser();

$this->view('template/header',$data);
$this->view('index',$data);
```

Kita akan mengambil data model dari database menggunakan PDO.

1. Kita buka buat tampilan untuk data mahasiswa terlebih dahulu .
Lalu kita membuat variable data['mahasiswa'] didalam file controllers - Mahasiswa.php.

```
pertemuan5 > app > controllers >  Mahasiswa.php > PHP Intelephense >  Mahasiswa >  index

1  <?php
2
3  class Mahasiswa extends Controller
4  {
5      public function index()
6      {
7          $data['title'] = 'Data Mahasiswa';
8          $data['active'] = 'active mahasiswa';
9          $data['mahasiswa'] = $this->model('Mahasiswa_model')->getAllMahasiswa();
10         $this->view('template/header',$data);
11         $this->view('mahasiswa/index',$data);
12         $this->view('template/footer');
13     }
14 }
```

2. Kita buat file model mahasiswa_model.php didalam folder model.

pertemuan5 > app > models > Mahasiswa_model.php > PHP Intelephense > Mahasiswa_model > __construct

```
1  <?php
2
3  class Mahasiswa_model
4  {
5      private $dbh, //database handler
6          $stmt;
7
8      public function __construct()
9      {
10         // data source name
11         $dsn = "mysql:host=".host.";dbname=".db_name;
12
13         // coba connect kan ke database
14         try{
15             $this->dbh = new PDO($dsn,username,password);
16         }
17         // jika gagal connect masukan ke variable $e
18         catch(PDOException $e){
19             // hentikan program dan berikan pesan eror
20             die($e->getMessage());
21         }
22     }
```

3. Membuat function getAllMahasiswa() juga disini .

```
public function getAllMahasiswa()
{
    // kita query dengan menggunakan function prepare
    $this->stmt = $this->dbh->prepare('SELECT * FROM tb_mahasiswa');
    // kita eksekusi querynya
    $this->stmt->execute();
    // kita return dengan mengeluarkan semua data dari database dan menghasilkan array assosiative
    return $this->stmt->fetchAll(PDO::FETCH_ASSOC);
}
```


4. Untuk menampilkan datanya di tampilan mahasiswa.

```
<h1>Data Mahasiswa</h1>

<?php foreach($data['mahasiswa'] as $mahasiswa): ?>

    <ul>

        <li><?=$mahasiswa['nim']?></li>

        <li><?=$mahasiswa['nama']?></li>

        <li><?=$mahasiswa['jurusan']?></li>

        <li><?=$mahasiswa['photo']?></li>

    </ul>

<?php endforeach; ?>
```

DATABASE WRAPPER

Supaya kita sekali membuat function untuk databasenya agar bisa digunakan banyak query

1. Pertama kita buat file Database.php didalam folder route.

```
pertemuan5 > app > routes > Database.php > PHP Intelephense > Database > single

1  <?php
2
3  class Database
4  {
5      private $db_host = db_host,
6              $db_username = db_username,
7              $db_password = db_password,
8              $db_name = db_name,
9              $dbh, //database handler
10             $stmt;
```

2. Lalu kita buat function construct

```
public function __construct()
{
    // data source name
    $dsn = "mysql:host=".$this->db_host.";dbname=".$this->db_name;

    $option = [
        PDO::ATTR_PERSISTENT => true, //menjaga agar tetap terkoneksi
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION //untuk error modenya tampilkan exception
    ];
    // coba connect kan ke database
    try{
        $this->dbh = new PDO($dsn,$this->db_username,$this->db_password,$option);
    }
    // jika gagal connect masukan ke variable $e
    catch(PDOException $e){
        // hentikan program dan berikan pesan eror
        die($e->getMessage());
    }
}
```

3. Lalu kita buat function query

```
// digunakan untuk mempersiapkan query jadi dia bisa select , insert ,update , delete.
public function query($query)
{
    $this->stmt = $this->dbh->prepare($query);
}
```

4. Lalu kita buat function binding.

pertemuan5 > app > routes > Database.php > PHP Intelephense > Database > bind

