Contents

[Scope Variable PHP 3](#_Toc159491493)

[Global 3](#_Toc159491494)

[Local 4](#_Toc159491495)

[Static 5](#_Toc159491496)

[Tipe Data/Variabel di PHP 6](#_Toc159491497)

[String 6](#_Toc159491498)

[Single quote 6](#_Toc159491499)

[Double quote 6](#_Toc159491500)

[Arrays 7](#_Toc159491501)

[Define Array 7](#_Toc159491502)

[Indexed Array 7](#_Toc159491503)

[Associative Array 7](#_Toc159491504)

[Multidimensional Array 7](#_Toc159491505)

[Constant 8](#_Toc159491506)

[Define Constant 8](#_Toc159491507)

[Redeclare Constant 8](#_Toc159491508)

[Scope Constant 8](#_Toc159491509)

[Operator 9](#_Toc159491510)

[Asignment Operator 9](#_Toc159491511)

[Comparasion 10](#_Toc159491512)

[Perbedaan Equal(==) dan Identical (===) 10](#_Toc159491513)

[Comparasion Ternary 10](#_Toc159491514)

[Comparasion, Null coalescing 10](#_Toc159491515)

[Increment & Decrement 11](#_Toc159491516)

[Logical Operator Xor 12](#_Toc159491517)

[String Operator 13](#_Toc159491518)

[Concat / Concatenation 13](#_Toc159491519)

[Concatenation assignment 13](#_Toc159491520)

[Looping 14](#_Toc159491521)

[For 14](#_Toc159491522)

[Foreach 14](#_Toc159491523)

[While 14](#_Toc159491524)

[Do While 15](#_Toc159491525)

[Contoh penggunaan 15](#_Toc159491526)

[Function 17](#_Toc159491527)

[Membuat dan memanggil function 17](#_Toc159491528)

[Parameter/Argument 17](#_Toc159491529)

[GET & POST 18](#_Toc159491530)

[Perbedaan GET dan POST 18](#_Toc159491531)

[Mendapatkan nilai 18](#_Toc159491532)

[Action 19](#_Toc159491533)

[SESSION 19](#_Toc159491534)

[Start session 19](#_Toc159491535)

[Unset session 19](#_Toc159491536)

[Destroy session 20](#_Toc159491537)

[Penulisan yg di rekomendasikan 20](#_Toc159491538)

[Login dengan Session 20](#_Toc159491539)

[Include & Require 22](#_Toc159491540)

[Perbedaan include dan require 22](#_Toc159491541)

[OOP pada PHP 23](#_Toc159491542)

[Perbedaan OOP dan procedural 23](#_Toc159491543)

[Class dan Object 23](#_Toc159491544)

[Membuat Class dan Object 24](#_Toc159491545)

[Method/ function 24](#_Toc159491546)

[Constructor & Destructor 25](#_Toc159491547)

[Unset Object 27](#_Toc159491548)

[Inheritance 27](#_Toc159491549)

[Mengakses property & method dari child class 28](#_Toc159491550)

[Override Method 29](#_Toc159491551)

[Keyword Final 30](#_Toc159491552)

[Static function 30](#_Toc159491553)

[Perbedaan self dan $this 31](#_Toc159491554)

[Access Modifer(public,private,protected) 32](#_Toc159491555)

[Public 32](#_Toc159491556)

[Private 32](#_Toc159491557)

[Protected 33](#_Toc159491558)

[Traits 34](#_Toc159491559)

[Membuat dan use Trait 35](#_Toc159491560)

# Scope Variable PHP

## Global

Variable yg di define di luar function dan hanya bisa di gunakan di luar function juga

OK:

<?php

    $panjang = 5;

    $lebar = 3;

    function myFunction(){

    }

    echo $panjang;

    echo '<br/>';

    echo $lebar;

?>

Error:

<?php

    $panjang = 5;

    $lebar = 3;

    function myFunction(){

        echo $panjang;

        echo '<br/>';

        echo $lebar;

    }

    myFunction();

?>

Jika tetap ingin menggunakan variable global dalam function gunakan keyword global

<?php

    $panjang = 5;

    $lebar = 3;

    function myFunction(){

        global $panjang, $lebar;

        echo $panjang;

        echo '<br/>';

        echo $lebar;

    }

    myFunction();

