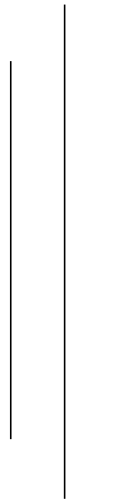


**MODUL MATA KULIAH  
PEMROGRAMAN WEB LANJUT**



**Disusun Oleh:**

Ajib Susanto

Abu Salam

Junta Zeniarja

**FAKULTAS ILMU KOMPUTER  
UNIVERSITAS DIAN NUSWANTORO  
2019**

## DAFTAR ISI

<b>BAB 1 WEB DINAMIS .....</b>	<b>5</b>
WEB DINAMIS .....	5
WEB SERVER.....	6
SERVER SIDE SCRIPTING .....	7
INSTALASI WEB SERVER & SERVER SIDE SCRIPTING.....	8
<b>BAB 2 DASAR PEMROGRAMAN PHP .....</b>	<b>15</b>
PHP.....	15
SINTAK DASAR .....	16
<i>Pemisahan Instruksi</i> .....	17
<i>Komentar / Remarks</i> .....	18
TYPE DATA .....	19
<i>Tipe Integer</i> .....	20
<i>Tipe Pecahan / Floating Point</i> .....	20
<i>Tipe String</i> .....	21
VARIABEL & KONSTANTA .....	24
<i>Penamaan Variable</i> .....	24
<i>Predefine Variable</i> .....	25
<i>Lingkup Variable (Scope)</i> .....	27
<i>Keyword Global</i> .....	28
<i>Keyword Static</i> .....	29
<i>Konstanta</i> .....	30
<i>Perbedaan Antara Konstanta Dan Variable:</i> .....	30
EKSPRESI .....	31
OPERATOR.....	32
<i>Operator Aritmatika</i> .....	34
<i>Hirarki Operator Aritmatika</i> .....	36
<i>Operator Pemberi Nilai</i> .....	36
<i>Operator Bitwise</i> .....	38
<i>Operator Perbandingan</i> .....	39
<i>Operator Kontrol Kesalahan (Error)</i> .....	41
<i>Operator Eksekusi</i> .....	41
<i>Operator Penambahan Dan Pengurangan</i> .....	42
<i>Operator Logika</i> .....	43
<i>Operator String</i> .....	44
<i>Operator Array</i> .....	45
STRUKTUR KONTROL .....	46

<i>IF</i> .....	47
<i>IF ...ELSE</i> .....	48
<i>IF ...ELSEIF ... ELSE</i> .....	50
<i>Operator Kondisi Ternary</i> .....	52
<i>SWITCH</i> .....	53
<i>Struktur Kontrol Alternatif</i> .....	55
PERULANGAN.....	56
<i>WHILE</i> .....	56
<i>DO – WHILE</i> .....	58
<i>FOR</i> .....	60
<i>FOREACH</i> .....	62
<i>BREAK</i> .....	64
<i>CONTINUE</i> .....	65
<i>REQUIRE, INCLUDE, REQUIRE_ONCE, INCLUDE_ONCE</i> .....	66
<i>LATIHAN</i> .....	67
<b>BAB 3 FUNGSI-FUNGSI DI PHP</b> .....	<b>70</b>
FUNGSI.....	70
<i>Fungsi Buatan Sendiri</i> .....	71
<i>Argumen/Parameter Fungsi</i> .....	73
<i>Nilai Balik Fungsi</i> .....	75
<i>Fungsi Internal (Built-In)</i> .....	75
FUNGSI ARRAY.....	76
FUNGSI STRING.....	93
FUNGSI TANGGAL & WAKTU.....	100
LATIHAN.....	106
<b>BAB 4 FORM, SESSION &amp; COOKIES</b> .....	<b>108</b>
PENANGANAN FORM.....	108
<i>Komponen Form</i> .....	108
<i>Method GET</i> .....	110
<i>Method POST</i> .....	111
SESSION.....	111
COOKIES.....	113
LATIHAN.....	124
<b>BAB 5 FILE &amp; DIREKTORI</b> .....	<b>129</b>
FILE.....	129
DIREKTORI.....	130
LATIHAN.....	130
<b>BAB 6 OBJECT ORIENTED PROGRAMMING DI PHP</b> .....	<b>136</b>

CLASS & OBJECT .....	136
TURUNAN CLASS .....	138
LATIHAN.....	138
<b>BAB 7 PHP &amp; DATABASE .....</b>	<b>146</b>
APLIKASI PHPMYADMIN .....	146
DATABASE MYSQL .....	146
KONEKSI DATABASE .....	146
APLIKASI CRUD.....	146
<b>REFERENSI.....</b>	<b>157</b>

# Bab 1

## WEB DINAMIS

---

### Bab ini membahas:

- Web Dinamis
  - Web Server
  - Server Side Scripting
  - Instalasi Web Server & Server Side Scripting
- 

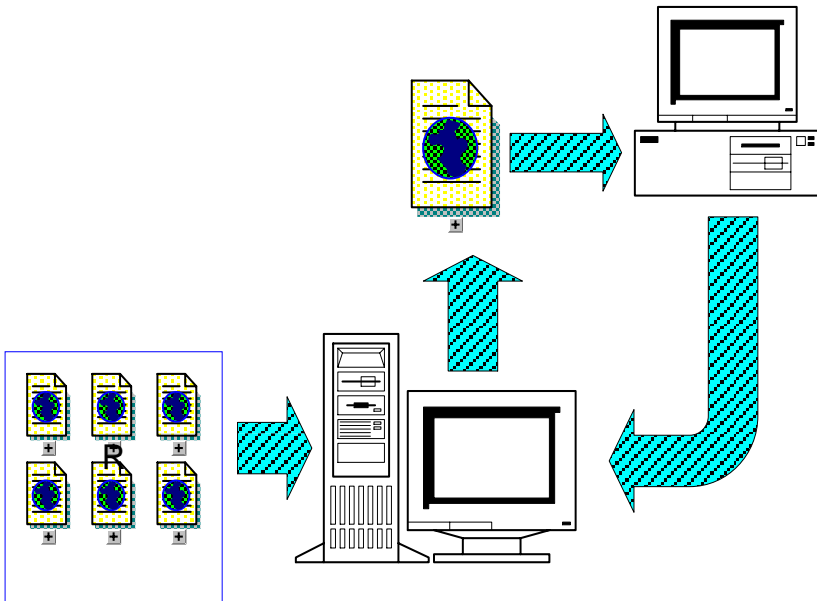
## Web Dinamis

Website Dinamis (*Dynamic Website*) adalah jenis halaman web yang disusun oleh konten dan layout yang kaya akan informasi didalamnya. Dinamakan website Dinamis karena kontennya dapat berubah-ubah. Dengan kata lain, adanya program yang berjalan untuk mengatur perubahan data yang ditampilkan dalam website Dinamis tersebut. Halaman web yang dibuat dengan menggunakan bahasa server seperti *PHP*, *Perl*, *ASP*, *ASP.NET*, *JSP*, *ColdFusion* dan bahasa yang lainnya. Jenis website Dinamis ini sangat cocok untuk website E-Commerce yang membutuhkan update data secara terus menerus. Dalam pemeliharaan website Dinamis pun lebih mudah daripada Website Statis karena dapat menggunakan *Content Management System (CMS)*.

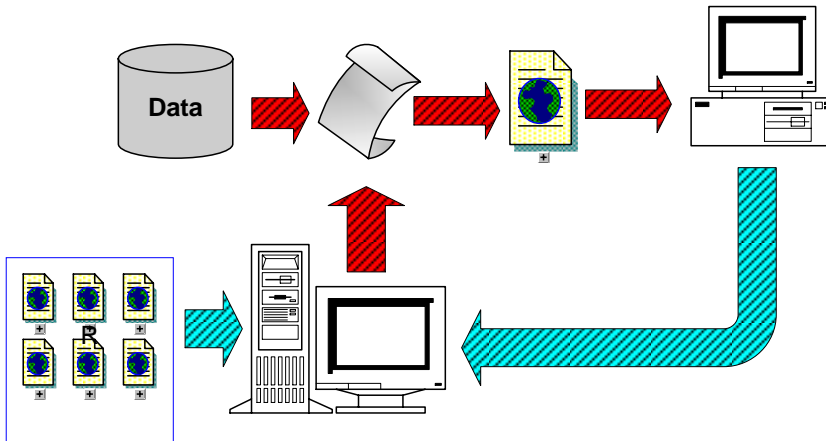
Kemudian, website Statis (*Static Website*) adalah sebuah website yang kontennya statis / tidak berubah-ubah. Sekali dibuat dan online di Internet, pada umumnya website tersebut tidak dapat diubah kecuali diubah secara manual melalui pengubahan bahasa pemrograman website tersebut. Oleh karena itu, terjadinya interaksi pun jarang sekali, sehingga dapat dikatakan seperti brosur online karena informasi yang diberikan juga terbatas.

## Web Server

Web Server merupakan sebuah perangkat lunak dalam server yang berfungsi menerima permintaan (request) berupa halaman web melalui HTTP atau HTTPS dari klien yang dikenal dengan browser web dan mengirimkan kembali (response) hasilnya dalam bentuk halaman halaman web yang umumnya berbentuk dokumen HTML.



Gambar 1.1 : Standart Web Arsitektur



Gambar 1.2 : Dynamic Web Arsitektur

Beberapa Web Server yang banyak digunakan di internet antara lain :

1. Apache Web Server (<https://httpd.apache.org/>)
2. Internet Information Service, IIS (<https://www.iis.net/>)
3. Nginx Web Server (<https://www.nginx.com/>)
4. LiteSpeed Web Server  
(<https://www.litespeedtech.com/products/litespeed-web-server>)

## Server Side Scripting

Server Side Scripting merupakan sebuah teknologi scripting atau pemrograman web dimana script (program) dikompilasi atau diterjemahkan di server. Dengan SSS, memungkinkan untuk menghasilkan halaman web yang dinamis.

Beberapa contoh Server Side Scripting (Programming) :

1. ASP (Active Server Page) dan ASP.NET
2. ColdFusion (<https://coldfusion.adobe.com/>)
3. Java Server Pages

(<https://www.oracle.com/technetwork/java/index-jsp-138231.html>)

4. Perl (<https://www.perl.org/>)
5. Python (<https://www.python.org/>)
6. PHP (<https://php.net/>)

Keistimewaan PHP :

1. Cepat
2. Free
3. Mudah dipelajari
4. Multi-platform
5. Dukungan technical-support
6. Banyaknya komunitas PHP
7. Aman

## Instalasi Web Server & Server Side Scripting

PHP tidak bisa bekerja dan digunakan secara mandiri, tetapi masih memerlukan beberapa perangkat lunak tambahan yang harus diinstall. Adapun perangkat lunak pendukung yang harus diinstal antara lain:

- Apache Server versi 2.4.38

Merupakan web server yang digunakan oleh PHP, berfungsi untuk menampilkan hasil dari proses script PHP ke kompter browser dalam bentuk tag HTML.

- PHP versi 7.3.2

Berfungsi sebagai mesin penterjemah saat halaman HTML yang mengandung script PHP dikirim ke server.

- MySQL versi 5



Merupakan database server yang paling sering digunakan dalam pemrograman PHP. Berfungsi untuk menyimpan data dalam database, serta memanipulasi data-data yang diperlukan.

- PHPMyAdmin

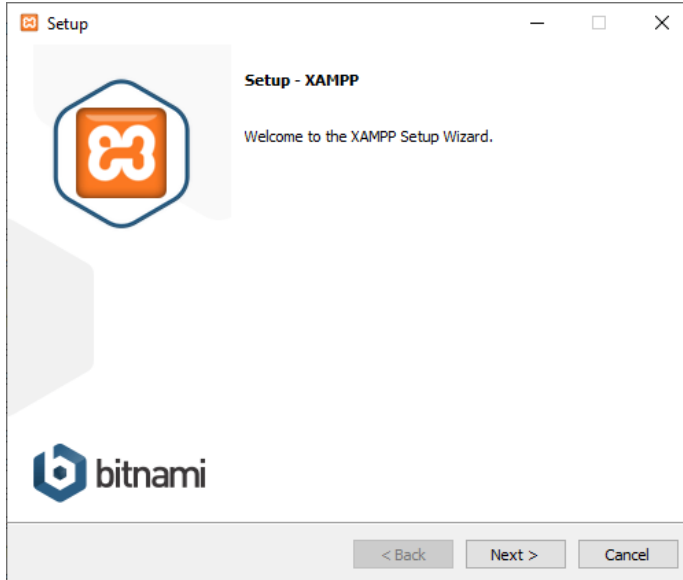
Adalah kakas untuk pengelolaan database yang berbasis web. PHPMyAdmin bukan merupakan suatu keharusan, bisa juga manipulasi data digantikan dengan kakas yang lain misalnya MySQL Console (berbasis teks). Dengan PHPMyAdmin pengelolaan atau manipulasi database menjadi lebih mudah.

Setiap platform yang berbeda, memiliki distribusi perangkat lunak pendukung yang berbeda pula. Misalnya, untuk platform Windows, maka Apache, PHP, dan MySQL yang akan diinstall harus dipilih dari distribusi perangkat lunak yang mendukung dengan platform Windows.

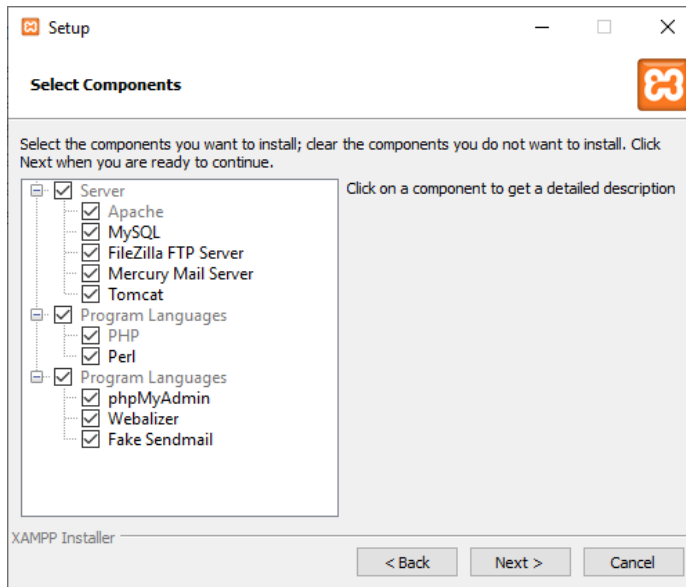
Begitu pula dengan platform LINUX, paket perangkat lunak yang diinstall harus sesuai dengan jenis platform LINUX. Perangkat lunak Apache, PHP, MySQL, dan PHPMyAdmin bisa didownload secara gratis di situs resminya masing-masing. Dalam situs resminya telah disediakan distribusi perangkat lunak untuk semua platform misalnya Windows, OS X, LINUX, Mac Apple, UNIX dan lain-lain, dari situ bisa didownload perangkat lunak yang sesuai dengan system operasi yang kita gunakan, termasuk dokumentasi bagaimana tata cara instalasinya.

Penulis berkeyakinan bahwa pembaca sebagian besar menggunakan sistem operasi windows dalam pengoperasiannya, oleh karena itu perangkat lunak yang digunakan dalam buku ini hanya yang mendukung dengan sistem operasi Windows saja.

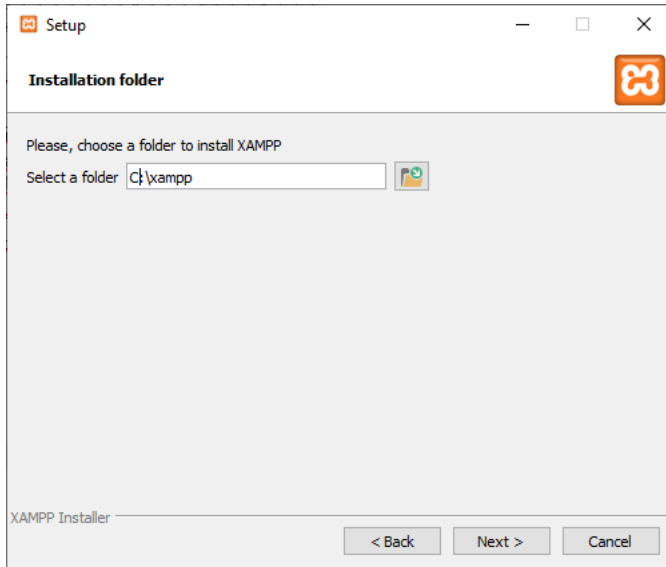
Pada modul ini akan menggunakan XAMPP untuk latihan instalasi web server, MySQL dan PHP.



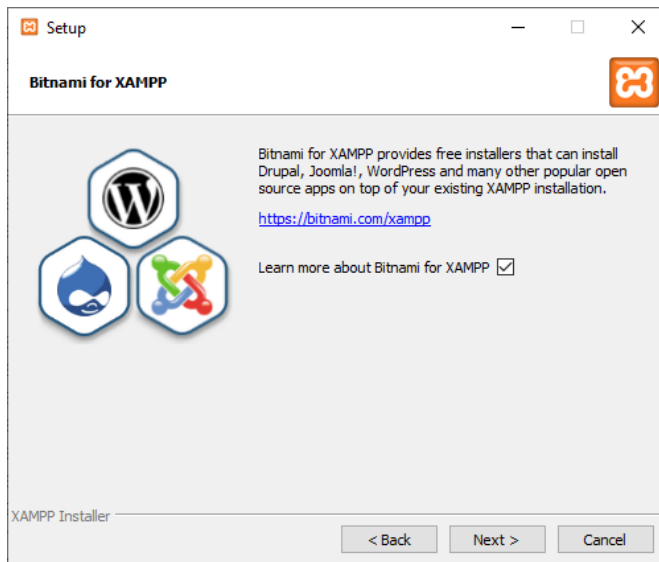
Gambar 1.1 : XAMPP Setup Wizard



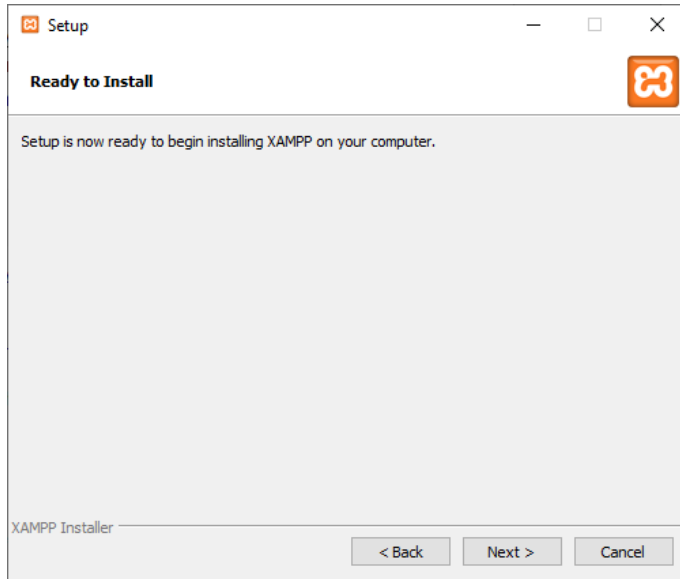
Gambar 1.2 : Component yang di install



Gambar 1.3 : Lokasi Instalasi XAMPP

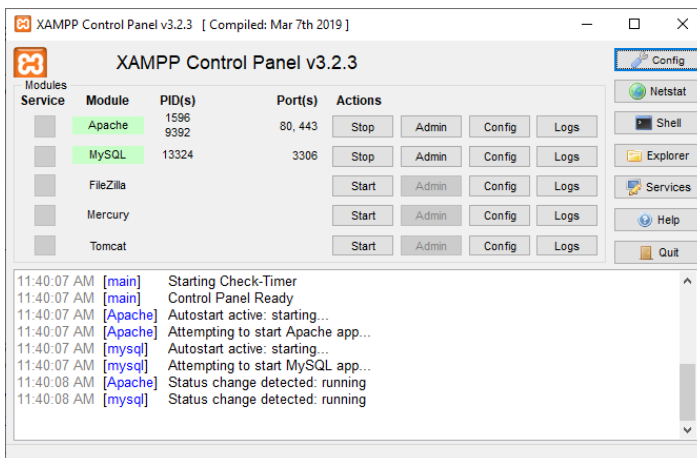


Gambar 1.4 : Bitnami for XAMPP



Gambar 1.5 : Ready to install

Tunggu proses instalasi sampai selesai dan klik tombol *finish*. Jalankan XAMPP control panel untuk pengecekan komponen yang sedang berjalan atau belum berjalan.

