PEMROGRAMAN WEB

OOP PHP

Muarifin
http://muarifin.lecturer.pens.ac.id
muarifin@pens.ac.id

Konsep OOP

Object Oriented Programming (OOP) adalah sebuah tata cara pembuatan program (programming **paradigm**) dengan menggunakan konsep "**objek**" yang memiliki data (atribut yang menjelaskan tentang objek) dan prosedur (function) yang dikenal dengan method. (Wikipedia)

Konsep OOP

- Secara sederhananya, OOP adalah konsep pembuatan program dengan memecah permasalahan program dengan menggunakan **objek**.
- Berbeda dengan konsep **fungsi** atau '**function**' di dalam pemrograman, sebuah objek bisa memiliki data dan function tersendiri. Setiap objek ditujukan untuk mengerjakan sebuah tugas, dan menghasilkan nilai akhir untuk selanjutnya dapat ditampilkan atau digunakan oleh objek lain.

OOP pada PHP

PHP bukan bahasa pemrograman yang '**murni**' berbasis **objek** seperti **Java**. Bahkan, konsep OOP dalam PHP baru hadir dalam PHP versi 4, dan disempurnakan oleh PHP versi 5. Dengan kata lain, OOP di PHP merupakan '**fitur tambahan**'. Bahkan Developer bisa membuat situs web dengan PHP tanpa menggunakan konsep OOP sama sekali.

Dalam studi pemrograman, pembuatan program dalam PHP tanpa menggunakan objek disebut juga dengan pemrograman ***prosedural*** atau pemrograman ***fungsional***. Dinamakan dengan pemrograman prosedural, karena kita memecah kode program menjadi bagian-bagian atau fungsi-fungsi kecil, kemudian menyatukannya untuk menghasilkan nilai akhir.

OOP pada PHP

Dengan membuat program secara prosedural, aplikasi bisa dibuat dengan **cepat** dan mudah dipelajari jika dibandingkan dengan pemrograman berbasis objek. Keuntungan pemrograman berbasis objek baru terasa ketika program tersebut telah '**besar**' atau kita bekerja dengan **tim** untuk membagi tugas.

Sebagai programmer **web**, OOP adalah salah satu hal yang **wajib**. Pembuatan website modern saat ini akan lebih mudah jika menggunakan template kode program yang dikenal dengan **framework**. Dan, framework PHP hampir semuanya dibuat menggunakan OOP.

Class

- Class adalah ‘**cetakan**’ atau ‘**blueprint**’ dari **object**.
- Class digunakan hanya untuk membuat **kerangka** dasar.
- Yang akan kita **pakai** nantinya adalah **hasil cetakan** dari class, yakni object.
- Misalkan pada Contoh disamping adalah sebuah Class **Mobil** yang merepresentasikan **Mobil** kedalam Bahasa Pemrograman PHP

```php
<?php

class Mobil
{
    //isi dari Class Mobil
}
```

Property

- Property (atau disebut juga dengan atribut) adalah **data** yang terdapat dalam sebuah class.
- Melanjutkan analogi tentang **Mobil**, property dari Mobil bisa berupa merk, warna, jenis, kecepat maksimal, dan lain-lain.
- Berikut contoh imlementasi **Property** pada Class **Mobil**.

```php
<?php

class Mobil
{
    var $pemilik;
    var $merk;
    var $warna;
    var $jenis;

    //isi dari lanjutan Class Mobil
}
```

Method

- Method adalah **tindakan** yang bisa dilakukan oleh Class.
- Jika menggunakan analogi class **Mobil**, maka contoh method adalah: manyalakan Mobil, menjalankan Mobil, isi bahan bakar, dll

```php
<?php

class Mobil
{
    var $pemilik;
    var $merk;
    var $warna;
    var $jenis;

    function nyalakan_mobil(){
        echo "Mobile dinyalakan";
    }

    function jalankan_mobil(){
        echo "Mobil dijalankan";
    }
}
```

Object

- Object atau Objek adalah **hasil cetak** dari class, atau hasil '**konkrit**' dari class.
- Jika menggunakan analogi class **Mobil**, maka objek dari class **Mobil** bisa berupa: mobil_rika, mobil_anto, dll.
- Objek dari class **Mobil** akan memiliki seluruh ciri-ciri **Mobil**, yaitu property dan method-nya.

```php
<?php
use Mobil;

$mobil_rika= new Mobil();
$mobil_anto= new Mobil();
```

Mengakses Element Object

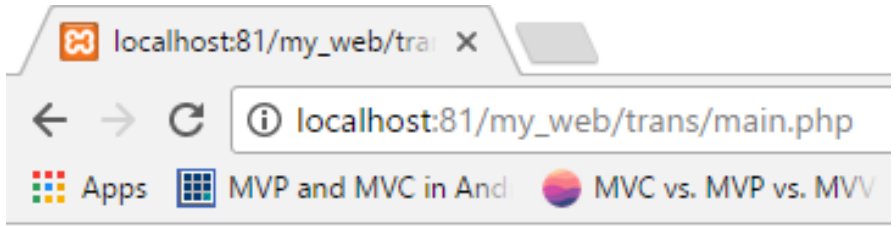
- Setelah object telah berhasil dibuat, maka selanjutnya adalah bagaimana agar object tersebut dapat beroperasi melakukan aksi tertentu untuk menyelesaikan masalah.
- Berikut adalah contoh untuk mengakses Property dan Method.