?>

## Local

Di define di tempat tertentu(function) dan hanya bisa di pakai di tempat itu/tertentu/dalam function

Ok:

<?php

    function myFunction(){

        $sisi = 5;

        echo $sisi;

    }

    myFunction();

?>

Error:

<?php

function myFunction(){

        $sisi = 5;

    }

    myFunction();

    echo $sisi;

?>

## Static

Mempertahankan nilai/kondisi dari suatu variable ketika function di panggil kembali

Contoh tanpa static

<?php

    function myFunction()

    {

        $x = 1;

        echo $x;

        $x++;

    }

    myFunction();

    echo '<br/>';

    myFunction();

?>

Outputnya:

1  
1

Pakai Static:

<?php

    function myFunction()

    {

        static $x = 1;

        echo $x;

        $x++;

    }

    myFunction();

    echo '<br/>';

    myFunction();

?>

Outputnya:

1  
2

Penjelasan:

Ketika myFunction di panggil untuk kedua kali, variabel $x tidak sama dengan 1, melainkan sudah menjadi 2, karena sebelumnya sudah $x++ saat pemanggilan pertama

# Tipe Data/Variabel di PHP

## String

Perbedaan single quote (‘) dan double quote (“)

### Single quote

<?php

    $x = "bebek";

    var\_dump('ada 10 $x berenang');

?>

Output:

string(18) "ada 10 $x berenang"

### Double quote

<?php

    $x = "bebek";

    var\_dump("ada 10 $x berenang");

?>

Output:

string(21) "ada 10 bebek berenang"

Jadi pada single quote, nilai $x benar2 diangap sebagai bagian dari string, sedangkan double quote kita bisa mesisipkan nilai variabel

## Arrays

Variabel yang dapat menampung lebih dari satu value/nilai

### Define Array

<?php

    $x = array('apel', 'jeruk', 'pisang');

    // atau

    $y = ['apel', 'jeruk', 'pisang'];

### Indexed Array

Array pada umumnya, index array di mulai dari 0

<?php

     $siswa = ['a','b', 'c'];

     echo $siswa[0]; // print nilai 'a'

     echo count($siswa); // banyak nilai dari array = 3

?>

### Associative Array

Array yg menggunakan key, saat di panggil kita bisa pakai key nya

<?php

     $nilaiBudi = [

          'matematika' => 9,

          'bahasa' => 8,

          'fisika' => 7

     ];

     echo $nilaiBudi['matematika']; // 9

?>

### Multidimensional Array

Ada array di dalam array

<?php

     // multi dimensional with associative array

     $siswa = [

          ["nama" => "Budi", "kelas" => "XI", "jurusan" => "IPA"],

          ["nama" => "Sinta", "kelas" => "XI", "jurusan" => "IPS"],

          ["nama" => "Jojo", "kelas" => "XI", "jurusan" => "Bahasa"],

          ["nama" => "Toni", "kelas" => "XI", "jurusan" => "Seni"],

     ];

     echo count($siswa); // 4

     echo $siswa[1]['nama']; // Sinta

     echo $siswa[1]['jurusan']; // IPS

     // multi dimensional with index array

     $siswa2 = [

          ["Budi", "XI", "IPA"],

          ["Sinta", "XI", "IPS"],

          ["Jojo", "XI", "Bahasa"],

          ["Toni", "XI", "Seni"],

     ];

     echo $siswa2[1][0]; // Sinta

     echo $siswa2[1][2]; // IPS

?>

## Constant

### Define Constant

<?php

   //declare const

   define('phi', 3.14);

   echo phi;

?>

Param 1= nama konstant nya

Param 2=nilai/ value nya

Untuk echo nya, langsung nama constant nya saja, tanpa $

### Redeclare Constant

<?php

    //declare variable

    $phi = 3.14;

    $phi = 5;

    echo($phi);

    echo('<br/>');

    //declare const

    define('phi', 3.14);

    phi = 9 // disini akan error, krn tdk bisa di declare ulang/ hanya bisa 1x

    echo phi;

?>

### Scope Constant

Constant tidak butuh scope

<?php

    //declare variable

    $phi = 3.14;

    //declare const

    define('phi', 3.14);

   function printVar()

   {

        global $phi;

        echo $phi;

   }

   function printConst()

   {

        echo phi;

   }

   printVar();

   echo '<br/>';

   printConst();