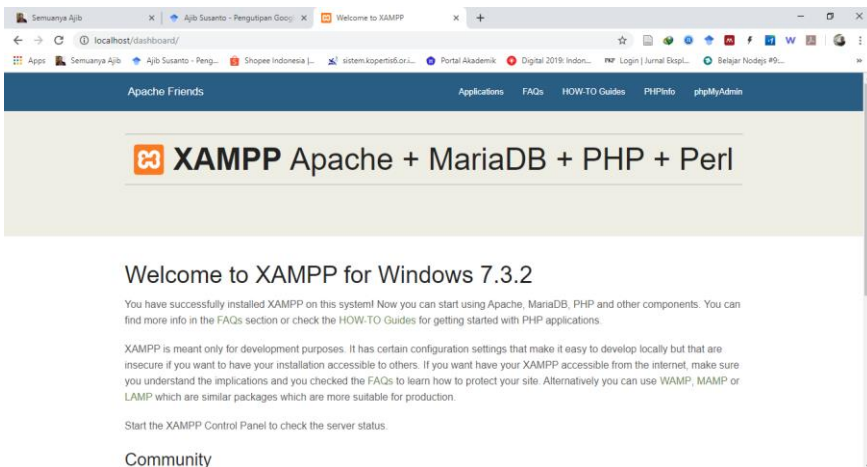


Gambar 1.6 : XAMPP Control Panel

Untuk memastikan bahwa XAMPP beserta PHP, Apache dan MySQL berjalan dengan baik, bukalah browser dan ketikkan URL sebagai berikut:

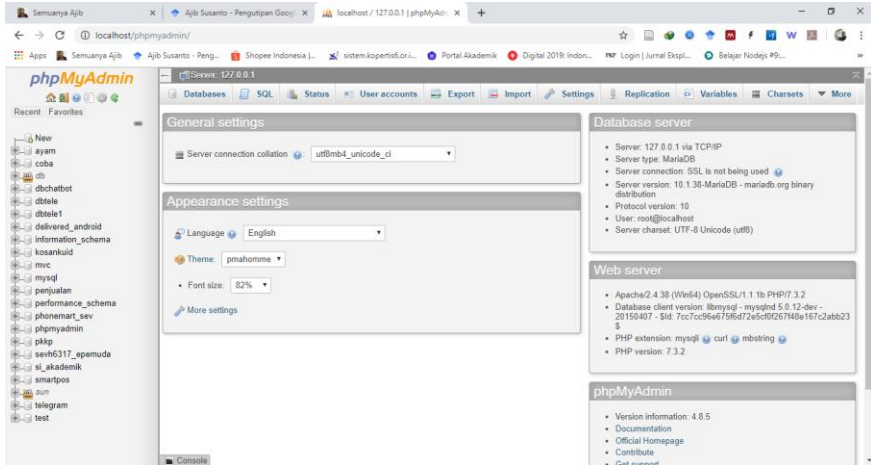
**http://localhost**

Jika semua sudah berjalan dengan baik, maka akan ditampilkan gambar sebagai berikut:



Gambar 1.7 : Web Server berjalan

Untuk mengecek akses ke database MySQL dapat diklik tab *phpMyAdmin*, maka akan ditampilkan halaman seperti berikut :

Gambar 1.8 : Halaman *phpMyAdmin*

# Bab 2

## DASAR PEMROGRAMAN PHP

---

**Bab ini membahas:**

- PHP
  - Sintak Dasar
  - Type Data
  - Variabel & Konstanta
  - Ekspresi
  - Operator
  - Struktur Kontrol
  - Perulangan
- 

## PHP

PHP merupakan singkatan dari Hypertext Preprocessor, tergolong sebagai perangkat lunak open source yang diatur dalam aturan general purpose licences (GPL). Pemrograman PHP sangat cocok untuk dikembangkan dalam lingkungan web, karena PHP bias dilekatkan pada script HTML atau sebaliknya. PHP dikhususkan untuk pengembangan web dinamis, maksudnya adalah bahwa PHP mampu menghasilkan website yang secara terus-menerus bisa berubah-ubah hasilnya sesuai dengan pola yang diberikan, hal tersebut tergantung dari permintaan client browser-nya (bias menggunakan browser opera, internet explorer, mozilla, dan lain lain). Dan biasanya pembuatan web dinamis dengan PHP berhubungan erat dengan database sebagai sumber data yang akan ditampilkan.

PHP tergolong juga sebagai bahasa pemrograman yang berbasis server (*server side scripting*), ini berarti bahwa semua *script PHP* diletakkan di server dan diterjemahkan oleh web server terlebih dahulu, kemudian hasil terjemahan di kirim ke browser client. Tentu hal tersebut berbeda dengan JavaScript, dimana kode program javascript harus didownload terlebih dahulu di computer client, selanjutnya diterjemahkan oleh browser internet. Oleh karena itu kode program JavaScript selalu nampak di halaman web bersangkutan. Secara teknologi Bahasa pemrograman PHP memiliki kesamaan dengan bahasa ASP (Active Server Page), Cold Fusion, JSP (Java Server Page) ataupun Perl.

## Sintak Dasar

Ketika PHP menerima suatu file, maka PHP akan mencari tags pembuka (“<?php”) dan tags penutup (“?>”), ini juga berarti sebagai pemberitahuan agar PHP mulai menterjemahkan baris-baris kode PHP tersebut, serta mengakhiri penterjemahannya sampai bertemu dengan tags penutup. Baris baris kode yang berada diluar pasangan penanda “<?php ... ?>” akan diabaikan (tidak diterjemahkan) oleh PHP. Sebagian besar baris-baris perintah PHP disisipkan dalam tags-tags HTML atau berlaku sebaliknya, contoh sederhana adalah sebagai berikut :

```
<p>Hai ini diabaikan.</p>
<?php echo 'Yang ini akan diparse atau
diterjemahkan.';?>
<p>Hai ini akan diabaikan juga.</p>
```

Ada empat jenis pasangan tags pembuka dan penutup berbeda yang bias digunakan oleh PHP. Antara lain :

**<?php...?>** dan **<script language="php">...</script>**,

kedua bentuk tags tersebut telah tersedia dalam PHP dan bisa langsung digunakan.

*Short tags* (<? . . . ?>) dan *ASP style tags* (<% . . . %>)



Kedua bentuk tags tersebut dapat diaktifkan atau dimatikan penggunaannya, tergantung pada pengaturan PHP di file *php.ini*. Dua jenis tags terakhir tidak disarankan penggunaannya karena beberapa server tidak mengenalinya. Contoh penggunaan tags pembuka dan penutup dalam PHP yang disarankan :

```
<?php
    echo 'Tulis seperti ini, jika menangani XHTML
    atau XML'; ?>
<script language="php">
    echo 'Beberapa editor (seperti FrontPage) tidak
    bisa memproses instruksi ini';
</script>
```

Contoh penggunaan tags pembuka dan penutup dalam PHP yang tidak disarankan :

```
<?
    echo 'ini paling sederhana, untuk memroses
    instruksi SGML';
?>
<?= expression ?> Ini merupakan cara pintas untuk
"<? echo expression?>"
<% echo 'Anda bisa menggunakan ASP-style tags'; %>
<%= $variable; # Ini merupakan cara pintas untuk
"<% echo . . ." %>
```

## Pemisahan Instruksi

Seperti bahasa C atau Perl, PHP membutuhkan penghentian baris pernyataan, serta memisahkan antara baris satu dengan baris lain dengan cara memberi tanda titik koma (“;”) diakhir setiap baris pernyataan. Kode tag penutup (“?>”) dari sebuah blok PHP secara otomatis akan berimplikasi sebagai titik koma (“;”), sehingga tidak diperlukan lagi adanya titik koma penghenti perintah di akhir baris suatu blok PHP.

```
<?php echo 'Ini hanya sebuah test'; ?>
<?php echo ' Ini hanya sebuah test ' ?>
<?php echo 'Kita mengabaikan tag penutup akhir';
```

## Komentar / Remarks

PHP mendukung pemberian komentar seperti yang ada di 'C', 'C++' dan Unix shell-style (Perl style). Terkadang dibutuhkan beberapa baris kalimat untuk memberi keterangan pada suatu baris program, hal ini sering disebut remarks. Dimana remarks tidak akan ikut dieksekusi oleh server. Jika remarks hanya satu baris maka gunakanlah (“//”) atau (“#”), jika remarks terdiri dari beberapa baris secara berurutan lebih efektif jika menggunakan (“/\* . . . \*/”) Lihat contoh berikut ini :

```
<?php
echo 'Ini hanyalah test';
// Ini komentar satu baris, model komentar C++
/* Ini komentar untuk baris lebih dari satu.
Dan diakhiri dengan tanda */
echo 'Ini masih test yang lain';
echo 'Satu lagi test terakhir';
# Komentar satu baris, model shell-style
?>
<?php
<h1>Ini hanya sebuah <?php # echo 'sederhana';?>
contoh.</h1>
<p>Header di atas akan tercetak 'Ini hanya sebuah
contoh'.</p>
?>
```

### Contoh yang salah:

```
<?php
/*
echo 'Ini hanya sebuah test'; /* Komentar ini
menyebabkan kesalahan
*/
*/
?>
```

---

## Type Data

PHP tidak memerlukan pendeklarasian tipe data suatu variable secara eksplisit, tetapi lebih ditentukan oleh *runtime program* PHP, tergantung dari konteks bagaimana variable tersebut digunakan. Untuk mengetahui jenis tipe data suatu variable saat dilakukan penelusuran (*debugging*) program gunakan fungsi *gettype()*. Sedangkan untuk memeriksa kebenaran suatu tipe data apakah sesuai dengan yang dimaksud, maka bisa gunakan fungsi *is\_type*.

```
<?php
$a_bool   = TRUE;           // type Boolean
$a_str    = "foo";         // type string
$a_str2   = 'foo';         // type string
$a_int    = 12;            // type integer
$a_float  = 3.17;
echo gettype($a_bool); // akan tercetak: Boolean
echo gettype($a_str);  // akan tercetak: string
// Jika type data integer,
// maka akan ditambah dengan empat.
if (is_int($a_int)) {
    $a_int += 4;
}
// Jika $a_bool adalah type string,
// maka akan dicetak.
// (Maka tidak mencetak apapun,
// karena $a_bool adalah boolean).
if (is_string($a_bool)) {
    echo "String: $a_bool";
}
?>
```

PHP mendukung delapan jenis tipe data, antara lain:

- Tipe *scalar*, tipe scalar tidak bisa dipecah lagi menjadi bagian yang lebih kecil, boleh dikatakan merupakan tipe dasar.
  - Boolean
  - Integer
  - Float (floating point, ‘double’)

- String
- Tipe *compound* / Tipe campuran
  - Array
  - Object (tidak dibahas di buku ini)
- Tipe *special* / Tipe khusus
  - Resource (tidak dibahas di buku ini)
  - Null

## Tipe Integer

Integer adalah sederet angka dimana dituliskan sebagai set  $Z = \{\dots, -2, -1, 0, 1, 2, \dots\}$ . Penulisan bilangan integer bisa dalam notasi decimal (10-based), hexadecimal (16 based) atau octal (8-based), termasuk penanda (- atau +).

Jika digunakan notasi octal maka penulisan harus didahului dengan angka 0 (nol) dan jika digunakan notasi hexadecimal maka penulisan harus didahului dengan angka 0x. Penulisan yang benar adalah sebagai berikut :

```
<?php
$a = 1234; # angka desimal positif
$a = -123; # angka desimal negative
$a = 0123; # angka oktal (sama dengan nilai 83 desimal)
$a = 0x1A; # angka hexadecimal (sama dengan 26 desimal)
?>
```

Besarnya ukuran integer bergantung pada platformnya, meskipun maksimum nilai integer berkisar dua milyar (32 bits signed/positif). Besar ukuran maksimal integer ditentukan dari *PHP\_INT\_SIZE* (melalui pengaturan di file *php.ini*) , dan maksimal nilainya ditentukan dari pengaturan *PHP\_INT\_MAX*.

## Tipe Pecahan / Floating Point

Jenis floating point merupakan bilangan pecahan dengan presisi tinggi, dan yang termasuk dalam floating point antara lain *float*, *double*, dan

*real*. Besar ukuran tipe data float bergantung pada platform yang digunakan, secara umum adalah  $\sim 1.8 \times 10^308$  atau  $1.8 \times 10308$  dengan tingkat presisi 14 digit desimal, ini mengacu pada standar format 64 bit IEEE.

Contoh penulisannya adalah sebagai berikut :

```
<?php
$a = 1.234;
$a = 1.2e3;
$a = 7E-10;
?>
```

Terkadang tipe floating point menghasilkan nilai yang tidak sesuai dengan yang diharapkan, misalnya saja:

```
<?php
    echo floor((0.1+0.7)*10); // Tercetak: 7
?>
```

Pembulatan dengan fungsi *floor()* di atas seharusnya menghasilkan nilai 8, tetapi yang tercetak adalah nilai 7. Ini disebabkan nilai pecahan dalam notasi decimal seperti 0.1 atau 0.7 tidak bisa dikonversikan dan dicarikan padanannya tanpa harus kehilangan ketelitiannya.

Contoh lain: misalnya  $1/3$  dalam decimal akan menghasilkan nilai 0.3333333..., sehingga sangat disarankan agar tidak menggunakan angka floating point sebagai perbandingan persamaan.

## Tipe String

String boleh dikatakan sebagai serangkaian character. Dimana besarnya character sama dengan byte. Ada tiga cara penulisan string, antara lain:

- **Single quoted / petik tunggal**

Penulisan string dengan *single quoted* harus diawali dan diakhiri petik tunggal (karakter `'`). Cara ini sedikit membingungkan jika ada karakter karakter khusus (*escaped character*) yang disertakan. Misalnya, jika ada karakter petik tunggal yang akan disisipkan maka sebelum karakter petik tunggal harus didahului dengan *backslash* (`\`)

```
<?php
echo 'Ini contoh string sederhana';
echo 'Anda juga bisa menambahkan baris baru dalam
strings, seperti cara berikut.';
echo 'Hari Jum\'at aku akan datang di kota Malang';
// output: ... "Hari Jum'at aku akan ..."
?>
```

- **Double quoted / petik ganda**

Penulisan string dengan *doubled quoted* harus diawali dan diakhiri petik ganda (karakter "). Pada dasarnya cara penulisan double quoted hampir sama dengan single quoted, tetapi double quoted lebih fleksibel dan memiliki lebih banyak *escaped character* yang bisa disisipkan.

Tabel 2.1 : Escaped Character

Urutan Escape	Pengertian
\n	linefeed (LF or 0x0A (10) dalam ASCII)
\r	carriage return (CR or 0x0D (13) dalam ASCII)
\t	horizontal tab (HT or 0x09 (9) dalam ASCII)
\\	Backslash
\\$	dollar sign
\"	Petik ganda ( <i>double-quote</i> )
\[0-7]{1,3}	Urutan character yang sesuai dengan regular ekspresion, dalam notasi oktal.
\x[0-9A-Fa-f]{1,2} atau \x00 s/d \xFF	Urutan character yang sesuai dengan regular ekspresion, dalam notasi hexadecimal.

Jika kita menuliskan karakter khusus (*escaped character*) selain yang terdaftar dalam tabel di atas, maka akan menyebabkan karakter backslash (\) akan ikut tercetak.

```
<?php
$minuman = 'Juice';
echo "$minuman's rasanya enak";
```

---

```
// Bisa bekerja, tetapi nilai variable tidak tampil,
// karena karakter (') pada $minuman's salah jika digunakan
// sebagai nama variabel.
```

```
echo "Dia minum segelas $minumans";
```

```
// Salah, karena 's' pada variable $minumans bukan nama
// variable yang sesungguhnya.
```

```
echo "Dia minum segelas ${minuman}s ";
```

```
// Benar
```

```
?>
```

### • Sintaks Heredoc

Cara lain untuk memenggal penulisan string kemudian disambung ke baris berikutnya adalah dengan menggunakan Sintaks *Heredoc* ("`<<<<`").

Adapun struktur heredoc adalah sebagai berikut:

```
<<<<nama_heredoc
string
nama_heredoc;
```

Aturan penulisan Heredoc Sintaks adalah sebagai berikut :

- Baris pertama diawali dengan tanda "`<<<<nama_heredoc`".
- *String*, berada di baris berikutnya.
- Baris terakhir diakhiri dengan "`nama_heredoc;`".

Contoh yang benar penggunaan Sintaks heredoc :

```
<?php
$var1="penulisan";
$var2="kesalahan";
$str = <<<<EOD
Heredoc sesuai untuk
$var1 string yang Panjang
tanpa mengalami $var2
EOD;
echo $str;
?>
```

## Variabel & Konstanta

Variable mutlak diperlukan dalam pemrograman PHP, karena berfungsi sebagai tempat untuk menampung suatu nilai data, baik berupa masukan (*input*) maupun keluaran (*output*).

Untuk membuat suatu variabel tidak diperlukan deklarasi awal seperti pemrograman berbasis *compiler* seperti Delphi atau Visual Basic, cukup dengan memberikan character dolar (\$) dan disambung dengan nama varibale (misalnya: *\$nama\_variable*), maka varibale sudah bisa digunakan. Nama variable bersifat *case-sensiteve*.

### Penamaan Variable

Pada dasarnya programmer diberi kebebasan dalam pemberian nama variable, namun ada sedikit aturan yang membatasinya. Yang diperbolehkan dalam penamaan variable, antara lain:

- Nama variable bisa terdiri dari huruf abjad, angka, dan underscore (`_`)
- Nama variable bisa diawali dengan tanda underscore (`_`)

Yang tidak diperbolehkan dalam penamaan variable, antara lain:

- Nama variable tidak boleh diawali dengan angka
- Nama variable tidak boleh mengandung operator aritmatika
- Nama variable tidak boleh mengandung karakter khusus seperti `@ ; # ! & . :`
- Nama variable tidak boleh mengandung spasi

```
<?php
```

```
$var = 'Ghiyatsi';  
$Var = 'Najwa';  
echo "$var, $Var"; // output "Ghiyatsi, Najwa"  
$4angka = 'Variable yang salah';  
// invalid; karena diawali dengan angka.  
$_4angka = 'Variable yang benar';
```



---

```
// valid; karena diawali dengan garis bawah
(underscore)
$ätäyte = 'mansikka';
// valid; 'ä' merupakan perluasan dari ASCII 228.
?>
```

## Predefine Variable

PHP juga memiliki sekumpulan *predefined variable* tambahan yang bisa berasal dari web server, environment, atau input user (berasal dari *form*). Kumpulan predefined tersebut memiliki sifat khusus sesuai dengan peruntukannya yang juga bersifat global, karena secara otomatis memiliki jangkuan/scope yang luas. Hal ini sering disebut juga dengan *superglobal*.

### PHP SUPERGLOBAL:

#### ***\$GLOBALS***

Merupakan rujukan bagi variable yang memiliki lingkup global disemua script. \$GLOBALS mulai tersedia di PHP mulai versi 3 ke atas.

#### ***\$\_SERVER***

Merupakan sekumpulan variable yang dihasilkan oleh web server. Sehingga semua variable yang berhubungan dengan penanganan server, misalnya untuk mengetahui alamat “*IP host*” dan “*nama host*” yang terhubung dengan server atau fungsi-fungsi lainnya yang berhubungan dengan server ditangani oleh \$\_SERVER. Analogi \$\_SERVER untuk PHP versi sebelumnya adalah \$HTTP\_SERVER\_VARS.

#### ***\$\_GET***

Merupakan variable yang dihasilkan oleh query string URL atau melalui HTTP GET. \$\_GET sangat berhubungan dengan penerimaan data yang berasal dari halaman website diluar PHP, terutama variable yang berasal dari FORM HTML yang menggunakan method GET.

Analogi `$_GET` untuk PHP versi sebelumnya adalah `$HTTP_GET_VARS`. Pernyataan GET akan dibahas lebih lanjut pada bab “**PENANGANAN FORM & SESSION**”, karena GET sangat penting untuk diketahui oleh pembaca.

### ***\$\_POST***

Merupakan variable yang dihasilkan melalui HTTP POST. `$_POST` sangat berhubungan dengan penerimaan data yang berasal dari halaman website diluar PHP, terutama variable yang berasal dari FORM HTML yang menggunakan method POST. Analogi `$_POST` untuk PHP versi sebelumnya adalah `$HTTP_POST_VARS`. POST akan dibahas lebih lanjut pada bab “**PENANGANAN FORM & SESSION**”, karena POST sangat penting untuk diketahui oleh pembaca.

### ***\$\_COOKIE***

Merupakan variable yang dihasilkan melalui HTTP COOKIE. Cookie bisa dianggap sebagai variable tampungan berisi data yang ditempatkan pada komputer client. `$_COOKIE` bisa digunakan dengan syarat layanan cookie pada browser client diaktifkan. Analogi `$_COOKIE` untuk PHP versi sebelumnya adalah `$HTTP_COOKIE_VARS`.

### ***\$\_FILES***

Merupakan variable yang dihasilkan melalui HTTP POST UPLOAD FILE, `$_FILES` berguna untuk meng-upload file dari komputer client menuju ke komputer server dengan menggunakan FORM HTML. Analogi `$_FILES` untuk PHP versi sebelumnya adalah `$HTTP_POST_FILES`.

### ***\$\_ENV***

Merupakan variable yang dihasilkan melalui environment. Analogi `$_ENV` untuk PHP versi lebih lama adalah `$HTTP_ENV_VARS`.

### ***\$\_REQUEST***

Merupakan variable yang dihasilkan melalui mekanisme input GET, POST dan COOKIE (bisa menerima variable dari form dengan method

POST atau GET), tetapi cara ini kurang bisa dipercaya kebenaran hasil yang diperoleh, sehingga kurang disarankan penggunaannya.

### ***\$\_SESSION***

Merupakan variable yang telah diregister-kan (disimpan dalam file sementara). `$_SESSION` memiliki fungsi yang hampir sama dengan `$_COOKIE`, perbedaanya hanya pada masalah penempatan variable tampungannya. Session meletakkan variable tampungannya di server, sedangkan cookie menempatkan variable tampungannya di client. Cara ini jauh lebih aman daripada menggunakan cookie. Analogi `$_SESSION` untuk PHP versi sebelumnya adalah `$HTTP_SESSION_VARS`. Session akan dibahas lebih lanjut dalam bab “**PENANGANAN FORM & SESSION**”, karena session sangat penting untuk diketahui oleh pembaca.