```
38      // digunakan untuk binding seperti ada wherenya ada parameternya apa .
39      // gunanya untuk menghindari mysql injection
40      public function bind($param, $value , $type = null)
41      {
42          if( is_null($type) ){
43              // supaya switchnya jalan aja
44              switch (true) {
45                  // jika valuenya integer maka type nya dijadikan integer.
46                  case is_int($value) :
47                      $type = PDO::PARAM_INT;
48                      break;
49                  // jika valuenya boolean maka type nya dijadikan boolean.
50                  case is_bool($value) :
51                      $type = PDO::PARAM_BOOL;
52                      break;
53                  case is_null($value) :
54                      $type = PDO::PARAM_NULL;
55                      break;
56                  default : $type = PDO::PARAM_STR;
57              }
58          }
59      }
60      $this->stmt->bindValue($param, $value, $type);
61  }
```

5. Lalu kita buat function execute
6. Buat function resultSet
7. Buat function single.

```

public function execute()
{
    return $this->stmt->execute();
}

// digunakan untuk mengambil banyak data
public function resultSet()
{
    $this->execute();
    return $this->stmt->fetchAll(PDO::FETCH_ASSOC);
}

// digunakan untuk hanya mengambil 1 data aja.
public function single()
{
    $this->execute();
    return $this->stmt->fetch(PDO::FETCH_ASSOC);
}
}

```

8. Lalu kita require di file init.php

```

pertemuan5 > app > init.php
1  <?php
2
3  require_once 'routes/App.php';
4  require_once 'routes/Controller.php';
5  require_once 'routes/Database.php';
6  require_once 'config/Config.php';

```

9. Lalu kita ke folder Controllers – Mahasiswa.php kita rubah menggunakan file di folder routes – Database.php.

```

pertemuan5 > app > models > Mahasiswa_model.php > PHP Intelephense > Mahasiswa_model > getAllMahasiswa
1  <?php
2
3  class Mahasiswa_model
4  {
5      private $table = 'tb_mahasiswa',
6              $tb;
7
8      public function __construct()
9      {
10         $this->db = new Database;
11     }
12     public function getAllMahasiswa()
13     {
14         $this->db->query("SELECT * FROM $this->table");
15         return $this->db->resultSet();
16     }

```

Kita akan membuat detail data masuk file mahasiswa – index.php.

1. Kita membuat tombol detail dan href menuju ke file mahasiswa yang di dalam folder controllers dan memiliki method detail_Mahasiswa kirimkan data id nya.

```

<h1>Data Mahasiswa</h1>
<ul class="list-group">
    <?php foreach($data['mahasiswa'] as $mahasiswa): ?>
        <li class="list-group-item d-flex justify-content-between align-items-center">
            <?=$mahasiswa['nama']?>
            <a href="<?=URL?>/mahasiswa/detail_Mahasiswa/<?=$mahasiswa['id']?>" class="badge bg-primary rounded-pil
        </li>
    <?php endforeach; ?>
</ul>

```

2. Kita ke file Mahasiswa.php didalam folder controllers.
Kita tambahkan method detail_Mahasiswa.

```

public function detail_Mahasiswa($id)
{
    $data['title'] = 'Data Mahasiswa';
    $data['active'] = 'active mahasiswa';
    $data['mhs'] = $this->model('Mahasiswa_model')->getMahasiswaById($id);
    $this->view('template/header',$data);
    $this->view('mahasiswa/detail_Mahasiswa',$data);
    $this->view('template/footer');
}

```

3. Lalu kita kirimkan datanya di file model – Mahasiswa_model.php dan membuat function getMahasiswaByld yang dikirimkan datanya \$id.

```

public function getMahasiswaById($id)
{
    $this->db->query("SELECT * FROM $this->table WHERE id=:id");
    // kalau memiliki parameter kita binding dulu
    $this->db->bind('id', $id);
    // hanya mengambil 1 data menggunakan function single yang didalam file Database.php
    return $this->db->single();
}

```

4. Lalu kita buat file detail_Mahasiswa didalam folder view – mahasiswa .