?>

Kalau pada variabel kita peru menggunakan global agar bisa akses, pada constant, kita bisa langsung panggil nama constant nya saja

# Operator

## Asignment Operator

Addition (a+=b)

<?php

    $a = 4;

    $b = 3;

    $a += $b;

    echo $a; // hasil = 7

?>

Penjelasan $a +=$b sama dengan $a = $a+$b

Subtraction(a-=b)

Multiplication (a\*=b)

Division(a/=b)

Modulus (a%=b)

## Comparasion

Menghasilkan nilai Boolean

## Perbedaan Equal(==) dan Identical (===)

<?php

    $a = 5;

    $b = "5";

    var\_dump($a == $b); // True

    var\_dump($a === $b); // False

?>

Identical (===) juga membandingkan tipe data nya

Lawan nya Not Equal (!=) dan not identical (!==)

### Comparasion Ternary

Penulisan singkat / 1 baris nya if else

<?php

     $hasil = gettype(5) == 'integer' ? 'ini adalah integer' : 'bukan integer';

     echo $hasil;

?>

Ada 3 bagian:

* gettype(5) == 'integer' merupakan kondisi
* ‘ini adalah integer’ kondisi kalau True
* ‘bukan integer’ kondisi kalau False

### Comparasion, Null coalescing

<?php

     $nama = 'Fajar';

     $alias = 'Cara Fajar';

     echo($nama ?? $alias); // Fajar

?>

<?php

     // $nama = 'Fajar';

     $alias = 'Cara Fajar';

     echo($nama ?? $alias); // Cara Fajar

?>

Ada dua kondisi:

* $nama , jika $nama ada isi / tidak NULL, gunakan $nama
* $alias, jika $nama NULL / undefined, gunakan $alias

## Increment & Decrement

Pre-Increment (++$a), ditambah 1 dulu, baru di return/print/echo

<?php

    $a = 1;

    echo(++$a); // $a = 2

    echo($a);  // $a = 2

?>

Post-Increment ($a++), di return/print/echo dulu, baru di tambah 1

<?php

    $a = 1;

    echo($a++); // $a = 1

    echo($a);  // $a = 2

?>

Pre-Decrement (--$a), dikurang 1 dulu, baru di return/print/echo

<?php

    $a = 10;

    echo(--$a); // $a = 9

    echo($a);  // $a = 9

?>

Post-Decrement ($a--), di return/print/echo dulu, baru di kurang 1

<?php

    $a = 10;

    echo($a--); // $a = 10

    echo($a);  // $a = 9

?>

## Logical Operator Xor

Return true jika salah satu nya benar, tapi bukan keduanya benar

True xor False = True

<?php

    $a = 10;

    $b = 5;

    var\_dump($a > 8 xor $b > 8); // -> di return true

          //true       //false

?>

True xor True = False

<?php

    $a = 10;

    $b = 5;

    var\_dump($a > 8 xor $b > 3); // -> di return false

          //true       //true

?>

False xor False = False

<?php

    $a = 10;

    $b = 5;

    var\_dump($a > 20 xor $b > 20); // -> di return false

            //false       //false

?>

## String Operator

### Concat / Concatenation

Mengabungkan string $x.$y

<?php

    $a = 'Hello';

    $b = ' World';

     echo($a.$b); // Hello World

?>

### Concatenation assignment

$x .= $y artinya menambah/menempelkan $y ke $x

<?php

    $a = 'Hello';

    $b = ' World';

     echo($a .= $b); // Hello World

?>

# Looping

## For

<?php

     for ($i=1; $i <= 100; $i++) {

          echo $i;

          echo '<br/>';

     }

?>

## Foreach

Hanya bisa digunakan untuk array

<?php

     $warna = ['merah', 'kuning', 'hijau', 'biru'];

     foreach ($warna as $key => $value) {

          echo $value;

          echo '<br/>';

     }

?>

## While

<?php

     $a = 1;

     while ($a <= 100) {

          echo $a;

          echo '<br/>';

          $a++;

     }

?>

## Do While

<?php

     $a = 1;

     do {

          echo $a;

          echo '<br/>';

          $a++;

     } while ($a <= 100);

?>

Perbedaan nya dengan while

<?php

     $a = 1;

     do {

          echo $a;

          echo '<br/>';

          $a++;

     } while ($a < 1);