## **Lingkup Variable (Scope)**

Lingkup atau scope variable sangat berhubungan dengan seberapa lama umur hidup suatu variable, serta sejauh mana variable tersebut bisa dikenali pada setiap baris-baris kode PHP. Lingkup variable lebih ditentukan saat dimana variable tersebut didefinisikan.

```
<?php
$a = 1;
include 'b.inc';
?>
```

Di sini variable `$a` juga bisa dikenali dalam script `b.inc`. Tetapi untuk fungsi buatan sendiri variable `$a` tidak bisa dikenali. Diperlukan adanya variable dengan scope local dalam tubuh fungsi buatan sendiri.

```
<?php
$a = 1; /* scope global*/
function Test()
{
    echo $a; /* merujuk pada variable lokal
    fungsi */
```

```
}  
Test ();  
?>
```

Pernyataan `echo $a` merujuk ke variable lokal fungsi, karena tidak ditemukan, maka script di atas tidak menghasilkan apa-apa. Ini sedikit berbeda dengan pemrograman bahasa C, dimana variable global dalam bahasa C secara otomatis juga bisa dikenali disetiap fungsi, kecuali jika ada nama variable yang sama, maka variable lokal dalam fungsi akan mengabaikan (*overriden*) variable global tersebut.

## Keyword Global

Agar sebuah variable memiliki lingkup global dalam suatu fungsi, maka diperlukan pendeklarasian ulang terhadap sebuah variable dengan cara menggunakan keyword ***global***.

```
<?php  
$a = 1;  
$b = 2;  
function Tambah()  
{  
global $a, $b;  
$b = $a + $b;  
}  
Tambah();  
echo $b;  
?>
```

Script di atas akan menghasilkan nilai “3”. Dengan pendeklarasian variable \$a dan \$b sebagai global dalam tubuh fungsi, maka variable tersebut akan merujuk ke variable global di atasnya. Tidak ada Batasan berapa banyak variable global yang bisa dimanipulasi oleh fungsi. Cara kedua untuk mengakses variable dari scope global adalah dengan mendefinisikannya sebagai array \$GLOBALS. Maka contoh sebelumnya bisa dituliskan dengan cara berikut ini:

```
<?php
```

```
$a = 1;
$b = 2;
function Tambah()
{
    $GLOBALS['b'] = $GLOBALS['a'] + $GLOBALS['b'];
}
Tambah();
echo $b;
?>
```

## Keyword Static

Variable *static* hanya bisa dikenal dalam lingkup lokal fungsi dan variable tersebut tidak akan kehilangan nilainya ketika eksekusi program meninggalkan scope.

```
<?php
function Test()
{
    $a = 0;
    echo $a;
    $a++;
}
?>
```

Fungsi di atas benar-benar tidak berguna, karena setiap kali fungsi *Test()* dipanggil, yang tercetak selalu nilai “0”. Pernyataan *\$a++* tidak berpengaruh apapun karena setiap kali eksekusi keluar dari tubuh fungsi maka nilai *\$a* juga ikut hilang. Agar jejak hasil penambahan yang dikenakan pada *\$a* selalu ada, maka variable *\$a* perlu dideklarasikan sebagai **static**.

```
<?php
function Test()
{
    static $a = 0;
    echo $a;
```

```
        $a++;  
    }  
?>
```

Sekarang, setiap kali fungsi Test() dipanggil maka yang tercetak adalah \$a dan penambahannya.

## Konstanta

Konstanta adalah pemberian nama baru terhadap suatu nilai. Dimana konstanta tersebut bersifat tetap dan tidak terpengaruh oleh eksekusi script apapun. Konstanta bersifat case-sensitive. Menurut kesepakatan bersama, penamaan konstanta selalu menggunakan huruf besar.

```
<?php  
// Penamaan konstanta yang benar  
define("KONS", "sesuatu");  
define("KONS2", "sesuatu yang lain");  
define("KONS_OK", "sesuatu yang lain lagi");  
?>
```

Cara mendefinisikan konstanta adalah dengan menggunakan fungsi *define()*. Sekali konstanta tersebut didefinisikan maka konstanta tersebut tidak bisa dirubah atau dihapus.

Konstanta hanya boleh berisi data scalar atau bertipe tunggal misalnya boolean, integer, float dan string (tidak berlaku terhadap array atau object). Konstanta juga tidak bias diisi dengan sesuatu yang berupa pernyataan atau ekspresi. Untuk mendapatkan nilai konstanta bisa dengan cara mencatumkan nama konstatanya. Tidak seperti variable, konstanta tidak memerlukan tanda "\$" didepan nama konstannya.

## Perbedaan Antara Konstanta Dan Variable:

- Konstanta tidak diberi tanda dollar (\$) sebelum nama konstannya.
- Konstanta hanya boleh didefinisikan dengan fungsi *define()*.

- Konstanta bisa didefinisikan dan diakses dimana saja tanpa mengikuti aturan scope.
- Konstanta tidak bisa didefinisikan ulang atau menghapusnya.
- Konstanta hanya boleh berisi nilai scalar.

```
<?php
define("CONSTANT", "Hello world.");
echo CONSTANT; // "Hello world."
?>
```

## Ekspresi

Ekspresi hampir pasti digunakan disetiap pemrograman, karena program disusun atas ekspresi-ekspresi tersebut. Ekspresi adalah sesuatu yang menghasilkan nilai dan kebanyakan berupa persamaan pemrograman. Bentuk dasar ekspresi terdiri atas konstanta dan variable. Ketika kita menuliskan “\$a = 5”, berarti kita memberi nilai 5 kepada variable \$a, atau dengan kata lain \$a merupakan ekspresi yang bernilai 5 (dalam kasus ini '5' merupakan konstanta integer). Setelah memberi nilai 5 pada variable \$a, maka jika kita menuliskan “\$a = \$b” sama artinya jika kita menuliskan \$b = 5.

```
<?php
function coba ()
{
    return 5;
}
?>
```

Anggap saja kita sudah mengerti konsep fungsi (selanjutnya lihat bab fungsi), jika kita menuliskan \$c = coba() sama artinya jika kita

menuliskan  $\$c = 5$ . Fungsi merupakan ekspresi yang memiliki nilai balik dari fungsi itu sendiri. Selama fungsi `coba()` mengembalikan nilai 5 maka ekspresi `coba()` juga bernilai 5.

Biasanya fungsi tidak hanya mengembalikan nilai balik saja, tetapi juga ada proses perhitungan di dalam tubuh fungsi terlebih dahulu. Untuk ekspresi, tidak perlu dibahas lebih lanjut, karena apa yang telah dan akan dilakukan selanjutnya selalu berkuat pada ekspresi-ekspresi.

Contoh ekspresi lain:

```
<?php
function double($i)
{
    return $i*2;
}
$b = $a = 5;
$c = $a++;
$e = $d = ++$b;
$f = double($d++);
$g = double(++$e);
$h = $g += 10;
?>
```

## Operator

Ada tiga jenis operator utama yang dikenal dalam PHP, antara lain:

- Operator **unary**, dimana operator mengoperasikan hanya satu nilai variable, contohnya operator lawan/negasi (“!”), operator penambahan (*increment*, “++”), dan operator pengurangan (*decrement*, “--”).
- Operator **binary**, terdiri dari beberapa operator yang mempertimbangkan urutan penyelesaian. Kelompok ini paling sering digunakan dalam PHP.



- Operator *ternary*, operator yang membandingkan dua pernyataan, dimana kebenarannya ditentukan oleh ekspresi ketiga. Operator memiliki beberapa urutan (hirarki) eksekusi, mana operator yang harus didahulukan dan mana operator yang dieksekusi selanjutnya. Urutan operator perlu diperhatikan jika ada ekspresi yang menggunakan lebih dari satu operator. Evaluasi urutan operator bisa dimulai dari kiri atau dari kanan, istilah ini disebut juga dengan *Associativity*.

Tabel 2.2 : Hirarki Operator

Associativity	Operator	Keterangan
non-associative	new	new
Kiri	[	array()
non-associative	++ --	penambahan/pengurangan
non-associative	~ - (int) (float) (string) (array) (object)	Type
	@	
non-associative	instanceof	Type
Kanan	!	logika
Kiri	* / %	aritmatika
Kiri	+ - .	aritmatika dan string
Kiri	<< >>	bitwise

Non-associative	< <= > >=	perbandingan
Non-associative	== != === !==	perbandingan
Kiri	&	bitwise dan referensi
Kiri	^	bitwise
Kiri		bitwise
Kiri	&&	logika
Kiri		logika
Kiri	? :	ternary
kanan	= += -= *= /= .= %= &=  = ^= <<= >>=	pemberian nilai
Kiri	and	logika
Kiri	xor	logika
Kiri	or	logika

Asosiatif kiri berarti ekspresi dievaluasi dari kiri ke kanan, dan sebaliknya asosiatif kanan berarti ekspresi dievaluasi dari kanan ke kiri.

```
<?php
```

```
$a = 3 * 3 % 5; // (3 * 3) % 5 = 4
$a = true ? 0 : true ? 1 : 2;
// (true ? 0 : true) ? 1 : 2 = 2
$a = 1;
$b = 2;
$a = $b += 3; // $a=($b+=3) -> $a=5, $b=5
?>
```

## Operator Aritmatika

Operasi aritmatika yang umum digunakan dalam pemrograman, antara lain:

Tabel 2.3 : Operasi Aritmatika

Operator	Keterangan
+	Tambah
-	Kurang
/	Bagi
*	Kali
%	Sisa bagi

Aturan penulisan aritmatika adalah **variable = ekspresi aritmatika**, ini berarti bahwa **variable** akan bernilai sesuai hasil yang diberikan dari **ekspresi aritmatika** yang dilakukan. Disisi sebelah kiri hanya boleh ada satu variable tunggal saja dan tidak boleh ada yang lain, sedangkan sebelah kanan bisa berupa ekspresi/rumus aritmatika, konstanta atau variable.

```
<?php
// Contoh penulisan operasi aritmatika yang
benar
$var_1 = 8;
$var_2 = $var_1;
echo $var_2;
$panjang = 10;
$lebar = 5;
$luas = $panjang * $lebar;
echo $luas;
// Contoh penulisan operasi aritmatika yang
salah
8 = $var_1;
4 + 7 = $var_2;
$panjang * $lebar = $luas;
?>
```

## Hirarki Operator Aritmatika

Seringkali ekspresi aritmatika memerlukan beberapa operator yang berbeda, sehingga kita harus tahu urutan penyelesaian dari masing-masing operator, agar nilai yang dihasilkan benar sesuai dengan yang diharapkan.

Tabel 2.4 : Hirarki Operator Aritmatika

Hirarki	Operator	Keterangan
1	* / % ()	Tergantung pada posisinya, urutan dari kiri didahulukan kemudian sebelah kanan.
2	+ atau -	Tergantung pada posisinya, urutan dari kiri didahulukan kemudian sebelah kanan.

```
<?php
$nilai1 = 12 + 4 * 3 / 6;
echo $nilai1;
// 12 + 12 / 6
// 12 + 2
// hasilnya 14
$nilai2 = 210 / ( 6 * ( 4 + 5 - 2 ) )
echo $nilai2;
// 210 / ( 6 * 7 )
// 210 / 42
// hasilnya 5
?>
```

## Operator Pemberi Nilai

Operator pemberi nilai diwakili oleh tanda sama dengan (“=”), yang berarti bahwa operan sebelah kiri akan diberi nilai dengan ekspresi disebelah kanan.

```
<?php
$a = ($b = 4) + 5;
// $a bernilai 9 dan $b bernilai 4.
?>
```

Operator pemberi nilai bisa juga dikembangkan lagi menjadi operator kombinasi, yang bertujuan untuk menyederhanakan sintaks. Tambahan operator tersebut berguna saat kita mencari nilai Total suatu nilai, dan metode yang digunakan adalah dengan cara perulangan (*looping*).

```
<?php
$total = 0;
while ($total < 10 )
{
    $total = $total + 1;
}
echo $total;
// akan sama artinya dengan berikut ini
$total = 0;
while ($total < 10 ){
    $total += 1;
}
echo $total;
?>
```

Operator kombinasi juga bisa dikenakan pada penggabungan tipe string

```
<?php
$a = 3;
$a += 5;
// $a bernilai 8, atau bisa dituliskan: $a = $a
+ 5;
$b = "Hello ";
$b .= "Najwa";
// $b bernilai "Hello Najwa", atau bisa
dituliskan:
$b = $b .
"Najwa";
?>
```

## Operator Bitwise

Operator bitwise adalah operator yang bertujuan untuk mengoperasikan bilangan biner (angka 0 dan 1). Jika bilangan operan berupa bilangan desimal maka harus dikonversikan terlebih dahulu menjadi bilangan biner, baru kemudian dioperasikan dengan operator bitwise. Jika operan kedua-duanya berupa string maka nilai operan harus dikonversikan terlebih dahulu menjadi nilai ASCII (sesuai table ASCII), baru kemudian dioperasikan dengan operator bitwise.

Tabel 2.5 : Operator Bitwise

Contoh	Nama	Hasil
$\$a \& \$b$	And	Bit di set 1 jika kedua-duanya yaitu $\$a$ dan $\$b$ bernilai 1.
$\$a   \$b$	Or	Bit di set 1 jika nilai salah satu dari $\$a$ atau $\$b$ bernilai 1.
$\$a \wedge \$b$	Xor	Bit di set 1 jika kedua nilai dari $\$a$ dan $\$b$ memiliki perbedaan.
$\sim \$a$	Not	Bit akan di set 1 jika $\$a$ bernilai 0, dan di set 0 jika $\$a$ bernilai 1.
$\$a \ll \$b$	$\$b$ digeser ke kiri	Menggeser bit $\$a$ sebanyak $\$b$ langkah ke kiri (disetiap langkah "dikalikan dengan dua")
$\$a \gg \$b$	$\$b$ digeser ke kanan	Menggeser bit $\$a$ sebanyak $\$b$ langkah ke kanan (disetiap langkah berarti "dibagi dengan dua")

```
<?php
```

```

$a = 8; // nilai dalam bit : 1000
$b = 9; // nilai dalam bit : 1001
$ab_and = $a & $b;
echo $ab_and; // output 8, dalam biner: 1000
$ab_or = $a | $b;
echo $ab_or; // output 9, dalam biner: 1001
$ab_xor = $a ^ $b;
```

```
echo $ab_xor; // output 1, dalam biner 0001
atau 1
$ab_not = ~$a;
echo $ab_not; // output 7, dalam biner 0111
atau 111
?>
<?php
echo "12" ^ "9";
// Output adalah Character Backspace (ascii 8)
// ('1' (ascii 49)) ^ ('9' (ascii 57)) = #8
echo "hallo" ^ "hello";
// Output adalah ascii #0 #4 #0 #0 #0
// 'a' ^ 'e' = #4
?>
```

## Operator Perbandingan

Operator perbandingan adalah untuk membandingkan dua nilai. Perlu diperhatikan bahwa operator “==” tidaklah sama dengan operator “=”, jika operator “==” merupakan operator perbandingan dan akan menghasilkan nilai benar atau salah, sedangkan operator “=” merupakan operator pemberi nilai. Jika tidak berhati-hati dalam penggunaannya, maka akan menyebabkan kesalahan program yang cukup sulit untuk dideteksi, karena secara sintaks PHP menganggap hal tersebut sudah benar, namun secara logika akan menghasilkan nilai yang salah.

Jika membandingkan antara integer dengan string, maka string akan dikonversi ke angka terlebih dahulu. (lihat tabel 2.7)

Jika membandingkan dua angka string, maka keduanya-duanya akan dikonversikan menjadi integer. (lihat tabel 2.7)

Tabel 2.6 : Operator Perbandingan

Contoh	Nama	Hasil
<code>\$a == \$b</code>	Sama Dengan	Benar jika \$a sama dengan \$b
<code>\$ === \$</code>	Identik	Benar jika \$ a sama dengan \$b, dan keduanya memiliki kesamaan type
<code>\$ != \$b</code>	Tidak Sama	Benar jika \$a tidak sama dengan \$b
<code>\$ &lt;&gt; \$b</code>	Tidak Sama	Benar jika \$a tidak sama dengan \$b
<code>\$a !==</code>	Tidak Identik	Benar jika \$a tidak sama dengan \$b, dan keduanya tidak memiliki kesamaan type
<code>\$a &lt; \$b</code>	Kurang Dari	Benar jika \$a kurang dari \$b
<code>\$a &gt; \$b</code>	Lebih Dari	Benar Jika \$a lebih besar dari \$b
<code>\$a &lt;= \$b</code>	Kurang Dari Atau Sama Dengan	Benar jika \$a kurang dari atau sama dengan \$b
<code>\$a &gt;= \$b</code>	Lebih Dari Atau Sama Dengan	Benar jika \$a lebih besar atau sama dengan \$b

Tabel 2.7 : Perbandingan dengan tipe berbeda

Tipe Operan 1	Tipe Operan 2	Hasil
null atau string	string	Konversi dari NULL ke "", perbandingan numerik atau lexical
bool atau null	anything	Konversi ke bool, FALSE < TRUE
string, resource atau number	string, resource atau number	Menterjemahkan string dan resource ke angka
Array	array	Array dengan anggota yang terkecil, jika key dari operan 1 tidak ditemukan dalam operan 2 maka array tidak bisa dibandingkan
Array	anything	array selalu lebih tinggi
object	anything	Object selalu lebih tinggi



```

<?php
$x="1"; //string dengan nilai angka 1
$y=1; //integer dengan nilai angka 1
echo "test == :".($x == $y);
// TRUE
echo "<br>test === :".($x === $y);
// FALSE, tidak identic
?>

```

## Operator Kontrol Kesalahan (Error)

PHP mendukung satu operator kontrol kesalahan yaitu tanda (“@”). Ketika operator kontrol kesalahan dikenakan pada ekspresi, bisa jadi jika ada kesalahan maka kesalahan tersebut akan diabaikan. Jika fitur *track\_errors* diaktifkan (ada file *php.ini*), maka pesan kesalahan akan bisa ditampilkan, ekspresi kesalahan disimpan dalam variable *\$php\_errormsg*. Variable ini akan selalu ditimpa setiap kali ada kesalahan.

```

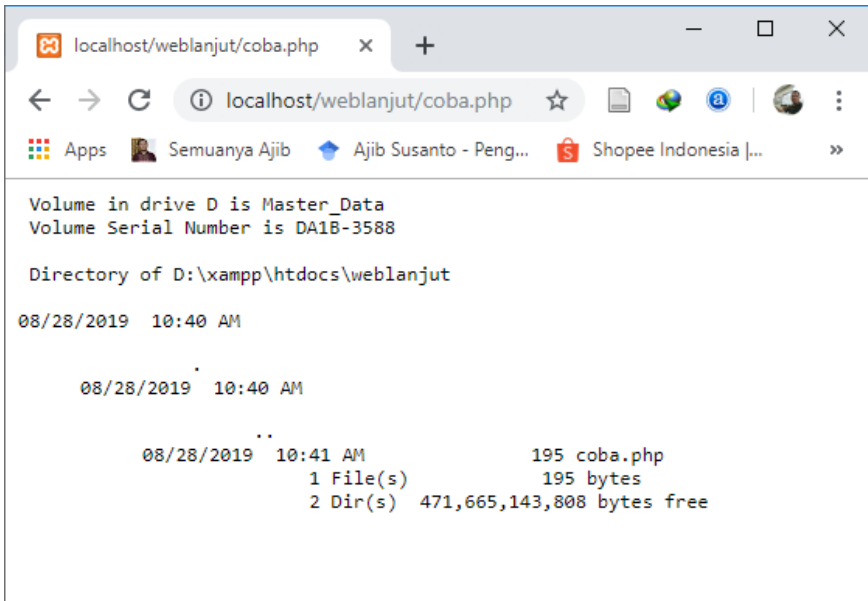
<?php
/* Kesalahan pada file */
$my_file = @file ('non_existent_file') or die
("File gagal dibuka: error was
'$php_errormsg'");
// tidak hanya fungsi tapi juga bekerja
ekspresi
$value = @$cache[$key];
?>

```

## Operator Eksekusi

PHP mendukung satu operator eksekusi yaitu dengan menggunakan *backticks* (` `). PHP akan mengeksekusi baris kode yang berada diantara *backticks* seperti mengeksekusi perintah shell (*shell command*). Penggunaan operator backtick identik dengan fungsi *shell\_exec()*.

```
<?php
// Tampilkan direktori dan file di UNIX/LINUX
$output = `ls -al`;
echo "<pre>$output</pre>";
// Tampilkan direktori dan file di Windows
$output = `dir`;
echo "<pre>$output</pre>";
?>
```



Gambar 2.1 : Hasil halaman tampil direktori

## Operator Penambahan Dan Pengurangan

Operator aritmatika khusus lainnya adalah penambahan (*increment*, `++`) dan pengurangan (*decrement*, `--`). Operator increment adalah operator untuk menambah nilai variable dengan satu, sebaliknya decrement adalah pengurangan nilai variable dengan satu. Peletakkan operator `++` atau `--` bisa sebelum atau sesudah variable, tergantung kebutuhan.