```

1 <div class="container mt-5">
2     <div class="row">
3         <div class="col-md-6">
4             <h1>Data Mahasiswa <?=$data['mhs']['nama']?></h1>
5             <div class="card" style="width: 18rem;">
6                 <div class="card-body">
7                     <h5 class="card-title"><?=$data['mhs']['nama']?></h5>
8                     <p class="card-text"><?=$data['mhs']['nim']?></p>
9                     <p class="card-text"><?=$data['mhs']['jurusan']?></p>
10                    <p class="card-text"><?=$data['mhs']['photo']?></p>
11                    <a href="<?=$URL?>/mahasiswa" class="card-link">kembali</a>
12                </div>
13            </div>
14        </div>
15    </div>
16 </div>

```

INSERT DATA MENGGUNAKAN MODAL

1. Pertama kita membuat tombol modal di file index.php didalam folder mahasiswa.

```

<!-- Button trigger modal -->
<button type="button" class="btn btn-primary mt-4 mb-4" data-bs-toggle="modal" data-bs-target="#exampleModal">
    <svg xmlns="http://www.w3.org/2000/svg" style="color:white" width="25" height="25" fill="currentColor" class="
    <path d="M16 8A8 8 0 1 0 8a8 8 0 0 1 16 0zM8.5 4.5a.5.5 0 0 0-1 0v3h-3a.5.5 0 0 0 1h3v3a.5.5 0 0 0 1 0
    </svg> Tambah Data Mahasiswa
</button>
<!-- End button -->

```

2. Lalu kita membuat modal formnya . masih di dalam file index.php folder mahasiswa.

```

<form action="<?=URL?>/mahasiswa/tambah_Mahasiswa" method="post">
  <div class="modal-body">
    <div class="mb-3">
      <label for="nama" class="form-label">Nama</label>
      <input type="text" name="nama" class="form-control" id="nama" required>
    </div>
    <div class="mb-3">
      <label for="nrp" class="form-label">NRP</label>
      <input type="number" name="nrp" class="form-control" id="nrp" required>
    </div>
    <div class="mb-3">
      <label for="email" class="form-label">Email</label>
      <input type="text" name="email" class="form-control" id="email" required>
    </div>
    <div class="mb-3">
      <label for="jurusan" class="form-label">Jurusan</label>
      <select class="form-select" id="jurusan" name="jurusan">
        <option value="Teknologi Informasi">Teknologi Informasi</option>
        <option value="Sistem Informasi">Sistem Informasi</option>
        <option value="Rekayasa Perangkat Lunak">Rekayasa Perangkat Lunak</option>
      </select>
    </div>
  </div>
  <div class="modal-footer">
    <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>
    <button type="submit" class="btn btn-primary">Tambah Data</button>
  </div>
</form>

```

3. Lalu kita membuat function didalam file mahasiswa di folder controllers dengan nama methodnya tambah_Mahasiswa .


```

public function tambah_Mahasiswa()
{
    // jika model Mahasiswa_model dijalankan tambahDataMahasiswa dengan mengirimkan data $_POST
    // berhasil dijalankan maka ada data yang tambah menjadi Lebih dari 0
    // redirect ke mahasiswa
    // dan exit dari function ini .
    if ( $this->model('Mahasiswa_model')->tambahDataMahasiswa($_POST) > 0 ){
        header('Location:' .URL. '/mahasiswa');
        exit;
    }
}

```

4. Lalu kita membuat function tambahDataMahasiswa di dalam model Mahasiswa_model.

```

public function tambahDataMahasiswa($data)
{
    $query = "INSERT INTO $this->table VALUES ('',:nama,:nrp,:email,:jurusan)";

    $this->db->query($query);
    $this->db->bind('nama', $data['nama']);
    $this->db->bind('nrp', $data['nrp']);
    $this->db->bind('email', $data['email']);
    $this->db->bind('jurusan', $data['jurusan']);

    $this->db->execute();
    // mengembalikan nilai jumlah baris yang bertambah,diedit,dihapus.
    return $this->db->rowCount();
}

```

5. Lalu jangan lupa kita membuat function rowCount di file Database.php di folder route.