?>

Contoh diatas akan tetap print angka 1, karena pada do while, di lakukan dulu proses dalam do nya baru di validasi kondisi while nya

## Contoh penggunaan

<?php

    $siswa = [

     ["nama" => "Budi", "kelas" => "XI", "jurusan" => "IPA"],

     ["nama" => "Sinta", "kelas" => "XI", "jurusan" => "IPS"],

     ["nama" => "Jojo", "kelas" => "XI", "jurusan" => "Bahasa"],

     ["nama" => "Toni", "kelas" => "XI", "jurusan" => "Seni"],

];

?>

<!DOCTYPE html>

<html lang="en">

<head>

     <meta charset="UTF-8">

     <meta name="viewport" content="width=device-width, initial-scale=1.0">

     <title>Document</title>

</head>

<style>

     table{

          border-collapse:collapse;

     }

</style>

<body>

     <h2>List Siswa</h2>

     Pakai Foreach

     <table border=1>

          <tr>

               <th>Nama</th>

               <th>Kelas</th>

               <th>Jurusan</th>

          </tr>

          <?php foreach($siswa as $key=>$v){

          ?>

               <tr>

                   <td> <?php echo $v['nama']; ?> </td>

                   <td> <?php echo $v['kelas']; ?> </td>

                   <td> <?php echo $v['jurusan']; ?> </td>

               </tr>

          <?php

          }?>

     </table>

     <br/>

     Pakai While

     <table border=1>

          <tr>

               <th>Nama</th>

               <th>Kelas</th>

               <th>Jurusan</th>

          </tr>

          <?php

               $a = 0;

               while($a < count($siswa)){

          ?>

               <tr>

                   <td> <?php echo $siswa[$a]['nama']; ?> </td>

                   <td> <?php echo $siswa[$a]['kelas']; ?> </td>

                   <td> <?php echo $siswa[$a]['jurusan']; ?> </td>

               </tr>

          <?php

               $a++;

               }

          ?>

     </table>

</body>

</html>

# Function

## Membuat dan memanggil function

<?php

function menyapa(){

     echo 'selamat pagi';

}

menyapa();

?>

Atau

<?php

 menyapa();

function menyapa(){

     echo 'selamat pagi';

}

?>

## Parameter/Argument

<?php

 menyapa('budi');

 echo '<br>';

 menyapa('sarah');

function menyapa($nama){

     echo 'selamat pagi '.$nama;

}

?>

Dengan nilai default

<?php

 menyapa('budi',18);

 echo '<br>';

 menyapa('sarah',20);

 echo '<br>';

 menyapa('rendi');

function menyapa($nama, $umur=17){

     echo 'selamat pagi '.$nama. ' umur kamu ' .$umur;

}

?>

Hasil

selamat pagi budi umur kamu 18  
selamat pagi sarah umur kamu 20  
selamat pagi rendi umur kamu 17

# GET & POST

## Perbedaan GET dan POST

GET

<form action="" method="get">

Ketika di submit, di URL akan terdapat query string

<http://localhost:8081/belajar-php?nama=budi&jurusan=ipa&submit=kirim>

POST

<form action="" method="post">

Ketika di submit, URL nya tetap

<http://localhost:8081/belajar-php/>

\*disarankan menggunakan method POST, agar data tidak terlihat di URL, gunakan GET untuk parameter data yg tidak rahasia

## Mendapatkan nilai

<?php

      echo $\_GET['nama'];

      echo $\_GET['jurusan'];

?>

<?php

       echo $\_POST['nama'];

       echo $\_POST['jurusan'];

?>

Bisa pakai validasi isset agar tidak muncul warning

<button type="submit" name="buttonKirim" value= "kirim"> Kirim</button>

<?php

          if(isset($\_POST['buttonKirim'])){

               echo $\_POST['nama'];

               echo $\_POST['jurusan'];

          }

?>

## Action

Mengirim ke halaman lain

<form action="proses.php" method="post">

Berarti ketika di submit, akan pindah ke proses.php

# SESSION

Menyimpan nilai / variable dan bisa di gunakan di beberapa halaman

## Start session

<?php

     session\_start();

     $\_SESSION['nama'] = 'jhon';

     var\_dump($\_SESSION['nama']); // jhon

     $\_SESSION['nama'] = 'budi';

     var\_dump($\_SESSION['nama']); // di overwrite jadi budi

?>

Di halaman proses.php, juga akan di kenal $\_SESSION nya

<?php

    session\_start();

    echo $\_SESSION['nama'];