Tabel 2.8 : Operator Penambahan dan Pengurangan

Operator	Keterangan	Contoh	Persamaan
++\$variable	Penjumlahan dengan 1	\$total += 5	\$total = \$total + 5
\$variable++	Penjumlahan dengan 1	\$total -= 5	\$total = \$total - 5
--\$variable	Pengurangan dengan 1	\$total *= 5	\$total = \$total * 5
\$variable--	Pengurangan dengan 1	\$total /= 5	\$total = \$total / 5

```
<?php
$nilai_1 = 100;
echo $nilai_1 . "<br>"; // 100
echo ++$nilai_1 . "<br>"; // 101
echo $nilai_1 . "<br>"; // 101
$nilai_2 = 100;
echo $nilai_2 . "<br>"; // 100
echo $nilai_2++ . "<br>"; //100
echo $nilai_2 . "<br>"; // 101
$nilai_3 = 100;
echo $nilai_3 . "<br>"; // 100
echo --$nilai_3 . "<br>"; // 99
echo $nilai_3 . "<br>"; // 99
$nilai_4 = 100;
echo $nilai_4 . "<br>"; //100
echo $nilai_4-- . "<br>"; //100
echo $nilai_4 . "<br>"; //99
?>
```

## Operator Logika

Beberapa nilai ekspresi hasil dari operator perbandingan bisa dihubungkan dengan beberapa ekspresi yang lain agar diperoleh perbandingan dan nilai logika baru. Untuk menghubungkannya maka diperlukan operator logika.

Tabel 2.9: Operator Logika

Contoh	Nama	Hasil
<code>\$a and \$b</code>	And	TRUE, jika keduanya: <code>\$a</code> dan <code>\$b</code> adalah TRUE.
<code>\$a or \$b</code>	Or	TRUE, jika <code>\$a</code> atau <code>\$b</code> adalah TRUE.
<code>\$a xor \$b</code>	Xor	TRUE, jika <code>\$a</code> atau <code>\$b</code> adalah TRUE, tapi tidak keduanya.
<code>! \$a</code>	Not	TRUE jika <code>\$a</code> bukan TRUE.
<code>\$a &amp;&amp; \$b</code>	And	TRUE, jika keduanya: <code>\$a</code> dan <code>\$b</code> adalah TRUE.
<code>\$a    \$b</code>	Or	TRUE, jika <code>\$a</code> atau <code>\$b</code> adalah TRUE.

## Operator String

Ada dua jenis operator string. Antara lain :

- *concatenation* (“.”), dimana string sebelah kanan digabungkan dengan string sebelah kiri yang akan menghasilkan string baru hasil dari penggabungan.
- *concatenation* dengan operator pemberi nilai (“.= ”), dimana string di sebelah kanan digabungkan dengan string sebelah kiri yang hanya membutuhkan satu variable penampung string saja.

```
<?php
```

```
$a = "Hello ";
$b = $a . "World!";
// sekarang $b berisi "Hello World!"
$a = "Hello ";
$a .= "World!";
// sekarang $a berisi "Hello World!"
?>
```

## Operator Array

Tabel 2.10 : Operator Array

Contoh	Nama	Hasil
<code>\$a + \$b</code>	Penggabungan	Gabungan dari \$a dan \$b.
<code>\$a == \$b</code>	Sama Dengan	TRUE jika \$a dan \$b memiliki kesamaan pasangan key/value.
<code>\$a === \$b</code>	Identik	TRUE jika \$a dan \$b memiliki kesamaan pasangan key/value termasuk kesamaan jenis tipe-nya (identik).
<code>\$a != \$b</code>	Tidak Sama Dengan	TRUE jika \$a tidak sama dengan \$b.
<code>\$a &lt;&gt; \$b</code>	Tidak Sama Dengan	TRUE jika \$a tidak sama dengan \$b.
<code>\$a !== \$b</code>	Tidak identik	TRUE jika \$a tidak identik dengan \$b.

Operator “+” adalah menggabungkan array sebelah kanan ke array sebelah kiri, jika ada key yang sama maka key tidak akan ditimpa.

```
<?php
```

```
$a = array("a" => "apple", "b" => "banana");
$b = array("a" => "pear", "b" => "strawberry",
"b" =>"cherry");
$c = $a + $b; // Menggabungkan $a dan $b
echo "Hasil penggabungan dari \$a and \$b: \n";
var_dump($c);
$c = $b + $a; // Menggabungkan $b dan $a
echo "Hasil penggabungan dari \$b dan \$a: \n";
var_dump($c);
/* Hasil Penggabungan adalah :
Penggabungan dari $a dan $b:
array(3) {
  ["a"]=>
  string(5) "apple"
  ["b"]=>
  string(6) "banana"
```

```

["c"]=>
string(6) "cherry"
}
Penggabungan dari $b dan $a:
array(3) {
["a"]=>
string(4) "pear"
["b"]=>
string(10) "strawberry"
["c"]=>
string(6) "cherry"
}
*/
?>

```

Contoh perbandingan array :

```

<?php
$a = array("apple", "banana");
$b = array(1 => "banana", "0" => "apple");
var_dump($a == $b); // bool(true)
var_dump($a === $b);
// bool(false) karena tidak identik, tidak setipe
?>

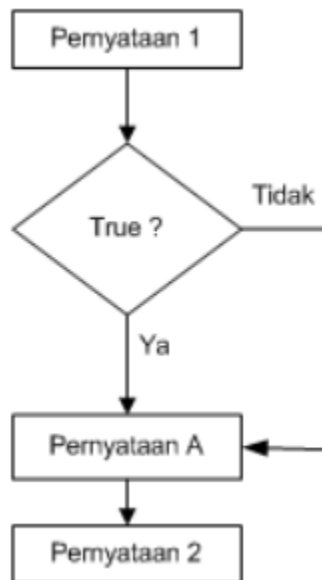
```

## Struktur Kontrol

Script PHP dibangun oleh serangkaian pernyataan, dimana pernyataan tersebut bias berupa pemberi nilai (*assignment*), pemanggil fungsi, perulangan (*loop*), pernyataan kondisi, atau apapun termasuk pernyataan kosong. Dan pernyataan-pernyataan tersebut bisa diatur alur kerjanya dengan adanya stuktur kontrol. Agar lebih jelas, berikut ini akan dijelaskan jenis struktur kontrol yang ada dalam PHP :

## IF

Pernyataan If merupakan percabangan bersyarat yang berfungsi untuk melewati suatu proses, jika syarat terpenuhi maka akan dilakukan proses yang berikutnya. Proses bisa berupa satu instruksi atau beberapa instruksi dalam satu kelompok



Gambar 2.2 : Flowchart IF

Struktur if satu instruksi :

**If (syarat)**

**instruksi;**

```
<?php
```

```
$panjang = 30;
```

```
$lebar = 80;
```

```
$luas = $panjang * $lebar;
```

```
$maks = 100;
```

```
if ( $luas > $maks)
```

```
echo "Luas lebih dari $maks";
```

```
?>
```

Struktur if dengan sekumpulan instruksi:

```
If (syarat)
```

```
{
```

```
    instruksi;
```

```
    instruksi;
```

```
    instruksi;
```

```
}
```

```
<?php
```

```
// Nilai $panjang dan $lebar bisa diganti,
```

```
// agar lebih mudah memahami.
```

```
$panjang = 30;
```

```
$lebar = 80;
```

```
$luas = $panjang * $lebar;
```

```
$maks = 100;
```

```
if ( $luas > $maks)
```

```
{
```

```
echo "Panjang = $panjang <br>";
```

```
echo "Lebar = $lebar <br>";
```

```
echo "Luas Yang Dihasilkan = $luas <br>";
```

```
echo "Luas Maksimal = $maks <br>";
```

```
echo "Luas lebih dari $maks";
```

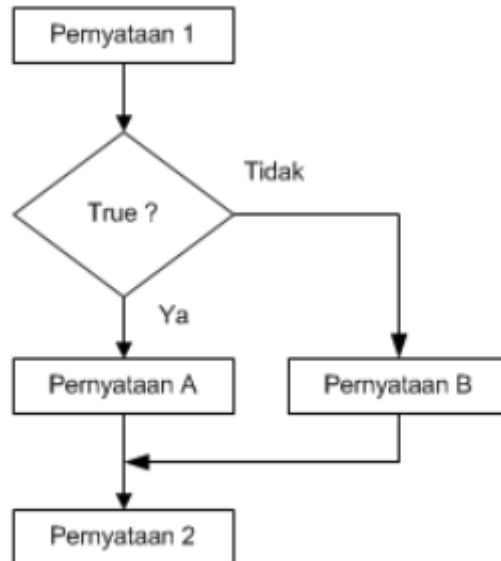
```
}
```

```
?>
```

## **IF...ELSE**

Struktur if...else memiliki dua alur percabangan, sehingga mempunyai alternatif. If...else bisa menuju ke proses berikutnya walaupun syarat terpenuhi atau tidak terpenuhi.





Gambar 2.3 : Flowchart IF ... ELSE

Struktur If ... Else dengan satu instruksi :

```
If (syarat)  
    Instruksi;  
Else  
    Instruksi;
```

Contoh:

```
<?php  
$angka = 4;  
$sisia = $angka % 2; // mencari sisa bagi  
if ($sisia == 0)  
echo "$angka merupakan bilangan genap";  
else  
echo "$angka merupakan bilangan ganjil";  
?>
```

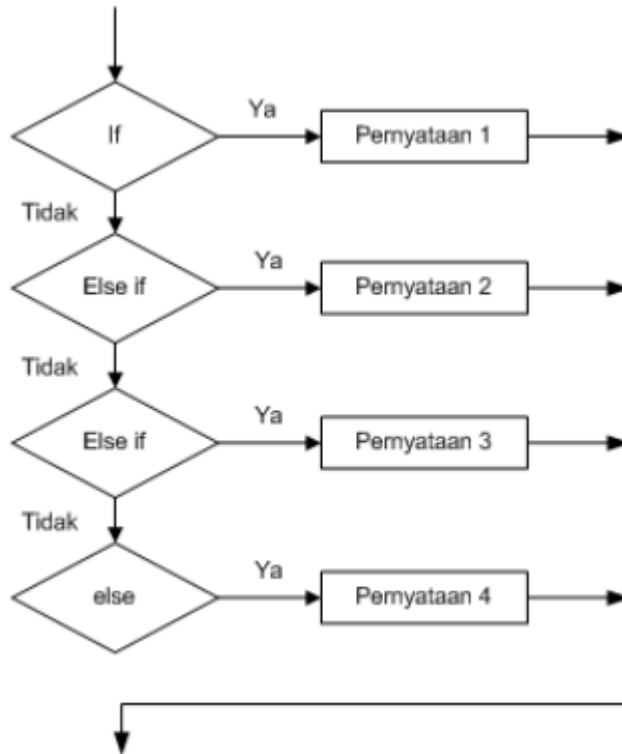
Struktur If ... Else dengan sekumpulan instruksi:

```
If (syarat)  
{  
    instruksi;
```

```
        instruksi;
    }
Else
{
    instruksi;
    instruksi;
}
<?php
$angka = 4;
$sisia = $angka % 2; // mencari sisa bagi
if ($sisia == 0)
echo "$angka merupakan bilangan genap";
else
echo "$angka merupakan bilangan ganjil";
?>
```

### **IF...ELSEIF...ELSE**

Pernyataan `if ... elseif ... else` merupakan pengembangan dari pernyataan `if ... else` namun memiliki percabangan lebih dari dua. `if ... elseif ... else` sesuai untuk memecahkan masalah yang membutuhkan banyak percabangan karena banyak alternatif yang bisa diperoleh. Pernyataan `if ... elseif ... else` boleh dikatakan juga sebagai `if` di dalam `if`.



Gambar 2.4 : Flowchart IF ... ELSEIF ... ELSE  
Struktur if ... elseif ... else adalah:

```
If (syarat)
{
    instruksi;
    instruksi;
} elseif (syarat)
{
    instruksi;
    instruksi;
} elseif (syarat)
{
    instruksi;
    instruksi;
```

```

} else
{
    instruksi;
    instruksi;
}
<?php
$a = 9; // silahkan dirubah nilainya untuk
mengetahui hasil yang diperoleh
$b = 3;
if ($a > $b) {
    echo "a is lebih besar dari b";
} elseif ($a == $b) {
    echo "a sama dengan b";
} else {
    echo "a lebih kecil dari b";
}
?>

```

## Operator Kondisi Ternary

Untuk menyederhanakan pernyataan `if ... else`, PHP memiliki operator kondisi ternary yang cukup singkat.

Dengan struktur sebagai berikut :

```

variable = ekspresi logik atau relasi ?
ekspresi B:ekspresi S

```

Penjelasan struktur Operator Kondisi :

- **Variable**, variable yang akan berisi nilai ekspresi 1 atau ekspresi 2.
- **Ekspresi logik atau relasi**, merupakan ekspresi yang dijalankan untuk memeriksa kondisi, sehingga menghasilkan nilai benar atau salah.
- **Ekspresi B**, akan dijalankan jika ekspresi logika atau relasi bernilai benar (true).
- **Ekspresi S**, akan dijalankan jika ekspresi logika atau relasi bernilai salah (false)

```
<?php
$umur = 5; // bagaimana jika $umur lebih dari
5?
If ($umur <= 5)
$usia = "Balita";
Else
$usia = "Remaja atau dewasa";
echo $usia . "<br>";
// Script di atas dapat dituliskan seperti
berikut ini :
$usia = $umur <= 5 ? "Balita" : "Remaja atau
dewasa";
echo $usia;
?>
```

## SWITCH

Pernyataan switch hampir sama dengan pernyataan if ... elseif ... else tetapi dengan pendekatan yang berbeda. Keduanya sama-sama yang memiliki banyak percabangan, perbedaannya adalah, variable yang dijadikan syarat harus bernilai pasti dan bukan lagi sebagai ekspresi.

Strukturnya Switch :

```
switch ($variable)
{
case nilai_variable_1:
    instruksi;
    instruksi;
    break;
case nilai_variable_2:
    instruksi;
    instruksi;
    break;
case nilai_variable_3:
    instruksi;
    instruksi;
    break;
```

```
default:
    instruksi;
    instruksi;
}
```

Penjelasan struktur switch :

- Switch diawali dengan tanda “{“ dan diakhiri dengan tanda “}”.
- Setiap case harus diakhiri dengan break, fungsi break adalah untuk menghentikan proses pencarian jika syarat sudah terpenuhi. Jika tidak menggunakan break maka meskipun nilai sudah ditemukan sesuai syaratnya, proses akan tetap berjalan sampai pada alternatif terakhir dan nilai yang dihasilkan adalah nilai case paling akhir walaupun nilai yang sesuai berada diposisi sebelumnya.
- Default bertipe pilihan (*optional*), bisa disertakan boleh juga tidak. Default disertakan jika dari sekian banyak alternatif case tidak satu pun yang memenuhi syarat, maka instruksi default akan dilakukan.

Contoh kemiripan penggunaan if dan switch :

```
<?php
// Percabangan dengan if
if ($i == 0) {
    echo "i sama dengan 0";
}
if ($i == 1) {
    echo "i sama dengan 1";
}
if ($i == 2) {
    echo "i sama dengan 2";
}
// Percabangan dengan switch
switch ($i) {
case 0:
    echo "i sama dengan 0";
    break;
case 1:
    echo "i sama dengan 1";
    break;
```

```
case 2:
    echo "i sama dengan 2";
    break;
}
?>
```

Contoh switch disertai default :

```
<?php
switch ($i) {
    case 0:
        echo "i sama dengan 0";
        break;
    case 1:
        echo "i sama dengan 1";
        break;
    case 2:
        echo "i sama dengan 2";
        break;
    default:
        echo "i tidak sama dengan 0, 1 atau 2";
}
?>
```

## Struktur Kontrol Alternatif

PHP menawarkan sintaks alternatif untuk struktur kontrol *if*, *while*, *for*, *foreach*, dan *switch* yang berguna untuk mengelompokkan beberapa instruksi yang akan dieksekusi. Sintaks alternatif tersebut diawali dengan tanda titik dua (“ : ”) dan diakhiri dengan *endif;*, *endwhile;*, *endfor;*, *endforeach;*, atau *endswitch;* sesuai dengan struktur kontrolnya. Fungsinya mirip dengan penanda pasangan kurawal (“ { }”) yang biasa digunakan untuk mengelompokkan beberapa instruksi.

```
<?php
if ($a == 5):
    echo "a sama dengan 5";
    echo "...";
elseif ($a == 6):
```

```
        echo "a sama dengan 6";
        echo "!!!";
else:
echo "a bukan 5 ataupun 6";
endif;
?>
```

## Perulangan

Perulangan adalah proses mengulang-ulang eksekusi blok kode tanpa henti, selama kondisi yang dijadikan acuan terpenuhi. Biasanya disiapkan variabel untuk iterasi atau variabel penanda kapan perulangan akan diberhentikan.

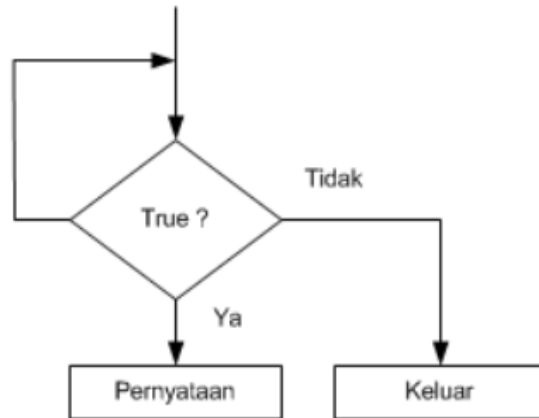
Pada PHP ada 4 jenis perulangan yang bisa kita gunakan:

1. Perulangan While
2. Perulangan Do/While
3. Perulangan For
4. Perulangan Foreach

### WHILE

Perulangan while akan melakukan pengecekan syarat di awal blok perulangan. Selama syarat bernilai benar maka perulangan terus berlanjut dan sebaliknya jika syarat bernilai salah maka perulangan akan dihentikan.





Gambar 2.5 : Flowchart WHILE

Struktur While untuk instruksi tunggal :

```
while (syarat)
    instuksi;
```

Struktur While dengan sekumpulan instruksi :

```
while (syarat)
{
    instuksi;
    instuksi;
}
```

atau juga bisa dalam bentuk berikut ini :

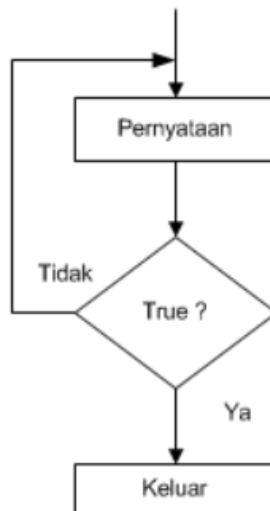
```
while (syarat)
    instuksi;
    instuksi;
endwhile;
```

```
<?php
/* Contoh 1 */
```

```
$i = 1;
while ($i <= 10)
{
    echo $i++; //12345678910
}
echo "<br>";
/* Contoh 2 */
$i = 1;
while ($i <= 10):
    echo $i; //12345678910
    $i++;
endwhile;
?>
```

## DO – WHILE

Perulangan do ... while merupakan kebalikan dari perulangan for dan while, karena perulangan do ...while melakukan pengecekan terhadap syarat diakhir blok perulangan. Jadi do... while paling sedikit akan melakukan satu kali proses perulangan.



Gambar 2.6 : Flowchart DO ... WHILE

Struktur do...while adalah sebagai berikut:

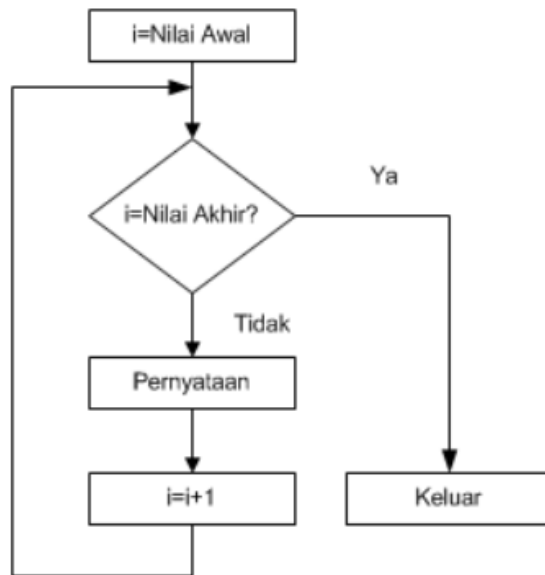
```
Do
{
    instruksi;
    instruksi;
} while (syarat)
```

Berikut contoh mencetak bilangan 1 sampai dengan 10, termasuk nilai totalnya:

```
<?php
$bil = 1;
$total = 0;
do
{
    $total += $bil;
    echo $bil . " " . $total . "<br>";
    $bil++;
} while ($bil <= 10)
/* Hasilnya
1 1
2 3
3 6
4 10
5 15
6 21
7 28
8 36
9 45
10 55
*/
?>
```

## FOR

Struktur kontrol For lebih cocok untuk perulangan dengan jumlah pencacah yang pasti atau sudah diketahui.