```
public function rowCount()
{
    return $this->stmt->rowCount();
}
```

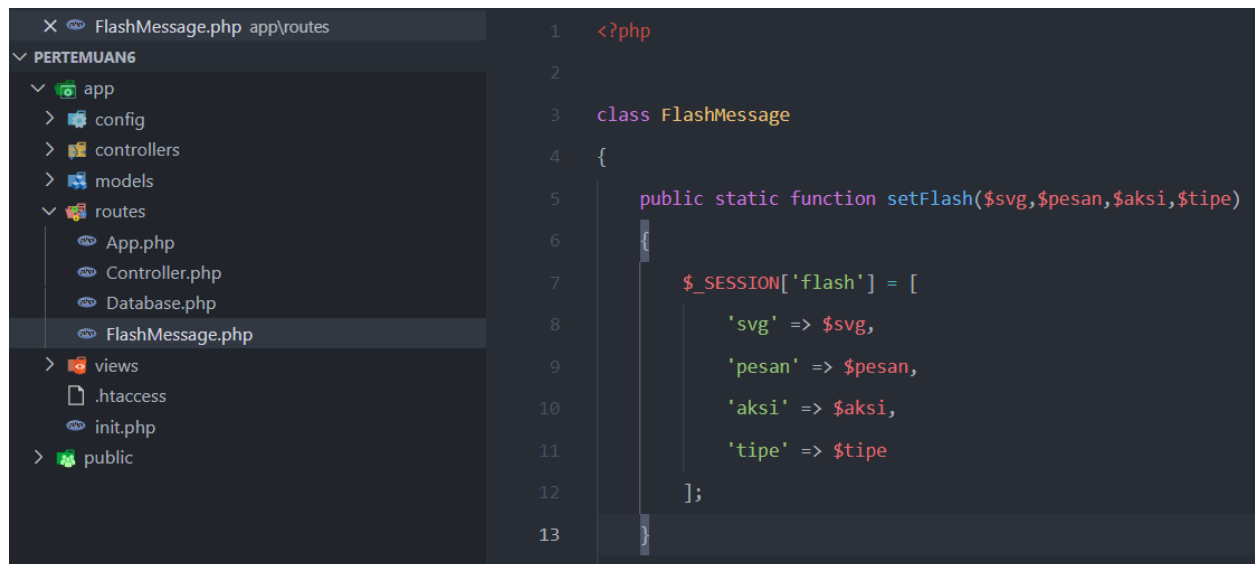
FLASH MESSAGE

Yaitu pesan kilat setelah kita melakukan aksi.

1. Pertama kita membuat file FlashMessage.php didalam folder routes dan membuat function static

Dan membuat class FlashMessage dan membuat function setFlash dengan parameter .

1. Untuk logo svg nya .
2. Untuk pesan yang dikirimkan , Berhasil/Gagal.
3. Aksi untuk di tambahkan/ di edit / dihapus.
4. Tipe warna alertnya .



```
1 <?php
2
3 class FlashMessage
4 {
5     public static function setFlash($svg,$pesan,$aksi,$tipe)
6     {
7         $_SESSION['flash'] = [
8             'svg' => $svg,
9             'pesan' => $pesan,
10            'aksi' => $aksi,
11            'tipe' => $tipe
12        ];
13    }
```

2. Membuat function statis untuk flashnya .
Kita cek apakah ada data session flashnya jika ada maka tempilkan alertnya.
Lalu kita unset dihapus session flashnya agar hanya bisa dipakai sekali .

```

public static function flash()
{
    if ( isset($_SESSION['flash']) ){
        echo '<div class="alert alert-'. $_SESSION['flash']['tipe'] .' alert-dismissible fade show" role="alert">
        <svg class="bi flex-shrink-0 me-2" width="24" height="24" role="img" ><use xlink:href="#'. $_SESSION['flash']
        ['svg'].'" /></svg>
        Data Mahasiswa <strong>' . $_SESSION['flash']['pesan'] . '</strong> ' . $_SESSION['flash']['aksi'] . '.
        <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
        </div>';

        unset($_SESSION['flash']);
    }
}

```

3. Lalu kita require di file init.php

```

app > init.php
1  <?php
2
3  require_once 'routes/App.php';
4  require_once 'routes/Controller.php';
5  require_once 'routes/Database.php';
6  require_once 'routes/FlashMessage.php';
7
8  require_once 'config/Config.php';

```

4. Lalu kita kirimkan datanya di file controller – Mahasiswa.php ketika berhasil menambahkan data mahasiswa .