Mobile.php

```php
<?php

class Mobil
{
    var $pemilik;
    var $merk;
    var $warna;
    var $jenis;

    function nyalakan_mobil(){
        return "Mobile dinyalakan";
    }

    function jalankan_mobil(){
        return "Mobil dijalankan";
    }
}
```

index.php

```php
<?php
include 'Mobil.php';

$mobil_civer= new Mobil();

$mobil_civer->pemilik= "Civer Yoshioka";
$mobil_civer->warna="Kuning";
$mobil_civer->merk="Ford";
$mobil_civer->jenis="Sport Car";

echo "Pemilik : ". $mobil_civer->pemilik."<br>";
echo "Warna : ". $mobil_civer->warna."<br>";
echo "Merk : ". $mobil_civer->merk."<br>";
echo "Jenis : ". $mobil_civer->jenis."<br>";
echo "Keadaan Mobil  : ". $mobil_civer->jalankan_mobil()."<br>";
```



Pemilik : Civer Yoshioka
Warna : Kuning
Merk : Ford
Jenis : Sport Car
Keadaan Mobil : Mobil dijalankan

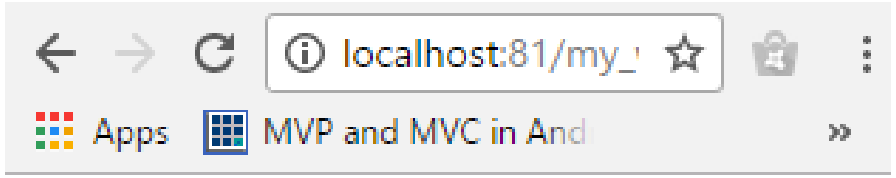
Encapsulation

- Enkapsulasi (encapsulation) adalah sebuah metode untuk **mengatur** cara mengakses struktur class dengan cara menyembunyikan alur kerja dari class tersebut.
- Enkapsulasi diterapkan dengan menggunakan 3 jenis hak akses: Public, Protected dan Private

```php
<?php

class Mobil
{
    private $pemilik;
    var $merk;
    var $warna;
    var $jenis;

    function nyalakan_mobil(){
        return "Mobile dinyalakan";
    }

    function jalankan_mobil(){
        return "Mobil dijalankan";
    }
}
```

index.php

```php
<?php
include 'Mobil.php';

$mobil_civer= new Mobil();

$mobil_civer->pemilik= "Civer Yoshioka";
$mobil_civer->warna="Kuning";
$mobil_civer->merk="Ford";
$mobil_civer->jenis="Sport Car";

echo "Pemilik : ". $mobil_civer->pemilik."<br>";
echo "Warna : ". $mobil_civer->warna."<br>";
echo "Merk : ". $mobil_civer->merk."<br>";
echo "Jenis : ". $mobil_civer->jenis."<br>";
echo "Keadaan Mobil  : ". $mobil_civer->jalankan_mobil()."<br>";
```

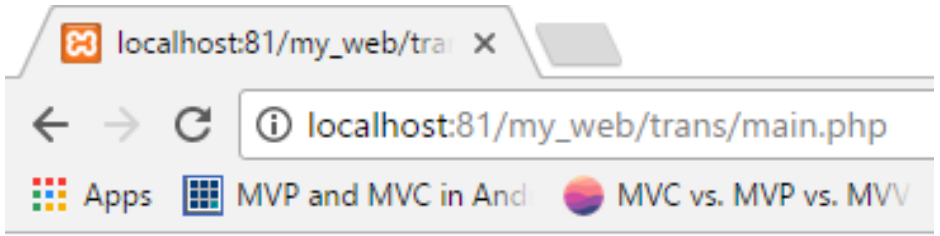


Fatal error: Uncaught Error: Cannot access private property Mobil::$pemilik in D:\xampp\htdocs\my_web\trans\main.php:6 Stack trace: #0 {main} thrown in **D:\xampp\htdocs\my_web\trans\main.php** on line **6**

- Untuk mengatasi Error karena akses variable pemilik dibatasi.
- Harus dibuatkan fungsi untuk mengakses variable **$pemilik**, sehingga Class Mobil akan diubah seperti berikut :

```php
<?php

class Mobil
{
    private $pemilik;
    var $merk;
    var $warna;
    var $jenis;

    function nyalakan_mobil(){
        return "Mobile dinyalakan";
    }

    function jalankan_mobil(){
        return "Mobil dijalankan";
    }

    function setPemilik($pemilik){
        $this->pemilik=$pemilik;
    }

    function getPemilik(){
        return $this->pemilik;
    }
}
```

```php
<?php
include 'Mobil.php';

$mobil_civer= new Mobil();

$mobil_civer->setPemilik("Civer Yoshioka");
$mobil_civer->warna="Kuning";
$mobil_civer->merk="Ford";
$mobil_civer->jenis="Sport Car";

echo "Pemilik : ". $mobil_civer->getPemilik()."<br>";
echo "Warna : ". $mobil_civer->warna."<br>";
echo "Merk : ". $mobil_civer->merk."<br>";
echo "Jenis : ". $mobil_civer->jenis."<br>";
echo "Keadaan Mobil  : ". $mobil_civer->jalankan_mobil()."<br>";
```



Pemilik : Civer Yoshioka
Warna : Kuning
Merk : Ford
Jenis : Sport Car
Keadaan Mobil : Mobil dijalankan

OPP

- Inheritance
- Polymorphism

Muarifin
http://muarifin.lecturer.pens.ac.id
muarifin@pens.ac.id

TERIMAKASIH

Muarifin
http://muarifin.lecturer.pens.ac.id
muarifin@pens.ac.id