Gambar 2.7 : Flowchart FOR

Struktur For dengan instruksi tunggal :

```
for (inialisasi; syarat ; pencacah)  
    instuksi;
```

Struktur For dengan sekumpulan instruksi :

```
for (inialisasi; syarat ; pencacah)  
{  
    instuksi;  
    instuksi;  
}
```

Penjelasan struktur for :

- **inisialisasi**, merupakan nilai awal saat perulangan for dilakukan.
- **syarat**, untuk mengevaluasi setiap kali perulangan dilakukan, berhenti tidaknya suatu perulangan ditentukan oleh syarat tersebut. Syarat bisa berupa ekspresi relasional atau ekspresi logika. Jika syarat bernilai benar maka perulangan dilanjutkan dan jika syarat bernilai salah maka perulangan akan dihentikan.
- **pencacah**, untuk mengatur perubahan nilai variable pencacah yang nilainya bisa diatur menaik atau menurun sesuai dengan kebutuhan.
- **inisialisasi**, **syarat** dan **pencacah** dalam perulangan for disebut dengan argumen dan argumen tidak harus diisi lengkap. Meskipun argumen tidak harus lengkap, tanda titik koma (;) harus tetap dicantumkan di setiap argumen yang kosong.

```
<?php
// Contoh perulangan for dengan argumen
lengkap.
for ($i=1;$i<=10;$i++)
{
    $total += $i;
    echo $i . " " . $total . "<BR>";
}
// Contoh perulangan for dengan argumen tidak lengkap.
// Argumen syarat yang kosong, tetapi diganti dengan
// syarat di bawahnya.
for ($i = 1;;$i++)
{
    if ($i > 10) {
        break;
    }
    echo $i;
}
// Perulangan for tanpa argumen sama sekali,
// menyebabkan perulangan dilakukan terus menerus.
// Perulangan dihentikan secara paksa dengan
// instruksi
```

```
        break.  
        $i = 1;  
for (;;)   
{  
    if ($i > 10)  
        { break; }  
    echo $i;  
    $i++;  
}  
?>
```

## FOREACH

Perulangan foreach mulai digunakan pada PHP versi 4 ke atas. Perulangan foreach banyak digunakan bersamaan dengan data array, karena untuk mengakses data array akan jauh lebih mudah dan praktis. Struktur foreach adalah sebagai berikut :

```
foreach(array_expression as $value) statement  
foreach(array_expression as $key => $value)  
statement
```

Untuk lebih jelasnya, lihat contoh berikut :

Contoh 1: foreach yang hanya menampilkan value saja.

```
<?php  
$a = array (1, 2, 3, 17);  
foreach ($a as $v)  
{  
    echo "Nilai \$a saat ini adalah: $v.<br>";  
}  
/*  
Nilai $a saat ini adalah: 1.  
Nilai $a saat ini adalah: 2.  
Nilai $a saat ini adalah: 3.  
Nilai $a saat ini adalah: 17.
```

```
*/  
?>
```

Contoh 2: foreach yang menampilkan value dengan key.

```
<?php  
$a = array (1, 2, 3, 17);  
$i = 0;  
foreach($a as $v)  
{  
    echo "\$a[$i] => $v.<br>";  
    $i++;  
}  
/* Hasilnya  
$a[0] => 1.  
$a[1] => 2.  
$a[2] => 3.  
$a[3] => 17.  
*/  
?>
```

Contoh 3: foreach yang menampilkan key dan value.

```
<?php  
$a = array ("one" => 1, "two" => 2, "three" =>  
3, "seventeen" =>17);  
foreach($a as $k => $v)  
{  
    echo "\$a[$k] => $v.<br>";  
}  
/* Hasilnya  
$a[one] => 1.  
$a[two] => 2.  
$a[three] => 3.  
$a[seventeen] => 17.  
*/  
?>
```

Contoh 4: foreach yang menampilkan array multi-dimensi

```
<?php
$a[0][0] = "a";
$a[0][1] = "b";
$a[1][0] = "y";
$a[1][1] = "z";
foreach($a as $v1)
{
    foreach ($v1 as $v2)
    {
        echo "$v2";
    }
}
// Hasilnya abyz
?>
```

## BREAK

Pada contoh-contoh sebelumnya pernyataan *break* sudah pernah disinggung sedikit, yaitu bertujuan untuk menghentikan dan keluar dari struktur perulangan *for*, *foreach*, *while*, *do...while* atau struktur *switch* kemudian melanjutkan eksekusi program ke baris perintah berikutnya. *Break* bisa ditambahkan dengan argumen angka, yang menunjukkan kelompok struktur keberapa yang akan dihentikan.

Contoh *break* pada struktur *for* :

```
<?php
for ($i=1;$i<=10;$i++)
{
if ($i==6)
    break;
    echo $i ." ";
}
echo "Akhir pengulangan";
?>
```



Contoh *break* yang dikenakan pada struktur *while* dan *switch*, dimana *break* memiliki dua tingkat stuktur :

```
<?php
$i = 0;
while (++$i)
{
switch ($i)
{
case 5:
    echo "Di 5<br>";
    break 1;
    /* break 1, keluar hanya di stuktur switch. */
case 10:
    echo "Di 10, keluar<br>";
    break 2;
    /* break 2, keluar dari struktur switch dan
    struktur while. */
default:
    break;
}
}
?>
```

## CONTINUE

Pernyataan *continue* digunakan pada struktur perulangan. Bertujuan untuk melewati suatu tahap perulangan dan melanjutkan kembali ke proses perulangan selanjutnya. *Continue* bisa ditambahkan argumen angka yang berarti memberitahukan, struktur seberapa yang akan dihentikan.

```
<?php
for ($i=1;$i<=10;$i++)
{
    if ($i == 7) continue;
}
```

```
        echo "  $i"; //1 2 3 4 5 6 8 9 10
    }
?>
```

## REQUIRE, INCLUDE, REQUIRE\_ONCE, INCLUDE\_ONCE

Terkadang suatu halaman web membutuhkan file lain agar halaman web tersebut bias bekerja. Agar file beserta isinya bisa dikenali di halaman bersangkutan, bisa digunakan pernyataan *REQUIRE()*, *INCLUDE()*, *REQUIRE\_ONCE()*, *INCLUDE\_ONCE()*.

Pemanggilan file sangat berguna untuk menghemat penulisan baris-baris program. Beberapa baris program (misalnya: fungsi, variable) yang penting dan sering digunakan di setiap halaman web bias dikumpulkan menjadi satu, selanjutnya digunakan secara berulang tanpa menulis baris program lagi.

Pernyataan *require()* dan *include()* memiliki perilaku yang sama, kecuali dalam hal penanganan kesalahan. Jika pernyataan *require()* digunakan kemudian timbul kesalahan, maka proses dihentikan dan halaman web tidak akan nampak, sebaliknya jika pernyataan *include()* digunakan, maka pesan peringatan kesalahan akan Nampak dan proses tetap berlanjut.

Pernyataan *require\_once()* dan *include\_once()* fungsinya sama persis dengan *require()* dan *include()*, perbedaannya bahwa pemeriksaan terhadap file yang di-*require\_once()* atau di- *include\_once()*-kan dilakukan sekali saja.

```
<?php
// Contoh require()
require 'prepend.php';
require $somefile;
require ('somefile.txt');
?>
```

```
<?php
// Contoh require_once()
require_once("a.php");
require_once("A.php");
?>
```

```
<?php
// Contoh include_once()
include_once("a.php");
include_once("A.php");
?>
```

## Latihan

### Pertemuan 1 : PHP Dasar

1. Download Materi contohphp.rar
2. Buat program dengan PHP, jgn lupa simpan file PHP di htdocs

#### a. Menampilkan tulisan

Nim : xxx  
Nama : xxxxxxxx  
Alamat : xxxxxxxxxxxx  
Telepon : xxxxxxxxxxxx  
Email : xxxxxxxx

#### b. Menghitung Nilai Pemrograman Web Lanjut, gunakan IF dan switch

Nim : xxxx  
Nama : xxxx  
N. Tugas : 999 30 prosen : 999  
N. UTS : 999 35 prosen : 999

N. UAS : 999      35 prosen : 999

N. Akhir : 999

N. Huruf : 999

- Keterangan
  - A       $\geq 85$
  - AB      $\geq 80$
  - B       $\geq 70$
  - BC      $\geq 65$
  - C       $\geq 60$
  - D       $\geq 50$
  - E       $< 40$

b. Buat Program Konversi Bilangan desimal ke Biner, octa, hexa, gunakan looping

c. Buat Program Konversi suhu, gunakan looping

Dari	ke			
	Celsius	Reamur	Fahrenheit	Kelvin
Celsius		$\frac{4}{5}C$	$\frac{9}{5}C + 32$	$C + 273$
Reamur	$\frac{5}{4}R$		$\frac{9}{4}R + 32$	$\frac{5}{4}R + 273$
Fahrenheit	$\frac{5}{9}(F - 32)$	$\frac{4}{9}(F - 32)$		
Kelvin	$K - 273$	$\frac{4}{5}(K - 273)$		

## Konversi Bilangan :

Desimal	Biner	Oktal	Hexa
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	a
11	1011	13	b
12	1100	14	c
13	1101	15	d
14	1110	16	e
15	1111	17	f
16	10000	20	10

## Konversi Suhu :

**Hasil Perhitungan**

Hasil : Celcius = 2 °C, Fahrenheit = 35.6 °F, Rheamur = 1.6 °R, Kelvin = 275.15 °K, Rankine = 495.27 °Ra

Hasil : Fahrenheit = 2 °F, Celcius = -16.667 °C, Rheamur = -13.333 °R, Kelvin = 256.483 °K, Rankine = 461.67 °Ra

Hasil : Rheamur = 2 °R, Celcius = 2.5 °C, Fahrenheit = 36.5 °F, Kelvin = 275.65 °K, Rankine = 496.17 °Ra

Hasil : Kelvin = 2 °K, Celcius = -271.15 °C, Fahrenheit = -456.07 °F, Rheamur = -216.92 °R, Rankine = 3.6 °Ra

Hasil : Rankine = 2 °Ra, Kelvin = 1.111 °K, Celcius = -272.039 °C, Fahrenheit = -457.67 °F, Rheamur = -217.631 °R

Pilihan konversi :

- Ke Celcius
- Ke Fahrenheit
- Ke Rheamur
- Ke Kelvin
- Ke Rankine

Nilai yang mau dikonversi : 2

# Bab 3

## FUNGSI-FUNGSI DI PHP

---

Bab ini membahas:

- Fungsi
  - Fungsi Array
  - Fungsi String
  - Fungsi Date
- 

### Fungsi

Fungsi adalah sekumpulan baris-baris program yang merupakan serangkaian perintah program yang disusun sedemikian rupa sehingga bisa menjadi satu module saja.

Ketika programmer membuat program yang cukup rumit/kompleks maka perlu digunakan fungsi, karena fungsi akan sangat membantu menyederhanakan penulisan program.

Adapun manfaat adanya fungsi adalah sebagai berikut :

- **Mengurangi penulisan program yang sama**  
Jika kita sering menggunakan program-program yang sama, maka kita dapat membuatnya menjadi suatu module fungsi, sehingga dengan fungsi tersebut bias dipanggil di tempat lain tanpa membuat ulang baris program tersebut, dengan demikian program yang dibuat akan menjadi lebih pendek.
- **Kemudahan untuk melacak dan memperbaiki program**  
Dengan fungsi akan sangat mudah dalam melacak dan memperbaiki baris program, karena jika ada kesalahan tidak

---

perlu untuk memeriksa isi program secara keseluruhan tetapi cukup dilihat per bagian fungsi saja.

- **Bisa dipanggil dimana saja termasuk di fungsi yang lain.**  
Karena fungsi bisa dipanggil di fungsi yang lain maka program yang kompleks bias dibuat menjadi lebih sederhana dan mudah untuk dipahami.

Dalam segi pembuatan, fungsi terbagi dua :

- **Fungsi Built-in yang disediakan PHP.**  
Pemrogram tinggal memanggil nama fungsinya saja dan siap digunakan. Seluruh isi module fungsi sudah dibuatkan oleh PHP secara Built-in.
- **Fungsi buatan sendiri.**  
Pemrogram juga diberi kebebasan untuk membuat fungsi sendiri, hal ini dilakukan jika dirasa fungsi yang dibutuhkan tidak disediakan oleh PHP secara built-in

## Fungsi Buatan Sendiri

Di dalam pemrograman PHP programmer diberi kebebasan untuk membuat fungsi sendiri kalau memang dirasa perlu untuk membuatnya. Tata cara penamaan sebuah fungsi mengikuti aturan yang sama dengan tata cara penamaan variable, selama masih sesuai dengan aturan PHP maka programmer diberi kebebasan memberi nama fungsi. Nama fungsi yang benar diawali dengan huruf atau garis bawah (underscores) kemudian diikuti dengan angka huruf, angka, huruf atau garis bawah. Struktur Fungsi :

```
Function nama_fungsi (argumen1 [, argumen2 , ...] )
{
    Baris program1;
    Baris program2;
    Baris programN;
    [return]
}
```

### Contoh Fungsi Kondisi :

```
<?php
$makefoo = true;
/* fungsi foo() tidak bisa dipanggil dari sini,
sebab fungsi foo() dianggap belum didefinisikan,
tetapi kita bisa memanggil fungsi bar() */
bar();
if ($makefoo) {
function foo()
{
echo "Aku tidak ada sampai eksekusi program
mencapaiku disini.\n";
}
}
/* Sekarang fungsi foo() bisa dipanggil disini,
selama $makefoo bernilai true */
if ($makefoo) foo();
function bar()
{
echo "Aku ada saat program dijalankan pertama
kali.\n";
}
?>
```

### Contoh fungsi dalam fungsi :

```
<?php
function foo()
{
function bar()
{
echo "Aku tidak ada sampai fungsi foo()
dipanggil.\n";
}
}
```



```

/* Kita belum bisa memanggil fungsi bar()
selama fungsi bar() belum didefinisikan */
foo();
/* Sekarang kita bisa memanggil fungsi bar(),
karena proses fungsi foo() membuat fungsi bar()
bisa diakses */
bar();
?>

```

Dalam PHP sebuah fungsi bisa memanggil dirinya sendiri secara berulang, atau yang lebih dikenal dengan fungsi rekursif. Perhitungan matematika yang paling sering digunakan adalah untuk mencari nilai faktorial suatu bilangan. Faktorial  $m!$  didefinisikan sebagai berikut : bernilai 1 jika  $m! = 0$  dan  $m*(m-1)$ , jika  $m > 0$ .

Contoh fungsi dalam fungsi yang bersifat rekursif (berulang) :

```

<?php
Function faktorial($m)
{
If ($m == 0)
    return 1;
Else
    return $m * faktorial ($m-1)
}
echo "Faktorial dari 4 : ". faktorial(4);
// hasilnya 24, diperoleh dari 4!=4*3*2*1
?>

```

## Argumen/Parameter Fungsi

Fungsi bisa melewati satu atau lebih argumen yang diletakkan diantara tanda kurung tutup dan kurung buka setelah nama fungsi. Jika jumlah argumen lebih dari satu maka disetiap argumen harus dibatasi dengan tanda koma (“,”). Argumen bisa berupa variable ataupun ekspresi.

Ada dua jenis argumen yang bisa dilewatkan dalam fungsi :

- **Melewatkan argumen fungsi berdasarkan value (*passing by value*).**

Argumen *passing by value* merupakan perlakuan standar dalam fungsi PHP dimana setelah keluar dari fungsi, nilai variabel argumen akan selalu tetap.

- **Melewatkan argumen fungsi berdasarkan rujukan (*passing by reference*).**

Argumen akan ikut berubah setelah keluar dari fungsi jika argumen bersifat *passing by reference*.

Agar argumen fungsi bersifat *passing by reference* maka argumen tersebut harus diawali dengan tanda *ampersan* (“&”).

Contoh 1, Melewatkan parameter dengan *passing by reference*.

```
<?php
function perluasan_string(&$string)
{
    $string .= 'dan tambahannya.';
}
$str = 'Ini adalah string, ';
perluasan_string($str);
echo $str;
// keluaran : 'Ini adalah string, dan
tambahannya.'
// nilai $str bukan lagi 'Ini adalah string,'
?>
```

Contoh 2, Melewatkan parameter *passing by value* yang disertai dengan argument bawaan.

```
<?php
function makecoffee($type = "cappuccino")
{
    return "Membuat secangkir $type.\n";
}
echo makecoffee();
// Membuat secangkir cappuccino.
```

```
echo makecoffee(null); // Membuat secangkir .
echo makecoffee("espresso");
// Membuat secangkir espresso.
?>
```

## Nilai Balik Fungsi

Fungsi bisa diberi nilai balik dengan memberi pernyataan *return*. Nilai balik fungsi tersebut bisa menerima berbagai jenis tipe data, termasuk *list* dan *object*.

Contoh 1, fungsi bernilai balik dengan cara menambahkan *return*.

```
<?php
function luas_kotak($num)
{
    return $num * $num;
}
echo luas_kotak(4);
// keluaran '16'.
?>
```

Pernyataan *return* tidak bisa menghasilkan nilai balik lebih dari satu. Tetapi hal tersebut bisa diatasi dengan menggunakan pernyataan *list*.

Contoh fungsi menggunakan *list* :

```
<?php
function angka_kecil()
{
    return array (0, 1, 2);
}
list ($nol, $satu, $dua) = angka_kecil();
?>
```

## Fungsi Internal (Built-In)

PHP menyediakan banyak sekali fungsi yang siap digunakan. Penggunaan fungsi *built in* tersebut harus sesuai dengan aturan yang telah ditentukan, termasuk jumlah argument yang harus disertakan.

Tidak semua fungsi *built-in* PHP bisa digunakan sebelum *extension library* diaktifkan. Misalnya, saat menggunakan fungsi-fungsi image seperti *imagecreatetruecolor()* maka pustaka “**php\_gd2**” harus diaktifkan, atau saat menggunakan fungsi *mysql\_connect()* maka pustaka “**php\_mysql**” harus diaktifkan dan jika menggunakan fungsi penyandian MHASH maka harus mengaktifkan pustaka “**php\_mhash**”.

Jika hal ini tidak dilakukan biasanya akan muncul peringatan kesalahan “*undefined function*” karena memang belum dikenali.

## Fungsi Array

### **array\_change\_key\_case**

Mengembalikan nama string key dari array menjadi huruf kecil semua (*lowercased*) atau huruf besar semua (*uppercased*). *int case* merupakan pilihan antara CASE\_UPPER atau CASE\_LOWER.

Struktur :

```
array array_change_key_case ( array input [,
int case])
<?php
$input_array = array("FirSt" => 1, "SecOnd" =>
4);
print_r(array_change_key_case($input_array,
CASE_UPPER));
// hasilnya : Array ( [FIRST] => 1 [SECOND] => 4 )
?>
```

### **array\_chunk**

Membagi array menjadi potongan - potongan array. *Preserve\_keys* bersifat pilihan, jika bernilai true maka keys asli akan dipertahankan. Jika bernilai false maka keys akan berubah seiring dengan perubahannya, keys akan dimulai dari 0.

Struktur :

```
array array_chunk ( array input, int size [,
bool preserve_keys])
```

```

<?php
$input_array = array('a', 'b', 'c', 'd', 'e');
print_r(array_chunk($input_array, 2));
// Array ( [0] => Array ( [0] => a [1] => b ) [1]
=> Array ( [0] => c [1] => d ) [2] => Array ( [0]
=> e ) )
print_r(array_chunk($input_array, 2, TRUE));
// Array ( [0] => Array ( [0] => a [1] => b ) [1]
=> Array ( [2] => c [3] => d ) [2] => Array ( [4]
=> e ) )
?>

```

### **array\_count\_values**

Menghitung semua isi atau nilai dalam array

Struktur :

**array array\_count\_values ( array input)**

```

<?php
$array = array (1, "hello", 1, "world",
"hello");
print_r(array_count_values ($array));
// Array ( [1] => 2 [hello] => 2 [world] => 1 )
?>

```

### **array\_diff**

Mencari perbedaan nilai suatu array terhadap array yang lain

Struktur :

**array array\_diff ( array array1, array array2  
[, array ...])**

```

<?php
// Membandingkan $array1 yang tidak dimiliki di $array2
$array1 = array ("a" => "green", "red", "blue",
"red");
$array2 = array ("b" => "green", "yellow",
"red");
print_r(array_diff ($array1, $array2));
// Array ( [1] =>blue )

```

?>

### **array\_fill**

Memberi nilai array dimulai dari index ke *start\_index* sebanyak *num*.

Struktur :

```
array array_fill ( int start_index, int num,  
mixed value)
```

```
<?php
```

```
print_r(array_fill(5, 6, 'banana'));  
// Array ( [5] => banana [6] => banana [7] =>  
banana [8]=> banana[9] => banana [10] => banana )  
?>
```

### **array\_filter**

Menyaring data / element array dengan suatu fungsi yang memiliki nilai kembalian.

Struktur :

```
array array_filter ( array input [, mixed  
callback])
```

```
<?php
```

```
function ganjil($var)  
{  
    return ($var % 2 == 1);  
}  
function genap($var)  
{  
    return ($var % 2 == 0);  
}  
$array1 = array ("a"=>1, "b"=>2, "c"=>3,  
"d"=>4, "e"=>5);  
$array2 = array (6, 7, 8, 9, 10, 11, 12);  
echo "Ganjil :";  
print_r(array_filter($array1, "genap"));  
// Ganjil : Array ( [a] => 1 [c] => 3 [e] => 5 )  
echo "Genap:";  
print_r(array_filter($array2, "ganjil"));
```

---

```
// Genap : Array ( [0] => 6 [2] => 8 [4] => 10 [6] => 12
)
?>
```

### **array\_flip**

Mentransformasi value menjadi key dan key menjadi value suatu array. Jika ada value yang sama, maka key terakhir yang digunakan.

Struktur :

```
array array_flip ( array trans)
```

```
<?php
$trans = array ("a" => 1, "b" => 1, "c" =>2);
$trans = array_flip ($trans);
print_r($trans);
// Array ( [1] => b [2] => c )
?>
```

### **array\_intersect**

Mencari kesamaan atau perpotongan suatu array terhadap array yang lain.