```
app > controllers > Mahasiswa.php > PHP Intelephense > Mahasiswa > tambah_Mahasiswa

24
25     public function tambah_Mahasiswa()
26     {
27         if ( $this->model('Mahasiswa_model')->tambahDataMahasiswa($_POST) > 0 ){
28             FlashMessage::setFlash('check-circle-fill','Berhasil','Di tambahkan', 'success');
29             header('Location:' .URL. '/mahasiswa');
30             exit;
31         }else{
32             FlashMessage::setFlash('exclamation-triangle-fill','Gagal','Di tambahkan', 'danger');
33             header('Location:' .URL. '/mahasiswa');
34             exit;
35         }
36     }
37 }
```

5. Lalu kita file index.php didalam folder public untuk jalankan sessionnya .

```
public > index.php > ...

1  <?php
2  if (!session_id()) session_start();
3  require_once '../app/init.php';
4
5  $app = new App;
```

OPEN EDITORS: index.php public

PERTEMUAN6

- app
- public
 - css
 - img
 - js
 - .htaccess
 - index.php

6. Lalu kita panggil sessionnya di halaman index.php didalam folder mahasiswa

```
<div class="flash mt-4">

    <?php FlashMessage::flash(); ?>

</div>
```

DELETE DATA

1. Pertama kita bikin tombol untuk meridirect ke function delete_Mahasiswa .

```
<a href="<?=URL?>/mahasiswa/delete_Mahasiswa/<?=$mahasiswa['id']?>" class="badge bg-danger"
onclick="return confirm('Apakah anda yakin ingin menghapus data ini ?');">Delete</a>
```

2. Kita ke controller mahasiswa bikin function delete_Mahasiswa.

```
public function delete_Mahasiswa($id)
{
    if ( $this->model('Mahasiswa_model')->hapusDataMahasiswa($id) > 0 ){
        FlashMessage::setFlash('check-circle-fill','Berhasil','Di Hapus', 'success');
        header('Location:' .URL. '/mahasiswa');
        exit;
    }else{
        FlashMessage::setFlash('exclamation-triangle-fill','Gagal','Di Hapus', 'danger');
        header('Location:' .URL. '/mahasiswa');
        exit;
    }
}
```

3. Kita ke model Mahasiswa_model untuk membuat function hapusDataMahasiswa.

```

public function hapusDataMahasiswa($id)
{
    $query = "DELETE FROM $this->table WHERE id=:id";

    $this->db->query($query);
    $this->db->bind('id',$id);

    $this->db->execute();

    return $this->db->rowCount();
}

```

UPDATE DATA + AJAX

1. Membuat tombol untuk edit.

```

<button class="badge bg-success editDataMahasiswa" data-bs-toggle="modal"
data-bs-target="#exampleModal" data-id="<?=$mahasiswa['id']?>">Edit</button>

```

2. Membuat file script.js dan jangan lupa dimasukan di dalam file footernya scriptnya , dan masukan cdn JQuerynya .

```

app > views > template > footer.php > script
1      <!-- JQuery -->
2      <script src="https://code.jquery.com/jquery-3.6.0.js"></script>
3      <!-- Bootstrap JS -->
4      <script src="<?=URL?>/js/bootstrap.js"></script>
5      <!-- script js -->
6      <script src="<?=URL?>/js/script.js"></script>
7
8      </body>
9      </html>

```

3. Kita mengubah isi dari modalnya ketika kita klik tombol edit .

```

// jalankan function kalau filenya sudah siap
$(function(){

    // ketika kita klik tombol edit data mahasiswa
    $('.editDataMahasiswa').on('click',function(){
        $('.modal-title').html('Edit Data Mahasiswa');
        $('.modal-footer button[type=submit]').html('Edit Data');
        // mengubah href ke function edit_mahasiswa
        $('.modal-
content form').attr('action', 'http://localhost/php.mvc/pertemuan6/public/mahasis
wa/edit_Mahasiswa');

        // mengambil data id yang dikirimkan dengan method data-id
        const id = $(this).data('id');

```

```

// ajax untuk mengambil data dari function getEdit
$.ajax({
    type : 'post',
    url : 'http://localhost/php_mvc/pertemuan6/public/mahasiswa/getEdit',
    dataType : 'json',
    data : {
        'id' : id
    },
    success : function(data){
        $('#id').val(data['id']);
        $('#nama').val(data['nama']);
        $('#nrp').val(data['nrp']);
        $('#email').val(data['email']);
        $('#jurusan').val(data['jurusan']);
    }
})
})

```

4. Dan juga kita akan mengubah jika diklik tambah data modalnya digunakan untuk menambahkan data karena isi dari htmlnya ketimpa oleh edit data mahasiswa.