?>

## Unset session

Menghapus nilai session

<?php

     session\_start();

     $\_SESSION['nama'] = 'jhon';

     $\_SESSION['jurusan'] = 'IPA';

     echo $\_SESSION['nama']; // jhon

     echo $\_SESSION['jurusan']; // IPA

     session\_unset();

     echo $\_SESSION['nama']; // Warning: Undefined array key "nama"

     echo $\_SESSION['jurusan']; // Warning: Undefined array key "jurusan"

?>

## Destroy session

<?php

     session\_start();

     $\_SESSION['nama'] = 'jhon';

     $\_SESSION['jurusan'] = 'IPA';

     echo $\_SESSION['nama']; // jhon

     echo $\_SESSION['jurusan']; // IPA

     session\_destroy();

     echo $\_SESSION['nama']; // jhon

     echo $\_SESSION['jurusan']; // IPA

?>

Session destroy hanya mengakhiri session\_start(), tetapi nilai nya masih tetap ada

## Penulisan yg di rekomendasikan

<?php

     session\_start();

     $\_SESSION['nama'] = 'jhon';

     $\_SESSION['jurusan'] = 'IPA';

     echo $\_SESSION['nama']; // jhon

     echo $\_SESSION['jurusan']; // IPA

     session\_unset();

     session\_destroy();

?>

## Login dengan Session

Index.php

<!DOCTYPE html>

<html lang="en">

<head>

     <meta charset="UTF-8">

     <meta name="viewport" content="width=device-width, initial-scale=1.0">

     <title>Document</title>

</head>

<body>

     <h2>Login</h2>

     <form action="" method="post">

          <label for="">Username</label>

          <input type="text" name="username" autocomplete="off">

          <br>

          <label for="">Password</label>

          <input type="password" name="password">

          <br>

          <input type="submit" value="login" name="loginButton">

     </form>

</body>

<?php

     if(isset($\_POST['loginButton'])){

          session\_start();

          $\_SESSION['username'] = $\_POST['username'];

          $\_SESSION['login'] = true;

          header("location:proses.php");

     }

?>

</html>

Proses.php

<?php

    session\_start();

    if(isset($\_SESSION['login'])== true){

        echo "sukses login, welcome " . $\_SESSION['username'];

    } else {

        header("location:index.php");

    }

?>

# Include & Require

Menyisipkan file

Include.php

<?php

echo 'ini dari include.php';

home.php

<?php

    include "include.php";

ketika akses <http://localhost:8081/belajar-php/home.php>

ini dari include.php

menggunakan require juga bisa

<?php

    require "include.php";

## Perbedaan include dan require

Pada require jika ada error, kita tidak dapat mengakses halaman tersebut, krn fatal error

**Warning**: require(ceklogins.php): Failed to open stream: No such file or directory in **D:\xampp8\htdocs\belajar-php\proses.php** on line **2**  
  
**Fatal error**: Uncaught Error: Failed opening required 'ceklogins.php' (include\_path='D:\xampp8\php\PEAR') in D:\xampp8\htdocs\belajar-php\proses.php:2 Stack trace: #0 {main} thrown in **D:\xampp8\htdocs\belajar-php\proses.php** on line **2**

Pada include, jika ada error, tetap bias mengakses halaman, krn Cuma di tampilkan warning

**Warning**: include(ceklogins.php): Failed to open stream: No such file or directory in **D:\xampp8\htdocs\belajar-php\proses.php** on line **2**  
  
**Warning**: include(): Failed opening 'ceklogins.php' for inclusion (include\_path='D:\xampp8\php\PEAR') in **D:\xampp8\htdocs\belajar-php\proses.php** on line **2**

anda sukses login

\* Untuk sisi keamanan lebih disarankan pakai require

# OOP pada PHP

## Perbedaan OOP dan procedural

Salah satunya, pada procedural, code di tulis berurutan dari atas ke bawah

Pada OOP, membuat object yg di dalamnya bisa menampung data dan fungsi/method

## Class dan Object

Class adalah blueprint / cetak awal dari object

Object adalah instance / hasil pembuatan dari class

Contoh:

Kelas Robot memiliki 3 object yakni robot 1 sampai 3

Robot 1 s/d 3 tidak akan terbentuk tanpa class robot

Class Robot memiliki property Nama, warna, bentuk

Object Robot 1 -3 memilki property yang sama

### Membuat Class dan Object

Model/Robot.php

<?php

class Robot{

    //property initiate

    public $name, $color, $size;

}

Index.php

<?php

require 'Model/Robot.php';

$robot1 = new Robot();

// assign value to property

$robot1->name = 'optimus prime';

$robot1->color = 'blue-white-red';

$robot1->size = 'large';

$robot2 = new Robot();

$robot2->name = 'blumble-bee';

$robot2->color = 'yellow';

$robot2->size = 'medium';

var\_dump($robot1); // object(Robot)#1 (3) { ["name"]=> string(13) "optimus prime" ["color"]=> string(14) "blue-white-red" ["size"]=> string(5) "large" }

echo '<br>';

var\_dump($robot2);

### Method/ function

Model/Robot.php

<?php

class Robot{

    //property initiate

    public $name, $color, $size;

    public function setName($name){

        $this->name = $name;

    }

    public function setColor($color){

        $this->color = $color;

    }

    public function setSize($size){

        $this->size = $size;

    }

    public function getName(){

        return $this->name;

    }

}

Index.php

<?php

require 'Model/Robot.php';

$robot1 = new Robot();

$robot1->setName('optimus prime');

$robot1->setColor('blue white red');

$robot1->setSize('large');

$robot2 = new Robot();

$robot2->setName('bumblebee');

$robot2->setColor('yellow');

$robot2->setSize('medium');

var\_dump($robot1);

echo '<br>';

var\_dump($robot2);

echo '<br>';

echo $robot1->getName();

## Constructor & Destructor

Constructor, fungsi yg berjalan pertama x ketika object dibuat

Syntax: \_\_construct()

Destructor, fungsi yg berjalan ketika object sudah selesai, atau code2 di class itu sudah semua, pada destruct tidak boleh ada parameter. Bila object di hancurkan/unset, juga akan memanggil destructor

Syntax: \_\_destructor()

Contoh construct

Model/Robot.php

<?php

class Robot{

    //property initiate

    public $name, $color, $size;

    function \_\_construct($name, $color, $size){

        $this->name = $name;

        $this->color = $color;

        $this->size = $size;

    }

Index.php

<?php

require 'Model/Robot.php';

$robot1 = new Robot('optimus prime', 'blue white red', 'large');

var\_dump($robot1);

Contoh destruct

Model/Robot.php

function \_\_destruct(){

        echo 'akhir panggil destruct';

    }

Index.php

<?php

require 'Model/Robot.php';

$robot1 = new Robot('optimus prime', 'blue white red', 'large');

var\_dump($robot1);

echo('<br>');

echo('next line ...');

echo('<br>');

Hasil:

object(Robot)#1 (3) { ["name"]=> string(13) "optimus prime" ["color"]=> string(14) "blue white red" ["size"]=> string(5) "large" }  
next line ...  
akhir panggil destruct

### Unset Object

Index.php

<?php

require 'Model/Robot.php';

$robot1 = new Robot('optimus prime', 'blue white red', 'large');

var\_dump($robot1);

echo('<br>');

unset($robot1); // destroy object

echo('<br>');

echo('next line ...');

echo('<br>');

Hasil:

object(Robot)#1 (3) { ["name"]=> string(13) "optimus prime" ["color"]=> string(14) "blue white red" ["size"]=> string(5) "large" }  
akhir panggil destruct  
next line ...

## Inheritance

Parent class mewariskan property dan method ke class children. Si child class juga bisa memiliki property & method nya sendiri

Model/Orang.php

<?php

class Orang {

    public $nama, $umur;

    public function \_\_construct($nama, $umur)

    {

        $this->nama = $nama;

        $this->umur = $umur;

    }

}

Model/Guru.php

<?php

require 'Model/Orang.php';

class Guru extends Orang{

}

Index.php

<?php

require 'Model/Guru.php';

$guru1 = New Guru('Budi', 39);

var\_dump($guru1);

### Mengakses property & method dari child class

Model/Orang.php

<?php

class Orang {

    public $nama, $umur;

    public function \_\_construct($nama, $umur)

    {

        $this->nama = $nama;