Struktur :

```
array array_intersect( array array1, array
array2[, array...])
```

```
<?php
$array1 = array ("a" => "green", "red",
"blue");
$array2 = array ("b" => "green", "yellow",
"red");
print_r(array_intersect ($array1, $array2));
// Array ( [a] => green [0] => red )
?>
```

### **array\_key\_exists**

Array\_key\_exists bernilai boolean, berfungsi untuk mengecek apakah element key ada dalam set array atau tidak. Bernilai true

jika key ada dalam set array, dan bernilai false jika key tidak ada dalam set array.

Struktur :

```
bool array_key_exists ( mixed key, array search)
```

```
<?php
$search_array = array("pertama" => 1, "kedua"
=> 4);
if (array_key_exists("pertama",
$search_array))
{
echo "Element 'pertama' berada dalam array ";
}
?>
```

### **array\_merge**

Menggabungkan element dari dua atau lebih array. Dengan catatan jika ada nama key yang sama (yang selain angka) maka value akan di timpa.

Struktur :

```
array array_merge ( array array1, array array2 [, array ...])
```

```
<?php
$array1 = array ("warna" => "merah", 2, 4);
$array2 = array ("a", "b", "warna" =>
"hijau", "bentuk" => "trapezoid", 4);
print_r(array_merge ($array1, $array2));
// Array ( [warna] => hijau [0] => 2 [1] => 4 [2]
=> a [3] => b [bentuk] => trapezoid [4] => 4 )
?>
```

### **array\_push**

Menambahkan satu atau lebih element ke dalam array.

Struktur :

```
int array_push ( array array, mixed var [, mixed ...])
```



```
<?php
$stack = array ("jeruk", "pisang");
array_push($stack, "apel", "strawberry");
//jumlah element menjadi 4
print_r($stack);
// Array ( [0] => jeruk [1] => pisang [2] => apel
[3] => strawberry)
?>
```

### **array\_rand**

Mengambil nilai acak element array

Struktur : **mixed array\_rand ( array input [, int num\_req])**

```
<?php
srand ((float) microtime() * 10000000);
// Mencacah waktu
$input = array ("Neo", "Morpheus", "Trinity",
"Cypher", "Tank");
// menghasilkan data acak sebanyak 2
$rand_keys = array_rand ($input, 2);
print          $input[$rand_keys[0]].
", ".$input[$rand_keys[1]];
?>
```

### **array\_reverse**

Membalik element array di mulai dari akhir element menuju ke awal element. Jika *preserve\_keys* bernilai **True** maka value dan key array akan ikut dibalik.

Struktur :

**array array\_reverse ( array array [, bool preserve\_keys])**

```
<?php
$input = array ("php", 4.0, array ("hijau",
"merah"));
print_r(array_reverse ($input));
```

```
// Element value terbalik dengan key tetap sesuai
urutan
// Array ( [0] => Array ( [0] => hijau [1] =>
merah ) [1] => 4 [2] => php )
print_r(array_reverse ($input, TRUE));
// Element value dan key ikut dibalik
// Array ( [2] => Array ( [0] => hijau [1] =>
merah ) [1] => 4 [0] => php )
?>
```

### **array\_slice**

Mengambil element array dimulai dari *offset* dan sejauh *length*. Jika *length* tidak ditentukan maka element diambil semua, mulai dari *offset* sampai akhir array. Jika *offset* bernilai positif maka *offset* dimulai dari kiri array sejauh *length* dan jika *offset* bernilai negative maka *offset* dimulai dari kanan array sejauh *length*.

Struktur :

```
array array_slice ( array array, int offset
[, int length])
```

```
<?php
$input = array ("a", "b", "c", "d", "e");
print_r(array_slice ($input, 2));
// returns "c", "d", and "e"
echo "<br>";
print_r(array_slice ($input, 2, -1));
// returns "c", "d"
echo "<br>";
print_r(array_slice ($input, -2, 1));
// returns "d"
echo "<br>";
print_r(array_slice ($input, 0, 3));
// returns "a", "b", and "c"
?>
```

### **array\_splice**

Membuang element array yang ditandai oleh *offset* dan lebar dari suatu input array, dan menggantinya dengan array lain. Jika *offset*

postif maka porsi yang dibuang dimulai dari kiri sejauh *offset* input array. Jika *offset* negative maka porsi yang dibuang dimulai dari kanan sejauh *offset* input array. Jika *length* tidak ditentukan maka array akan dihapus semua, dimulai dari *offset* sampai akhir array.

Jika *length* ditentukan dan bernilai positif maka beberapa element akan dibuang, dimulai dari *offset* sampai *length* array mulai dari kiri.

Jika *length* ditentukan dan bernilai negatif maka beberapa element akan dibuang, dimulai dari *offset* dan lebarnya dimulai dari akhir array sesuai dengan *length*nya.

Struktur :

```
array array_splice( array input, int offset[,  
int length [,array replacement]])
```

```
<?php  
$input = array ("red", "green", "blue",  
"yellow");  
print_r($input);  
// array input: Array ( [0] => red [1] =>  
green [2] => blue [3] => yellow )  
echo "<br>";  
print_r(array_splice ($input, 2));  
// element yang dibuang: Array ( [0] => blue  
[1] => yellow )  
echo "<br>";  
print_r($input);  
// array input saat ini: Array ( [0] => red  
[1] => green )  
echo "<br><br>";  
$input = array ("red", "green", "blue",  
"yellow");  
print_r($input);
```

```
// array input: Array ( [0] => red [1] =>
green [2] => blue [3] => yellow )echo "<br>";
print_r(array_splice ($input, 1, -1));
// element yang dibuang: Array ( [0] => green
[1] => blue )
echo "<br>";
print_r($input);
// array input saat ini: Array ( [0] => red
[1] => yellow )
echo "<br><br>";
$input = array ("red", "green", "blue",
"yellow");
print_r($input);
// array input: Array ( [0] => red [1] =>
green [2] => blue [3] => yellow )
echo "<br>";
print_r(array_splice ($input, 1,
count($input), "orange"));
// array yang dibuang dan diganti: Array (
[0] => green [1] => blue [2] => yellow )
echo "<br>";
print_r($input);
// array input saat ini: Array ( [0] => red
[1] => orange )
?>
```

### **array\_sum**

Menjumlahkan semua nilai array.

Struktur :

**mixed array\_sum ( array array)**

```
<?php
```

```
$a = array(2, 4, 6, 8);
echo "Jumlah(a) = ".array_sum($a)."<br>";
$b = array("a"=>1.2,"b"=>2.3,"c"=>3.4);
echo "Jumlah(b) = ".array_sum($b)."<br>";
// Jumlah(a) = 20
```

---

```
// Jumlah(b) = 6.9
?>
```

### **array\_unique**

Membuang elemen ganda pada array, sehingga hanya ada satu nilai yang bersifat unik untuk setiap array.

Struktur :

```
array array_unique ( array array)
$input = array ("a" => "green", "red", "b" =>
"green", "blue","red");
$result = array_unique ($input);
print_r($result);
// Array ( [a] => green [0] => red [1] =>
blue )
?>
```

### **array\_values**

Mengembalikan semua nilai array yang diurutkan berdasarkan key angka, dimulai dari urutan ke-0.

Struktur :

```
array array_values ( array input)
<?php
$array = array ("ukuran" => "XL", "warna" =>
"Gold");
print_r(array_values ($array));
// Array ( [0] => XL [1] => Gold )
?>
```

### **Array**

Menciptakan array baru.

Struktur :

```
array array ( [mixed ...])
<?php
// Contoh 1, membuat array 2 dimensi.
$buah = array (
```

```

"fruits"      =>      array      ("a"=>"orange",
"b"=>"banana", "c"=>"apple"),
"numbers" => array (1, 2, 3, 4, 5, 6),
"holes"  => array ("first", 5 => "second",
"third")
);
// Contoh 2
$array = array( 1, 1, 1, 1, 1, 8=>1, 4=>1,
19, 3=>13);
print_r($array);
// index ke-3 yang bernilai 1 akan diisi dengan
3=>13
// Array ( [0] => 1 [1] => 1 [2] => 1 [3] => 13 [4] =>
1 [8] => 1 [9] => 19 )
// Contoh 3
$bulan = array(1 => 'Januari', 'Februari',
'Maret');
print_r($bulan);
// array dengan key sebagai acuan
// Array ( [1] => Januari [2] => Februari [3] => Maret )
?>

```

### Count

Menghitung jumlah element array.

Struktur:

```

int count ( mixed var)
<?php
$a[0] = 1;
$a[1] = 3;
$a[2] = 5;
echo count ($a); // 3
$b[0] = 7;
$b[5] = 9;
$b[10] = 11;
echo count ($b); // 3
?>

```

**Reset**

Mengembalikan pointer atau penunjuk element ke awal array, setelah array dimanipulasi

**Sizeof**

Mengambil jumlah element dari variable array. Fungsinya sama dengan fungsi *count*

**Current**

Untuk mengetahui posisi element array saat ini, fungsi ini sama dengan fungsi *pos*

**Pos**

Untuk memperoleh posisi element array saat ini, fungsinya sama dengan fungsi *current*

**End**

Menempatkan pointer array di akhir element

**Next**

Memajukan pointer array ke kanan satu langkah

**Prev**

Memundurkan pointer ke kiri satu langkah.

```
<?php
// penggunaan fungsi array current, next, prev,
dan end
$transport = array('kaki', 'sepeda', 'mobil',
'pesawat');
$mode = current($transport); // $mode = 'kaki';
$mode = next($transport); // $mode = 'sepeda';
$mode = current($transport); // $mode =
'sepeda';
$mode = prev($transport); // $mode = 'kaki';
$mode = end($transport); // $mode = 'pesawat';
```

```
$mode = current($transport); // $mode =
'pesawat';
?>
```

### **in\_array**

Mencari string dalam array, bernilai **true** jika string ada di dalam array dan **false** jika string tidak ditemukan dalam array. *bool strict* merupakan pilihan, jika bernilai true maka pembandingan akan memperhatikan tipe datanya (tipe yang identik).

Struktur :

```
bool in_array ( mixed needle, array haystack [,
bool strict])
```

```
<?php
$os = array("Mac", "NT", "Irix", "Linux");
if (in_array("Irix", $os)) {
echo "Got Irix"; // Ketemu, ditampilkan
}
if (in_array("mac", $os)) {
echo "Got mac"; // Tidak ketemu, tidak
ditampilkan
}
?>
```

```
<?php
$a = array('1.10', 12.4, 1.13);
if (in_array('12.4', $a, TRUE)) echo "'12.4'
\n";
// Hasilnya tidak ketemu karena tipe data yang
berbeda atau tidak identic
if (in_array(1.13, $a, TRUE)) echo "1.13 \n";
// Hasilnya 1.13
?>
```

### **Sort**

Menata ulang urutan array, yang diurutkan berdasarkan nilainya dimulai dari kecil ke besar (*ascending*).

Struktur :



---

```

void rsort ( array array [, int sort_flags])
<?php
$buah = array ("lemon", "orange", "banana",
"apple");
sort ($buah);
reset ($buah);
while (list ($key, $val) = each ($buah)) {
echo "buah[".$key."] = ".$val."<br>";
}
// Hasil pengurutan
// buah[0] = apple
// buah[1] = banana
// buah[2] = lemon
// buah[3] = orange
?>

```

### **Krsort**

Menata ulang urutan array, yang diurutkan berdasarkan key-nya dimulai dari besar ke kecil (*descending*)

Struktur :

```

int krsort ( array array [, int sort_flags])
<?php
$buah = array ("d"=>"lemon",
"a"=>"orange", "b"=>"banana", "c"=>"apple");
krsort ($buah);
reset ($buah);
while (list ($key, $val) = each ($buah)) {
echo "$key = $val<br>";
}
// d = lemon
// c = apple
// b = banana
// a = orange
?>

```

**Ksort**

Mengurutkan nilai array dari kecil ke besar (*ascending*) berdasarkan key element array.

Struktur :

```
int ksort ( array array [, int sort_flags])
```

```
<?php
$buah = array ("d"=>"lemon", "a"=>"orange",
"b"=> "banana","c"=>"apple");
ksort ($buah);
reset ($buah);
while (list ($key, $val) = each ($buah)) {
echo "$key = $val<br>";
}
// a = orange
// b = banana
// c = apple
// d = lemon
?>
```

**Rsort**

Mengurutkan array dari besar ke kecil (*descending*) berdasarkan nilai/value array.

Struktur :

```
void rsort ( array array [, int sort_flags])
```

```
<?php
$buah = array ("lemon", "orange", "banana",
"apple");
rsort ($buah);
reset ($buah);
while (list ($key, $val) = each ($buah)) {
echo "$key = $val<br>";
}
// Hasil
// 0 = orange
// 1 = lemon
// 2 = banana
```

```
// 3 = apple  
?>
```

### List

Memberikan nilai pada sejumlah daftar variable yang diperoleh dari nilai array.

Struktur :

```
void list ( mixed ...)
```

```
<?php  
$info = array('kopi', 'coklat', 'caffeine');  
// Menampilkan semua variable  
list($drink, $color, $power) = $info;  
echo "$drink $color dan $power membuatnya  
spesial.<br>";  
// kopi coklat dan caffeine membuatnya spesial.  
// Menampilkan hanya beberapa variable  
list($drink, , $power) = $info;  
echo "$drink meningkatkan $power.<br>";  
// kopi meningkatkan caffeine.  
// Melewatkan beberapa variable, kecuali pada  
variable ketiga  
list( , , $power) = $info;  
echo "Saya membutuhkan $power!<br>";  
// Saya membutuhkan caffeine!  
?>
```

### range

Menampilkan rentang nilai array dimulai dari kecil ke besar atau sebaliknya dari besar ke kecil.

Struktur :

```
array range ( mixed low, mixed high)
```

```
<?php  
foreach(range(0, 9) as $number) {  
echo $number; // 0123456789  
}  
echo "<br>";
```

```

foreach(range('a', 'z') as $letter) {
echo $letter;
//hasil: abcdefghijklmnopqrstuvwxyz
}
echo "<br>";
foreach(range('z', 'a') as $letter) {
echo $letter;
//hasil: zyxwvutsrqponmlkjihgfedcba
}
?>

```

### **array\_keys**

Mengambil nilai key suatu array. Jika *mixed search\_value* diisi, maka key yang ditampilkan hanya yang sesuai dengan yang disyaratkan.

Struktur :

```
array array_keys ( array input [, mixed search_value]
```

```

<?php
$array = array (0 => 100, "warna" => "merah");
print_r(array_keys ($array));
// Hasil: Array ( [0] => 0 [1] => warna )
echo "<br>";
$array = array ("biru", "merah", "hijau",
"biru","biru");
print_r(array_keys ($array, "biru"));
// Hasil: Array ( [0] => 0 [1] => 3 [2] => 4 )
echo "<br>";
$array = array ("warna" => array("biru",
"merah", "hijau"),"ukuran" => array("kecil",
"sedang", "besar"));
print_r(array_keys ($array));
// Hasil: Array ( [0] => warna [1] => ukuran )
?>

```

## Fungsi String

Fungsi string berhubungan dengan memanipulasi tipe string.

### **chr**

Mendapatkan satu character string yang telah ditentukan, dan sesuai daftar table ascii.

Struktur:

**string chr ( int \$ascii )**

```
<?php
$str = "String diakhiri oleh escape: ";
$str .= chr(27);
/* penambahan character escape diakhir $str */
/* Lebih berguna jika: */
$str = sprintf("String diakhiri oleh escape:
%c", 27);
?>
```

### **explode**

Membagi string menjadi potongan-potongan array, dimana character pembagi (delimiter) telah ditentukan.

Struktur:

**array explode ( string \$delimiter, string \$string [, int \$limit])**

```
<?php
// Contoh 1
$pizza = "piecel piece2 piece3 piece4 piece5
piece6";
$pieces = explode(" ", $pizza);
echo $pieces[0]; // piecel
echo $pieces[1]; // piece2
// Contoh 2
$data = "foo:*:1023:1000::/home/foo:/bin/sh";
list($user, $pass, $uid, $gid, $gecos, $home,
$shell) = explode(":", $data);
```

```
echo $user; // foo
echo $pass; // *
?>
```

Contoh explode disertai parameter batas:

```
<?php
$str = 'satu|dua|tiga|empat';
// batas positive
print_r(explode('|', $str, 2));
// batas negative (sejak PHP 5.1)
print_r(explode('|', $str, -1));
?>
```

Keluaran dari contoh di atas:

```
Array
(
    [0] => satu
    [1] => dua|tiga|empat
)
Array
(
    [0] => satu
    [1] => dua
    [2] => tiga
)
```

### **htmlentities**

Mengkonversikan semua character menjadi entitas HTML. Fungsi ini identik dengan fungsi *htmlspecialchars()*, dimana semua character yang memiliki kesamaan atau dimiliki oleh character entitas HTML akan diterjemahkan ke entitas tersebut.

Struktur :

```
string htmlentities (string $string [, int $quote_style[, string $charset]] )
```

Tabel 3.1 : *\$quote\_style*

Konstanta	Keterangan
ENT_COMPAT	Akan mengkonversi petik ganda ( <i>double-quotes</i> ) dan membiarkan petik tunggal ( <i>single-quotes</i> ) sendirian.
ENT_QUOTES	Akan mengkonversi petik ganda dan petik tunggal.
ENT_NOQUOTES	Petik tunggal dan petik ganda dibiarkan saja tanpa dilakukan konversi.

```
<?php
$str = "A 'quote' is <b>bold</b>";
// Outputs: A 'quote' is <b>bold</b>
echo htmlentities($str);
// Outputs: A 'quote' is <b>bold</b>
echo htmlentities($str, ENT_QUOTES);
?>
```

### **implode**

Menggabungkan setiap element array menjadi string baru, dimana setiap element array akan direkatkan dengan character *delimiter* (character batas) yang nantinya menjadi string baru. Fungsi implode identic dengan fungsi *join()*

Struktur :

```
string implode ( string $glue, array $pieces )
```

```
<?php
$array = array('lastname', 'email', 'phone');
$comma_separated = implode(",", $array);
echo $comma_separated; // lastname,email,phone
?>
```

### **ltrim**

Memotong spasi kosong (*whitespace*) atau character lain yang posisinya diawal. Struktur : **string ltrim ( string \$str [, string \$charlist] )** Jika parameter kedua (*\$charlist*) tidak diisi, maka fungsi *ltrim()* juga memotong charater berikut ini :

- " " (ASCII 32 (0x20)), ordinary space.
- "\t" (ASCII 9 (0x09)), tab.
- "\n" (ASCII 10 (0x0A)), baris baru (line feed).

- `"\r"` (ASCII 13 (0x0D)), Enter / carriage return.
- `"\0"` (ASCII 0 (0x00)), byte-NUL.
- `"\x0B"` (ASCII 11 (0x0B)), tab vertical.

```
<?php
$text = "\t\tThese are a few words :) ... ";
$binary = "\x09Example string\x0A";
$hello = "Hello World";
var_dump($text, $binary, $hello);
print "\n";
$trimmed = ltrim($text);
var_dump($trimmed);
$trimmed = ltrim($text, " \t.");
var_dump($trimmed);
$trimmed = ltrim($hello, "Hdle");
var_dump($trimmed);
// trim the ASCII control characters at the
beginning of
$binary
// (from 0 to 31 inclusive)
$clean = ltrim($binary, "\x00..\x1F");
var_dump($clean);
?>
```

#### Keluaran :

```
string(32) " These are a few words :) ... "
string(16) " Example string"
string(11) "Hello World"
string(30) "These are a few words :) ... "
string(30) "These are a few words :) ... "
string(7) "o World"
string(15) "Example string"
```



**number\_format**

Memformat angka yang dikelompokkan dalam ribuan.

Struktur :

```
string number_format ( float $number [, int $decimals [,string $dec_point, string $thousands_sep]] )
```

Notasi Perancis menggunakan dua desimal, koma (',') sebagai pemisah desimal dan spasi kosong (' ') sebagai pemisah ribuan, contoh :

```
<?php
$number = 1234.56;
// Notasi Inggris (default)
$english_format_number =number_format($number);
// 1,235
// Notasi Perancis
$nombre_format_francais=number_format($number,
2, ',', ' ');
// 1 234,56
$number = 1234.5678;
// Notasi Inggris tanpda pemisah ribuan
$english_format_number = number_format($number,
2, '.', ' ');
// 1234.57
// Notasi Indonesia
$Indonesia_format_angka
number_format($number,          2,          ',', ' ')
?>
```

**ord**

Memperoleh nilai ASCII dari suatu character. Untuk melihat table ASCII bisa dilihat di <http://www.asciitable.com>.

Struktur :

```
int ord ( string $string )
```

```
<?php
$str = "\n";
if (ord($str) == 10) {
```

```
echo "character pertama dari \$str adalah line
feed.\n";}
?>
```

### **rtrim**

Memotong spasi kosong (*whitespace*) atau character lain yang posisinya diakhir.