```

// ketika kita klik tombol tambah data
$('.tambahDataMahasiswa').on('click',function(){
    $('.modal-title').html('Tambah Data Mahasiswa');
    $('.modal-footer button[type=submit]').html('Tambah Data');
    $('.modal-
content form').attr('action', 'http://localhost/php_mvc/pertemuan6/public/mahasiswa/tambah_Mahasiswa');

    $('#nama').val('');

```



```

$( '#nrp' ).val( '' );

$( '#email' ).val( '' );

$( '#jurusan' ).val( 'Sistem Informasi' );

})

```

5. Lalu kita ke route – Mahasiswa.php kita akan tambahkan method getEdit untuk mengambil data mahasiswa sesuai dengan idnya .

```

public function getEdit()

    // kita tangkap dulu data yang dikirimkan melalui post yaitu berupa id
    $id = $_POST['id'];

    // tampilkan dan kita rubah datanya menjadi json

    echo json_encode( $this->model('Mahasiswa_model')->getMahasiswaById($id) );

```

6. Sampai sini kita sudah berhasil menampilkan datanya kita akan membuat method untuk meng update datanya . kita ke file Mahasiswa.php didalam folder controller dan kita tambahkan function edit_Mahasiswa .

```

public function edit_Mahasiswa()

    if ( $this->model('Mahasiswa_model')->editDataMahasiswa($_POST) > 0 ){

        FlashMessage::setFlash('check-circle-fill','Berhasil','Di Edit', 'success');

        header('Location:' .URL. '/mahasiswa');

        exit;

    }else{

        FlashMessage::setFlash('exclamation-triangle-fill','Gagal','Di Edit', 'danger');

        header('Location:' .URL. '/mahasiswa');

        exit;

    }

```

7. Lalu kita membuat function editDataMahasiswa didalam model Mahasiswa_model.

```
public function editDataMahasiswa($data)
{
    $query = "UPDATE $this->table SET
        nama = :nama,
        nrp = :nrp,
        email = :email,
        jurusan = :jurusan
        WHERE id = :id";

    $this->db->query($query);
    $this->db->bind('id', $data['id']);
    $this->db->bind('nama', $data['nama']);
    $this->db->bind('nrp', $data['nrp']);
    $this->db->bind('email', $data['email']);
    $this->db->bind('jurusan', $data['jurusan']);

    $this->db->execute();

    return $this->db->rowCount();
}
```

SEARCH

1. Kita membuat form untuk search data .

```
<!-- form Search Data -->
<form action="<?=URL?>/mahasiswa/searchData" method="post">
  <div class="input-group mb-3">
    <input type="text" name="keyword" id="keyword" class="form-control" placeholder="Cari Mahasiswa"
    aria-label="Recipient's username" aria-describedby="button-addon2" autocomplete="off">
    <button class="btn btn-outline-primary" type="submit" id="tombolCari">Search </button>
  </div>
</form>
<!-- End Form Search Data -->
```

2. Lalu kita membuat method searchData didalam file mahasiswa controller.

```
public function searchData()
{
    $data['title'] = 'Data Mahasiswa';
    $data['active'] = 'active mahasiswa';
    $data['mahasiswa'] = $this->model('Mahasiswa_model')->searchDataMahasiswa();
    $this->view('template/header',$data);
    $this->view('mahasiswa/index',$data);
    $this->view('template/footer');
}
```

3. Lalu kita membuat function searchDataMahasiswa didalam model Mahasiswa.

```
public function searchDataMahasiswa()  
{  
    $keyword = $_POST['keyword'];  
    $query = "SELECT * FROM $this->table WHERE nama LIKE :keyword";  
  
    $this->db->query($query);  
    $this->db->bind('keyword', "%$keyword%");  
  
    return $this->db->resultSet();  
}
```