        $this->umur = $umur;

    }

    public function sayHello()

    {

        return "halo, nama saya {$this->nama} dan umur saya {$this->umur} tahun";

    }

}

Model/Guru.php

<?php

require 'Model/Orang.php';

class Guru extends Orang{

    public function aktifitasGuru(){

        echo 'saya mengajar TIK';

    }

}

Index.php

require 'Model/Guru.php';

$guru1 = New Guru('Budi', 39);

echo $guru1->nama; //children memanggil properti parent(Orang.php)\*/

echo '<br/>';

echo $guru1->sayHello(); //children memanggil method parent(Orang.php)\*/

echo '<br/>';

echo $guru1->aktifitasGuru(); // memanggil method sendiri / yg ada children class

### Override Method

Model/Orang.php

<?php

class Orang {

    public $nama, $umur;

    public function \_\_construct($nama, $umur)

    {

        $this->nama = $nama;

        $this->umur = $umur;

    }

    public function sayHello()

    {

        return "halo, nama saya {$this->nama} dan umur saya {$this->umur} tahun";

    }

}

Model/Guru.php

<?php

require 'Model/Orang.php';

class Guru extends Orang{

    public function aktifitasGuru(){

        echo 'saya mengajar TIK';

    }

    public function sayHello(){ //mengoverride method parent

        echo 'hallo saya adalah guru';

    }

}

Index.php

require 'Model/Guru.php';

$guru1 = New Guru('Budi', 39);

echo $guru1->sayHello();

Hasil:

hallo saya adalah guru

### Keyword Final

Membuat children tidak bisa mengoverride method dari parent

Model/Orang.php

final public function sayHello()

    {

        return "halo, nama saya {$this->nama} dan umur saya {$this->umur} tahun";

    }

Hasil jika tetap di override:

**Fatal error**: Cannot override final method Orang::sayHello() in **D:\xampp8\htdocs\belajar-php\Model\Guru.php** on line **10**

## Static function

Static function adalah method yg bisa diakses langsung tanpa membuat instance/object dahulu

class Guru {

static public function StaticMethod(){

        echo 'saya adalah static';

    }

Index.php

<?php

require 'Model/Guru.php';

Guru::StaticMethod();

## Perbedaan self dan $this

Self digunakan untuk mengakses static function, sedangkan $this untuk mengakses non-static function

Lebih jelas nya <https://phppot.com/php/php-self-vs-this/>

<?php

class Toys

{

    public $toys\_name;

    public $toys\_category;

    public static $shop\_name;

    public function \_\_construct($name, $category)

    {

        $this->toys\_name = $name;

        $this->toys\_category = $category;

    }

    public function getToyName()

    {

        return $this->toys\_name;

    }

    public function getToyCategory()

    {

        return $this->toys\_category;

    }

    public function getToyShop\_nonStatic()

    {

        return self::getToyShop();

    }

    public static function getToyShop()

    {

        return self::$shop\_name;

    }

    public static function setToyShop($shopname)

    {

        self::$shop\_name = $shopname;

    }

}

$objToys = new Toys("Battery Car", "Battery Toys");

$toys\_name = $objToys->getToyName();

$toys\_category = $objToys->getToyCategory();

echo "<br/>Toy: " . $toys\_name . ", Category: " . $toys\_category;

Toys::$shop\_name = "Disney";

$shop\_name = Toys::getToyShop();

echo "<br/>Shop Name: " . $shop\_name;

Toys::setToyShop("ToyShop");

$shopname = Toys::getToyShop\_nonStatic();

echo "<br/>Shop Name via non static function: " . $shopname;

## Access Modifer(public,private,protected)

### Public

Bentuk default, bisa di akses dari mana saja

<?php

class Makanan {

    public $nama = 'Semangka';

}

// diluar class

$makanan1 = new Makanan();

echo $makanan1->nama;

### Private

Hanya bisa di akses di class itu saja.