Struktur :

```
string ltrim ( string $str [, string $charlist]
)
```

Jika parameter kedua (*\$charlist*) tidak diisi, maka fungsi *rtrim()* juga memotong charater berikut ini :

- " " (ASCII 32 (0x20)), spasi.
- "\t" (ASCII 9 (0x09)), tab.
- "\n" (ASCII 10 (0x0A)), baris baru (line feed).
- "\r" (ASCII 13 (0x0D)), Enter / carriage return.
- "\0" (ASCII 0 (0x00)), byte-NUL.
- "\x0B" (ASCII 11 (0x0B)), tab vertikal.

```
<?php
```

```
$text = "\t\tThese are a few words :) ... ";
$binary = "\x09Example string\x0A";
$hello = "Hello World";
var_dump($text, $binary, $hello);
print "\n";
$trimmed = rtrim($text);
var_dump($trimmed);
$trimmed = rtrim($text, " \t.");
var_dump($trimmed);
$trimmed = rtrim($hello, "Hdle");
var_dump($trimmed);
// trim the ASCII character kontrol di akhir
$binary
// (from 0 to 31 inclusive)
```

```
$clean = rtrim($binary, "\x00..\x1F");
var_dump($clean);
?>
```

Keluaran :

```
string(32) " These are a few words :) ... "
string(16) " Example string"
string(11) "Hello World"
string(30) " These are a few words :) ..."
string(26) " These are a few words :)"
string(9) "Hello Wor"
string(15) " Example string"
```

### **str\_repeat**

Untuk memperoleh string baru yang duhasilkan dari perulangan string.

Struktur :

```
string str_repeat ( string $input, int $multiplier )
```

```
<?php
echo str_repeat("-", 10);
?>
```

Keluaran :

```
-----
```

### **str\_replace**

Mengganti semua string yang sesuai dengan string pencarian. Fungsi *str\_replace()* bersifat case-sensitive untuk memperoleh hasil yang case-insensitive gunakan fungsi *str\_ireplace()*.

Struktur :

```
mixed str_replace ( mixed $search, mixed $replace, mixed $subject [, int &$count] )
```

```
<?php
// Menghasilkan: <body text='black'>
$bodytag = str_replace("%body%", "black",
"<body
```

```
text='%body%>");
// Menghasilkan: Hll Wrld f PHP
$vowels = array("a", "e", "i", "o", "u", "A",
"E", "I", "O", "U");
$onlyconsonants = str_replace($vowels, "",
"Hello World of PHP");
// Menghasilkan: You should eat pizza, beer,
and ice cream every day
$phrase = "You should eat fruits, vegetables,
and fiber every day.";
$healthy = array("fruits", "vegetables",
"fiber");
$yummy = array("pizza", "beer", "ice cream");
$newphrase = str_replace($healthy, $yummy,
$phrase);
$str = str_replace("ll", "", "good golly miss
molly!",
$count);
echo $count; // 2
$str = "Line 1\nLine 2\rLine 3\r\nLine 4\n";
$order = array("\r\n", "\n", "\r");
```

## Fungsi Tanggal & Waktu

Merupakan kumpulan fungsi untuk mendapatkan tanggal dan waktu disisi computer server dimana script PHP berada. Anda dapat menggunakan fungsi-fungsi tersebut untuk membuat format tanggal dan format waktu dengan berbagai cara.

### **checkdate**

Memeriksa apakah argumen tanggal yang ditentukan sudah benar sesuai dengan tanggal Gregorian. Dimana *\$month* merupakan bulan antara 1 sampai dengan 12. *\$day* merupakan tanggal yang sesuai dengan bulan *\$month* bersangkutan, misalnya: di bulan Februari tentu

tidak mungkin jika tanggal *\$day* bernilai 31. Tahun *\$year* merupakan tahun antara 1 sampai dengan 32767

Struktur: **bool checkdate ( int \$month, int \$day, int \$year )**

```
<?php
var_dump(checkdate(12, 31, 2000)); //
bool(true)
var_dump(checkdate(2, 29, 2001)); //
bool(false)
?>
```

### **date**

Mengembalikan nilai tanggal/waktu dalam bentuk string yang telah terformat, dimana format ditentukan oleh argumen yang diberikan. Argumen bersifat pilihan, jika tidak diisi maka akan mengacu pada nilai fungsi *time()*.

Struktur:

**string date ( string \$format [, int \$timestamp] )**

Adapun argumen *\$format* yang bisa dilewatkan antara lain:

Tabel 3.2 : \$format Date

Format Character	Description	Contoh nilai yang dihasilkan
Day	---	---
D	Hari dalam bulan, terdiri dari 2 digit disertai dengan angka 0.	01 sampai 31
D	Hari secara tekstual, terdiri dari tiga huruf.	Mon sampai Sun
J	Hari dalam bulan tidak disertai dengan angka 0.	1 sampai 31
l (lowercase)	Hari dalam minggu, ditampilkan secara	Sunday sampai Saturday

L')	tekstual lengkap	
N	Angka yang mewakili hari dalam satu minggu .	1 (untuk Monday) sampai 7 (untuk Sunday)
S	Character akhiran hari dalam Bahasa Inggris	st, nd, rd or th.
W	Angka yang mewakili hari dalam satu minggu.	0 (untu Sunday) sampai 6 (untuk Saturday)
Z	Hari dalam satu tahun (dimulai dari 0).	0 sampai 365
Week	---	---
W	Jumlah satuan minggu dalam satu tahun, minggu diawali pada hari senin.	Contoh: 42 (Munggu ke 42 pada tahun ini)
Month	---	---
F	Menampilkan bulan tekstual secara lengkap.	January sampai December
M	Angka yang mewakili bulan 2 digit, diawali angka 0.	01 sampai 12
M	Menampilkan bulan secara tekstual pendek tiga huruf	Jan sampai Dec
n	Angka ang mewakili bulan tanpa diawali angka 0.	1 sampai 12
t	Jumlah hari sesuai dengan bulan yang diberikan.	28 sampai 31
Year	---	---
L	Whether it's a leap year	1 if it is a leap year, 0 otherwise.
o	Jumlah tahun. Ini sama sama dengan nilai Y.	Contoh: 1999 atau 2003
Y	Angka yang mewakili tahun secara lengkap 4 digit.	Contoh: 1999 or 2003

y	Angka yang mewakili tahun dengan 2 digit.	Contoh: 99 atau 03
Time	---	---
A	Huruf kecil mewakili Ante meridiem dan Post meridiem	am atau pm
A	Huruf besar mewakili Ante meridiem and Post meridiem	AM atau PM
B	Swatch Internet time	000 sampai 999
G	Format 12-jam tanpa diawali angka 0.	1 sampai 12
G	Format 24-jam tanpa diawali angka 0.	0 sampai 23
H	Format 12-jam diawali dengan angka 0.	01 sampai 12
H	Format 24-jam diawali dengan angka 0.	00 sampai 23
I	Menit, diawali dengan angka 0.	00 sampai 59
S	Detik, diawali dengan angka 0.	00 sampai 59
U	Milidetik	Contoh: 54321
Timezone	---	---
E	Identifikasi Timezone (wilayah waktu).	Contoh: UTC, GMT, Atlantic/Azores
I (capital i)	Apakah hari dalam <i>daylight saving time</i> atau tidak.	1 jika Daylight Saving Time, 0 untuk sebaliknya.
O	Perbedaan Greenwich time (GMT) dalam jam.	Contoh: +0200
P	Perbedaan Greenwich time (GMT) yang disertai dengan titik dua antara jam dan menit.	Contoh: +02:00
T	Singkatan waktu wilayah (Timezone).	Contoh: EST, MDT ...

Full Date/Time	---	---
C	Tanggal sesuai ISO 8601 (ada di PHP 5)	2004-02-12-T15:19:21+00:00
r	Format tanggal sesuai » <a href="#">RFC 2822</a> .	Contoh: Thu, 21 Dec 2000 16:01:07 +0200
U	Jumlah detik dalam jangka waktu UNIX(January 1 1970 00:00:00 GMT)	Lihat juga <i>time()</i>

```
<?php
echo date("D, d F Y") . "<br>";
echo date("g A") . "<br>";
echo date("ds");
?>
```

Hasil :

```
Wed, 19 September 2007
2 PM
19th
```

### **getdate**

Mengembalikan sekumpulan nilai array yang berisi informasi waktu, jika argument *\$timestamp* tidak disertakan maka waktu sekarang yang akan digunakan sebagai acuan adalah fungsi *time()*.

Struktur:

```
array getdate ( [int $timestamp] )
```



Tabel 3.3 : Elemen array yang dihasilkan dari fungsi *getdate()*.

Key	Keterangan	Contoh Nilai
"seconds"	Angka yang mewakili detik	0 sampai 59
"minutes"	Angka yang mewakili menit	0 sampai 59
"hours"	Angka yang mewakili jam	0 sampai 23
"mday"	Angka yang mewakili hari dalam satu bulan	1 sampai 31
"wday"	Angka yang mewakili hari dalam satu minggu	0 (Minggu) sampai 6 (Sabtu)
"mon"	Angka yang mewakili bulan	1 sampai 12
"year"	Angka yang mewakili tahun, ada 4 digit	Contoh: 1999 atau 2003
"yday"	Angka yang mewakili hari dalam satu tahun	0 sampai 365
"weekday"	Mewakili hari dalam satu minggu secara tertulis	Minggu sampai Sabtu
"month"	Mewakili bulan secara tertulis, seperti January atau March	Januari sampai Desember

```
<?php
$today = getdate();
print_r($today);
?>
```

Hasil :

```
Array
(
    [seconds] => 40
    [minutes] => 58
    [hours] => 21
    [mday] => 17
    [wday] => 2
    [mon] => 6
    [year] => 2003
    [yday] => 167
```

```
[weekday] => Tuesday  
[month] => June  
[0] => 1055901520  
)
```

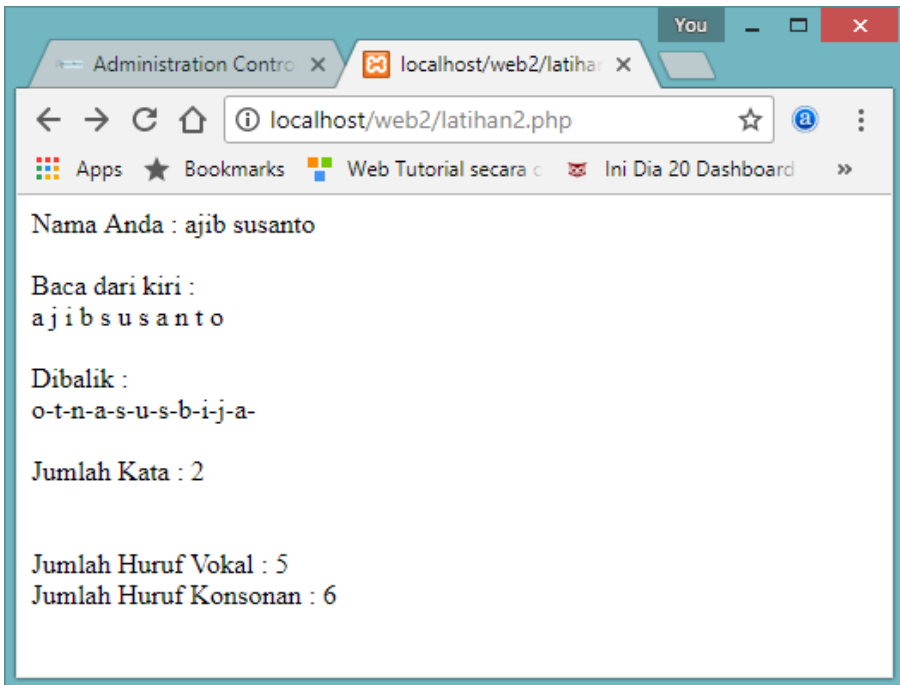
## Latihan

### Pertemuan 2 : String, Array, Fungsi, Date

1. Buat program berikut ini :

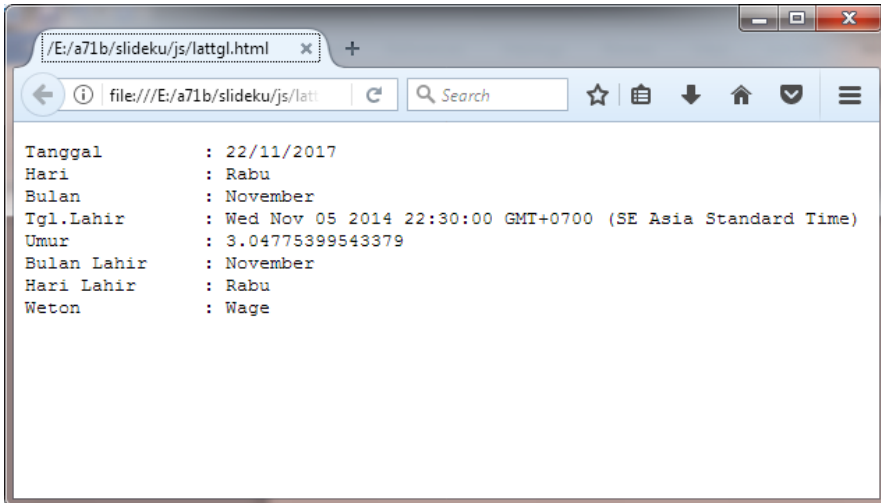
Input : <kalimat> misalkan nama anda

Output :



Gunakan fungsi String, Array dan buat function cekVokal() dan function cekKonsonan()

2. Buat program untuk menampilkan umur, hari, bulan dan weton dengan hasil sebagai berikut :



```
Tanggal      : 22/11/2017
Hari         : Rabu
Bulan        : November
Tgl.Lahir    : Wed Nov 05 2014 22:30:00 GMT+0700 (SE Asia Standard Time)
Umur         : 3.04775399543379
Bulan Lahir  : November
Hari Lahir   : Rabu
Weton        : Wage
```

Ket :gunakan array untuk Hari, Bulan dan Weton

3. Buat program sorting dengan PHP

- bubble sort
- quick sort
- insertion sort
- selection sort
- merge sort

Cetak iterasi perubahan posisi angka

Input :

7 1 3 5 2

1 7 3 5 2

1 3 7 5 2

dst

---

# Bab 4

## FORM, SESSION & COOKIES

---

Bab ini membahas:

- Penanganan Form
  - Session
  - Cookies
- 

### Penanganan Form

Sebuah website dinamis seringkali memerlukan interaksi antara browser client dan server bisa berupa pemasukkan data text, angka, atau upload file (file gambar, file exeutable, dan lain-lain) untuk diproses oleh server.

#### Komponen Form

Untuk mewadahi suatu data yang dikirimkan oleh browser client, dibutuhkan adanya FORM HTML.

Contoh penggunaan form misalnya untuk pendaftaran keanggotaan, pemasukkan kode kartu kredit, login user, transaksi perbelanjaan, upload file, dan lain-lain.

Dalam FORM HTML terdapat beberapa komponen yang bisa digunakan, antara lain :

- **Text Box**, untuk menginputkan data string ataupun angka.

TAG           HTML:           <input            type="text"  
name="nama\_variable" size="30">

- **Text Area**, untuk menginputkan data string ataupun angka yang terdiri dari banyak baris.

---

TAG HTML: `<textarea rows="2" cols="25" name="nama_variable"></textarea>`

- **File Upload**, untuk mengupload data bertipe file.

TAG HTML: `<input type="file" name="nama_variable" size="21">`

- **Radio Buton**, untuk memilih satu pernyataan dari beberapa pernyataan yang disediakan.

TAG HTML: `<input type="radio" value="V1" checked name="nama_variable">`

- **Combo Box**, untuk menampilkan daftar data.

TAG HTML: `<select size="1" name="nama_variable">`  
`<option>Combo1</option>`  
`<option>Combo2</option>`  
`</select>`

- **Check Box**, untuk memilih satu atau lebih pernyataan dari beberapa pernyataan yang disediakan.

TAG HTML: `<input type="checkbox" name="nama_variable" value="ON" checked>`

- **Submit** untuk mengirimkan semua variable data pada komponen-komponen form yang ada.

TAG HTML: `<input type="submit" name="submit" value="Submit">`

- **Reset** untuk membatalkan semua inputan yang telah dituliskan.

TAG HTML: `<input type="reset" name="reset" value="Reset">`

Agar komponen-komponen FORM HTML bisa dikenali sebagai variable yang bernilai data, maka semua komponen harus diletakkan

diantara tanda `<form>...</form>`. Khusus untuk komponen file upload,

TAG HTML FORM harus diubah menjadi `<form enctype="multipart/form-data">...</form>`.

Struktur penulisan HTML Form adalah sebagai berikut :

```
<form name="nama_form" action="tujuan.php"
method="GET">
```

komponen form

```
</form>
```

Atau

```
<form name="nama_form" action="tujuan.php"
method="POST">
```

komponen form

```
</form>
```

Keterangan properti FORM adalah sebagai berikut :

**Name** adalah penamaan untuk form, penamaan form tidak harus disertakan.

**Action** adalah nama file web tujuan yang akan menerima variable data, yang dikirimkan melalui form. File web tujuan bisa juga dikirimkan pada dirinya sendiri, ini berarti setelah menekan tombol submit posisi halaman web akan tetap sama.

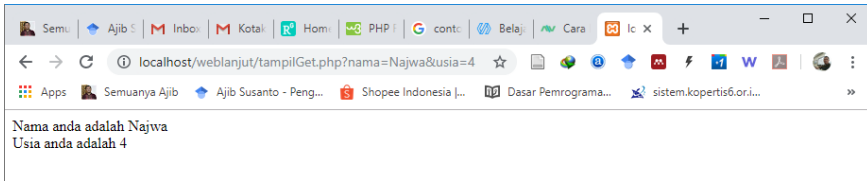
**Method** adalah jenis pengiriman variable data, yang terbagi menjadi dua jenis yaitu

**METHOD=GET** dan **METHOD=POST**.

## Method GET

Dengan method GET, maka nama dan nilai variable akan tampak di address URL browser. Method GET lebih cocok untuk pengiriman

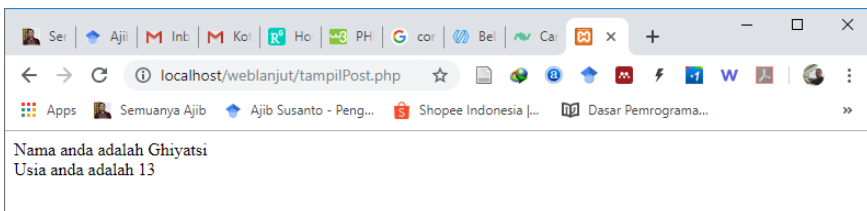
variable disertai argumen yang panjang, serta tidak membutuhkan keamanan lebih.



Gambar 4.1 : Contoh Get

## Method POST

Dengan method POST, maka nama dan nilai variable tidak akan tampak di address URL browser. Method POST lebih cocok untuk pengiriman variable yang membutuhkan pengamanan. Misalnya : untuk halaman web yang memerlukan *login user* dan *password*.



Gambar 4.2 : Contoh Post

## Session

Session merupakan hal yang cukup penting dalam aplikasi berbasis web. Dengan session memungkinkan programmer menyimpan informasi user secara semi-permanen, artinya selama masa tertentu informasi akan tersimpan.

Penyimpanan isi variabel session berada di server, jadi tidak bias diakses secara langsung oleh client. Dalam aplikasi berbasis web, session banyak digunakan sebagai autentifikasi login. Dengan session memungkinkan programmer mengatur siapa saja yang bisa mengakses

suatu halaman. Misalnya saja, untuk melihat halaman kotak surat pada email, kita harus login terlebih dahulu. Dalam proses login antara lain akan terjadi pembuatan suatu session yang akan dibawa oleh user di setiap halaman. Di halaman kotak surat, session tersebut akan diperiksa. Jika session benar maka user dipersilahkan membuka halaman kotak surat, namun jika salah maka user tidak bias membuka halaman kotak surat dan biasanya akan diminta untuk login terlebih dahulu. Itulah sebabnya, user tidak bisa mengakses halaman kotak surat secara langsung tanpa melakukan login.

Dalam penanganan session terdapat beberapa proses yang perlu diperhatikan :

1. Proses pembuatan session
2. Proses pemeriksaan session
3. Proses penghapusan session

Selanjutnya bagaimana session itu sendiri dijalankan? Agar proses penyimpanan dalam session berjalan, PHP melakukan beberapa hal berikut ini :

1. PHP meng-generate (membentuk) sebuah ID session. ID session ini merupakan sejumlah deret angka random yang unik untuk setiap user dan hampir tidak mungkin bisa ditebak. ID session disimpan oleh PHP di dalam variabel sistem PHP dengan nama PHPSESSID
2. PHP menyimpan nilai yang akan Anda simpan dalam session di dalam file yang berada di server. Nama file tempat penyimpanan session tersebut sesuai (sama) dengan ID. File disimpan dalam suatu direktori yang ditunjukkan oleh session.save\_path dalam file php.ini.
3. PHP melempar ID session ke setiap halaman.
4. PHP mengambil nilai session dari file session untuk setiap halaman session.

Beberapa pengaturan session antara lain :



**session.name**

Pemberian nama session terdiri dari karakter alphanumeric, nama standarnya adalah PHPSESSID.

- **session.auto\_start**

Penggunaan session harus diawali dengan *session\_start()*, jika **session.auto\_start** bernilai 1 maka secara otomatis *session\_start()* akan dijalankan saat start up atau komputer dinyalakan, jika bernilai 0 maka session harus diaktifkan secara manual, sehingga *session\_start()* harus dideklarasikan terlebih dahulu.

- **session.cookie\_lifetime**

Untuk menentukan umur atau durasi session, jika bernilai 0, maka session akan dihapus secara otomatis saat keluar dari browser internet.

- **session.cookie\_path**

Untuk menentukan letak path file-file session\_cookie.