Kalau mau di akses dari luar class, bisa menggunakan public method

<?php

class Makanan {

    public $nama = 'Semangka';

    private $jenis = 'buah';

    public function getJenis() {

        return $this->jenis;

    }

}

// diluar class

$makanan1 = new Makanan();

echo $makanan1->nama;

// echo $makanan1->jenis; //Fatal error: Uncaught Error: Cannot access private property

echo '<br/>';

echo $makanan1->getJenis(); // buah

### Protected

Hanya bisa di akses di class itu dan turunan nya(child class)

Untuk mengakses nya, harus di extends dan menggunakan public method

<?php

class Makanan {

    public $nama = 'Semangka';

    private $jenis = 'buah';

    protected $harga = 15000;

    public function getJenis() {

        return $this->jenis;

    }

}

class Buah extends Makanan {

    public function getHarga(){

        return $this->harga;

    }

}

// diluar class

// echo $makanan1->harga; //Fatal error: Uncaught Error: Cannot access protected property

$buah1 = new Buah();

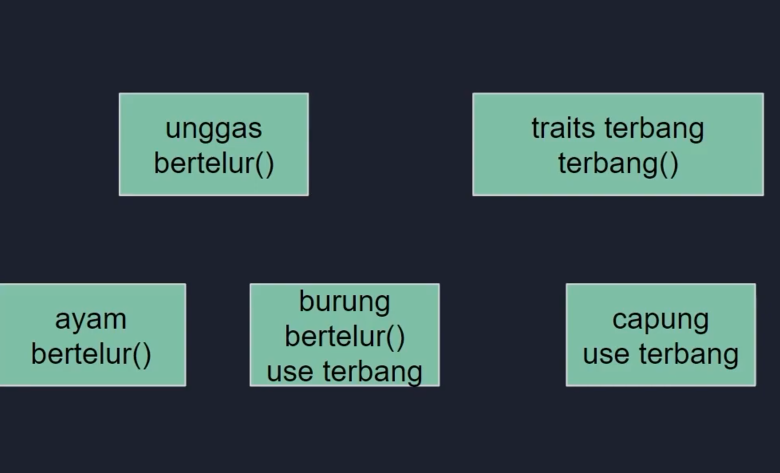
// echo $buah1->harga; //Fatal error: Uncaught Error: Cannot access protected property

echo $buah1->getHarga(); // 15000

## Traits

Digunakan ketika suatu class ingin meng-inheritance method yg berasal dari 2 atau lebih sumber yg berbeda

Karena Class hanya bias meng-extend 1 class parent (hanya punya 1 parent), kita bisa pakai Traits



Contoh ada 4 class, Class unggas, ayam, burung, capung

Class Ayam dan Burung inheritance / extend ke class unggas

Class Capung berdiri sendiri

Pada Class ayam dan burung, kita dapat mengiheritance method bertelur, krn kedua nya bisa bertelur

Ada kebutuhan membuat method terbang di burung dan capung, disini kita tidak bisa mendefine method terbang di unggas, krn nanti di class ayam jadi bisa terbang

Oleh karena itu agar tetap 1x tulis method terbang, kita gunakan Traits bernama terbang, yg di use di class burung dan capung

Kenapa tidak membuat Class Terbang? Karena class hanya bisa extend 1x, tidak bisa Class Burung extends Unggas extends Terbang

Index.php

require 'Model/Traits/Ayam.php';

require 'Model/Traits/Burung.php';

$ayam = new Ayam();

$burung = new Burung();

Unggas.php

<?php

class Unggas{

    function bertelur(){

        echo 'saya bertelur';

    }

}

Ayam.php

<?php

require\_once 'Unggas.php';

class Ayam extends Unggas {

}

Burung.php

<?php

require\_once 'Unggas.php';

class Burung extends Unggas {

}

Kita menggunakan require\_once karena di index kita menggunakan Ayam dan Burung, yg sama2 memanggil Unggas.php, agar tidak 2 x declare

### Membuat dan use Trait

Terbang.php

<?php

trait Terbang {

    function terbang(){

        echo 'saya bisa terbang';

    }

}

Burung.php

<?php

require\_once 'Unggas.php';

require\_once 'Terbang.php';

class Burung extends Unggas {

    use Terbang;

}

Capung.php

<?php

require\_once 'Terbang.php';

class Capung{

    use Terbang;

}

Index.php

require 'Model/Traits/Ayam.php';

require 'Model/Traits/Burung.php';

require 'Model/Traits/Capung.php';

$ayam = new Ayam();

$burung = new Burung();

$capung = new Capung();

echo $ayam->bertelur();

echo '<br>';

echo $burung->bertelur();

echo '<br>';

echo $burung->terbang();

echo '<br>';

echo $capung->terbang();