Fungsi built-in PHP yang berhubungan dengan session, antara lain :

**SESSION\_START ()**

Agar bisa menggunakan fungsi-fungsi session, maka disetiap halaman website yang mengandung fungsifungsi session harus diawali dengan *session\_start()*.

```
<?php
    session_start();
?>
```

**SESSION\_DESTROY ()**

*session\_destroy()* berguna untuk menghapus dan mengakhiri session, sekaligus menghapus semua elemen yang ada. Jika browser telah ditutup maka secara otomatis session akan diakhiri walaupun fungsi *session\_destroy()* tidak dituliskan secara eksplisit.

```
<?php
session_start();
session_destroy();
?>
```

### **SESSION\_UNSET()**

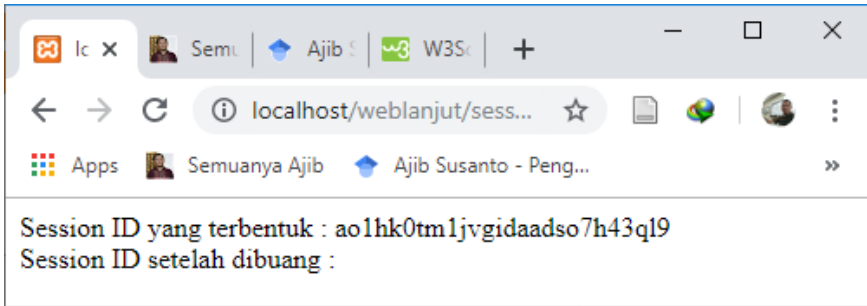
Untuk menghapus elemen-elemen dari session, tetapi tanpa membuang atau mengakhiri session itu sendiri maka bisa digunakan fungsi `session_unset()`.

```
<?php
session_start();
session_unset();
?>
```

### **SESSION\_ID()**

Untuk mendapatkan atau memberi nilai id pada session, dimana setiap kali pengunjung membuka website maka akan diberikan identifikasi session yang bersifat unik atau berbeda disetiap session yang telah tercipta.

```
<?php
session_start();
echo "Session ID yang terbentuk : ".
session_id() . "<br>";
session_destroy();
echo "Session ID setelah dibuang : " .
session_id() . "<br>";
?>
```



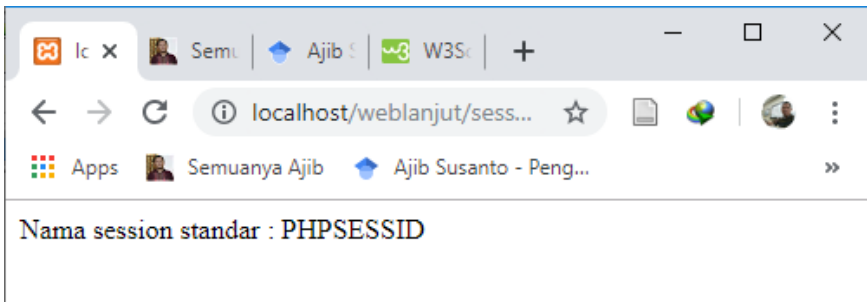
Gambar 4.3 : Session ID

**SESSION\_NAME ()**

Fungsi `session_name()` adalah untuk memperoleh atau memberi nilai terhadap nama sebuah session.

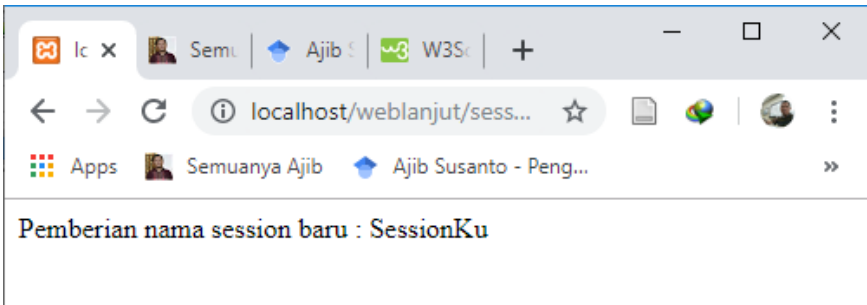
Nama session secara standar adalah PHPSESSID. Jika diinginkan nama lain bisa dituliskan sebagai berikut :

```
<?php
session_start();
// menampilkan nama session standar
echo "Nama session standar : ".session_name().
"<br>";
?>
```



Gambar 4.4 : Nama Session standart

```
<?php
// memberi nama session sendiri
session_name("SessionKu");
session_start();
echo "Pemberian nama session baru : " .
session_name();
?>
```



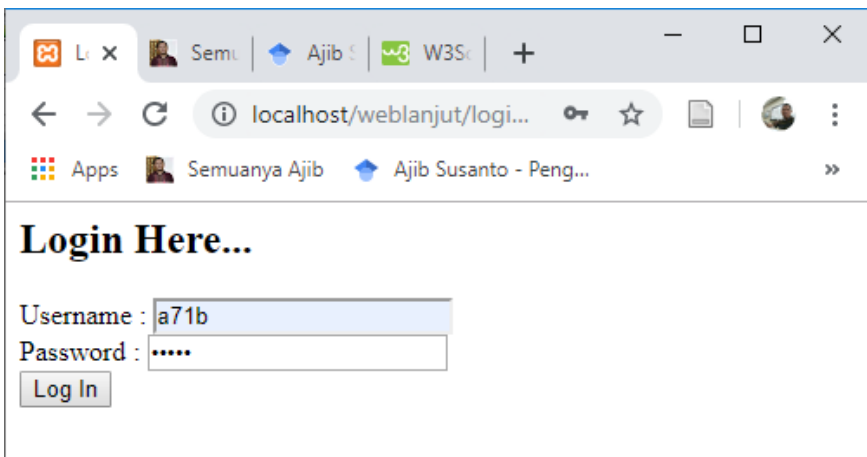
Gambar 4.5 : Nama Session baru

Contoh penerapan session :

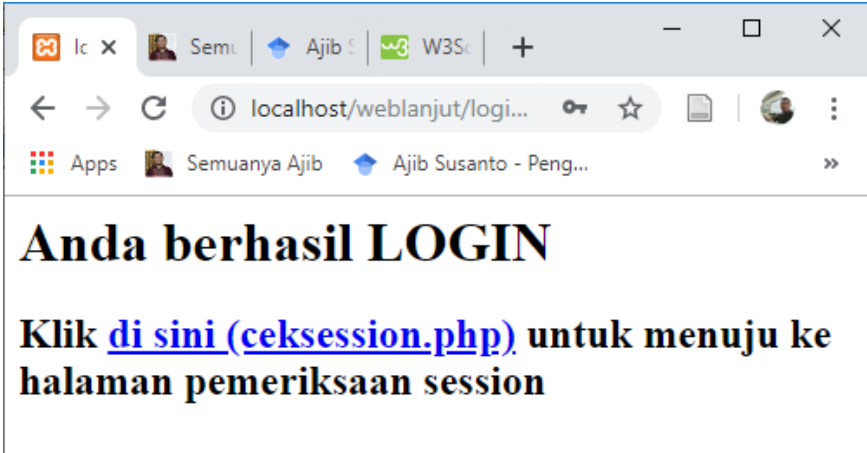
File : loginsession.php

```
<?php
session_start();
if (isset ($_POST['Login'])) {
$user = $_POST['user'];
$pass = $_POST['pass'];
//periksa login
if ($user == "a71b" && $pass = "12345") {
//menciptakan session
$_SESSION['login'] = $user;
//menuju ke halaman pemeriksaan session
echo "<h1>Anda berhasil LOGIN</h1>";
echo "<h2>Klik <a href='ceksession.php'>di sini
(ceksession.php)</a> untuk menuju ke halaman
pemeriksaan session";
}
}
```

```
} else {  
?>  
<html>  
<head>  
<title>Login here...</title>  
</head>  
<body>  
<form action="" method="post">  
<h2>Login Here...</h2>  
Username : <input type="text" name="user"><br>  
Password : <input type="password"  
name="pass"><br>  
<input type="submit" name="Login" value="Log  
In">  
</form>  
</body>  
</html>  
<?php } ?>
```



Gambar 4.6 : Halaman Login



Gambar 4.7 : Halaman setelah Login

File : ceksession.php

```
<?php
session_start();
//pemeriksaan session
if (isset($_SESSION['login'])) {
//jika sudah login
//menampilkan isi session
echo "<h1>Selamat Datang ". $_SESSION['login'].
"</h1>";
echo "<h2>Halaman ini hanya bisa diakses jika
Anda sudah login</h2>";
echo "<h2>Klik <a href='logout.php'>di
sini(logout.php)</a> untuk LOGOUT</h2>";
} else {
//session belum ada artinya belum login
die ("Anda belum login! Anda tidak berhak masuk
ke halaman ini.Silahkan login <a
href='loginsession.php'>di sini</a>");
}
?>
```



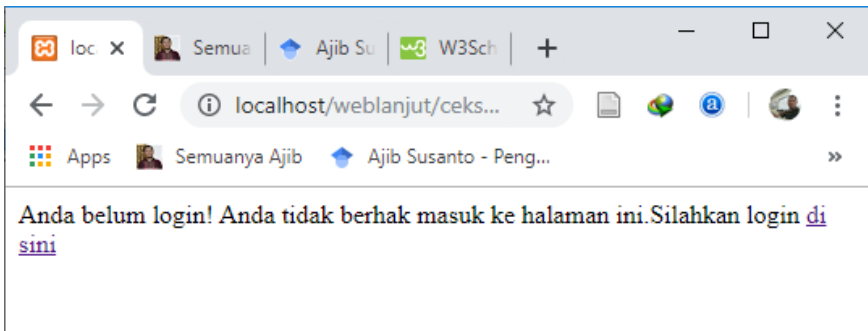
Gambar 4.7 : Halaman berhasil login

File : logout.php

```
<?php
session_start();
if (isset($_SESSION['login'])) {
unset ($_SESSION);
session_destroy();
//
echo "<h1>Anda sudah berhasil LOGOUT</h1>";
echo "<h2>Klik <a href='loginsession.php'>di
sini</a> untuk LOGIN kembali</h2>";
echo "<h2>Anda sekarang tidak bisa masuk ke
halaman <a href='ceksession.php'>ceksession.php
</a> lagi</h2>";
}
?>
```



Gambar 4.8 : Halaman logout



Gambar 4.9 : Halaman jika belum login

## Cookies

Seperti halnya session, cookies juga merupakan sebuah konsep penyimpanan informasi user. Hanya saja, jika session tempat penyimpanan berada di server, cookies berada di client. Oleh karena itu, konsep cookies sebaiknya jangan digunakan untuk menyimpan informasi login user seperti username, password dsb. Selain user bias melihat informasi yang disimpan, user juga bisa men-disable cookies itu



sendiri. Jika cookies di-disable, maka program yang memanfaatkan cookies tentunya tidak akan berjalan dengan baik.

Cookies sendiri biasanya dipakai dalam aplikasi shopping cart. Biasa digunakan untuk menyimpan sementara, produk-produk yang dipilih oleh pengunjung pada saat berbelanja.

Dalam penanganan cookies juga terdapat beberapa proses yang perlu diperhatikan :

- Proses pembuatan cookies
- Proses pemeriksaan cookies
- Proses penghapusan cookies

Contoh Cookies :

File : cookie1.php

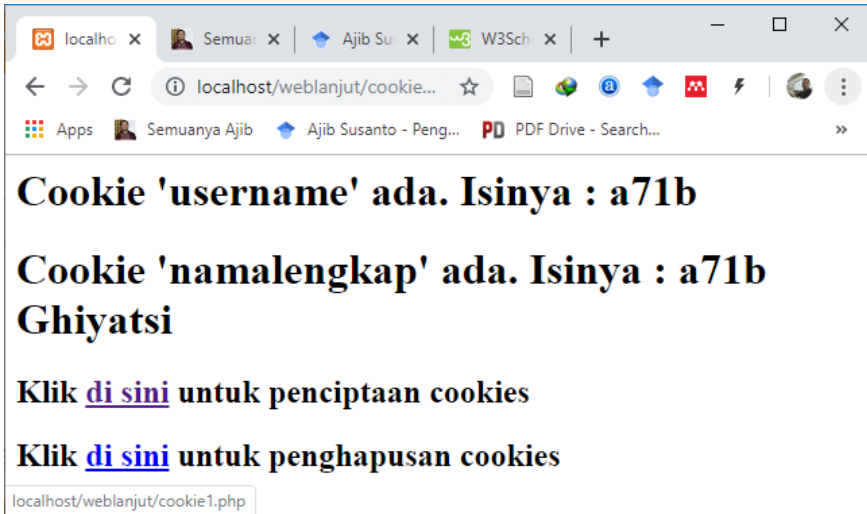
```
<?php
$value = 'a71b';
$value2 = 'a71b Ghyatsi';
setcookie("username", $value);
setcookie("namalengkap", $value2, time()+3600);
/* expire in 1 hour */
echo "<h1>Ini halaman pengesetan cookie</h1>";
echo " <h2>Klik <a href='cookie2.php'>di
sini</a> untuk pemeriksaan cookies</h2>";
?>
```



Gambar 4.10 : Halaman pengesetan cookies

File : cookie2.php

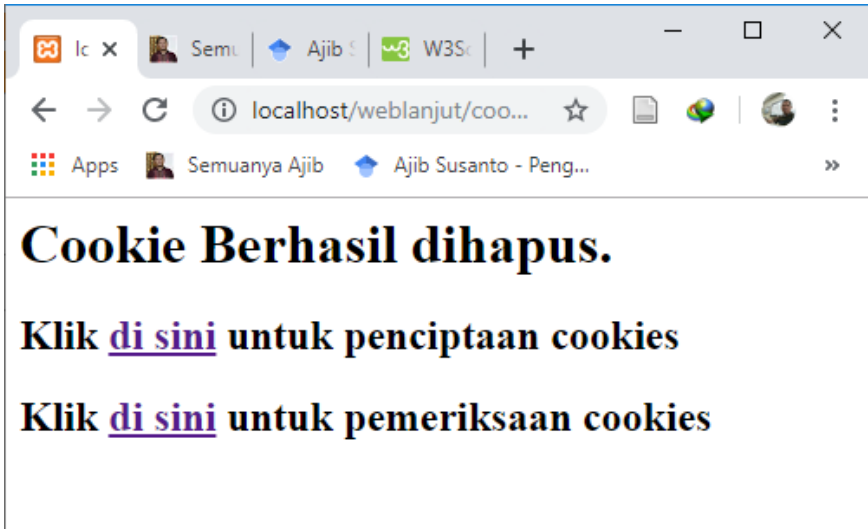
```
<?php
if(isset($_COOKIE['username'])) {
echo "<h1>Cookie 'username' ada. Isinya : " .
$_COOKIE['username'];
} else {
echo "<h1>Cookie 'username' TIDAK ada.</h1>";
}
if(isset($_COOKIE['namalengkap'])) {
echo "<h1>Cookie 'namalengkap' ada. Isinya : "
.$_COOKIE['namalengkap'];
} else {
echo "<h1>Cookie 'namalengkap' TIDAKada.</h1>";
}
echo "<h2>Klik <a href='cookie1.php'>di
sini</a> untuk penciptaan cookies</h2>";
echo "<h2>Klik <a href='cookie3.php'>di
sini</a> untuk penghapusan cookies</h2>";
?>
```



Gambar 4.11 : Halaman hasil cookies

File : cookie3.php

```
<?php
// set the expiration date to one hour ago
setcookie ("username", "", time() - 3600);
setcookie ("namalengkap", "", time() - 3600);
echo "<h1>Cookie Berhasil dihapus.</h1>";
echo " <h2>Klik <a href='cookie1.php'>di
sini</a> untuk penciptaan cookies</h2>";
echo " <h2>Klik <a href='cookie2.php'>di
sini</a> untuk pemeriksaan cookies</h2>";
?>
```



Gambar 4.12 : Halaman penghapusan cookies

## Latihan

### Pertemuan 3 : Form, Session, Cookies

1. Buat program berikut ini :

#### Hitung Nilai

Nilai Tugas (30%) :   
 Nilai UTS (35%) :   
 Nilai UAS (35%) :   
 Nilai Akhir :   
 Nilai Huruf :   
 Predikat :

#### Kalkulator Mini

Nilai 1 :   
 Nilai 2 :   
 Hasil :

2. Perhatikan contoh cookie dari <http://w3schools.com> di bawah ini :

#### Example

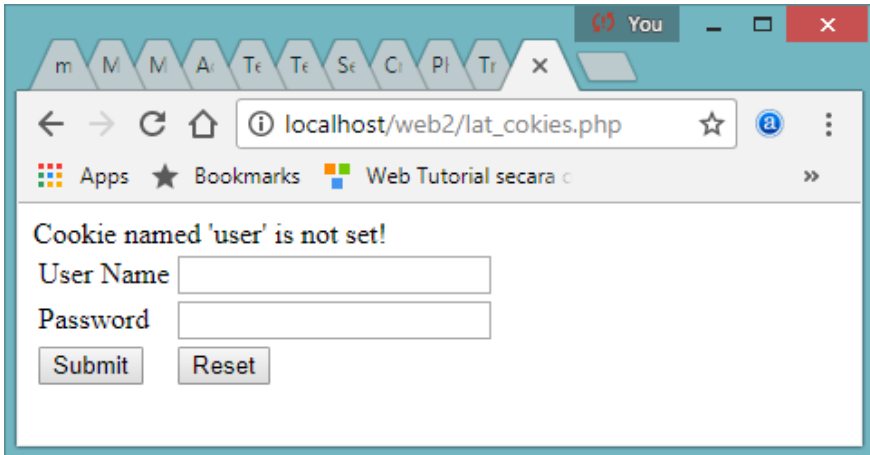
```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

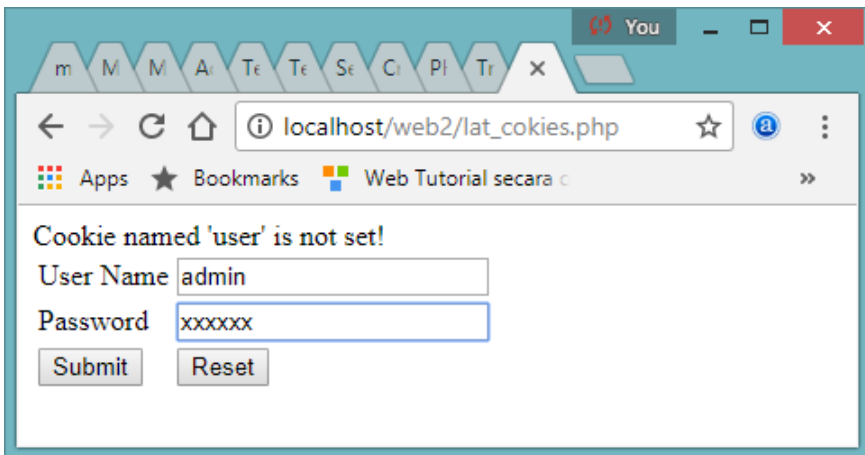
</body>
</html>
```

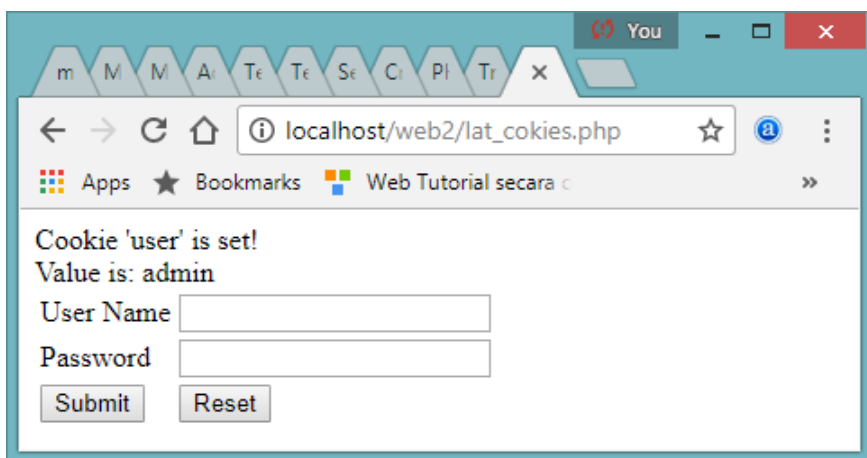
Ketik ulang dan jalankan, perhatikan hasilnya!

Buat program login sederhana untuk mengecek user sudah pernah login atau tidak menggunakan cookies, set waktu 1 menit user tetap dikenali browser.

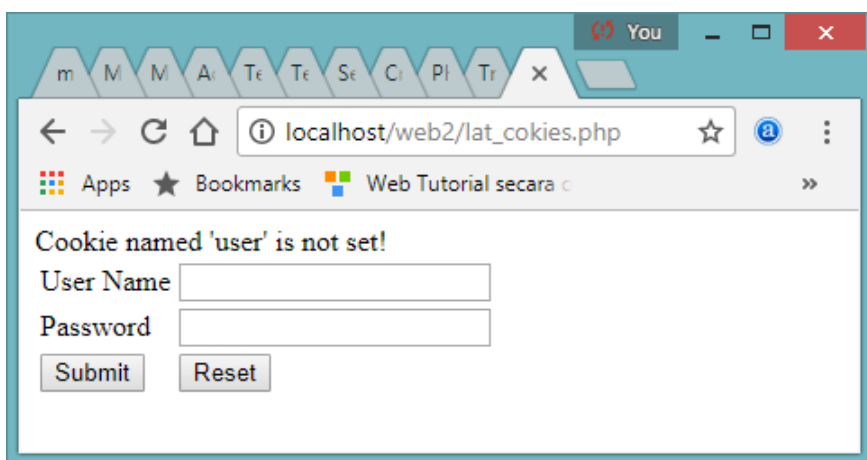


Masukkan User dan submit

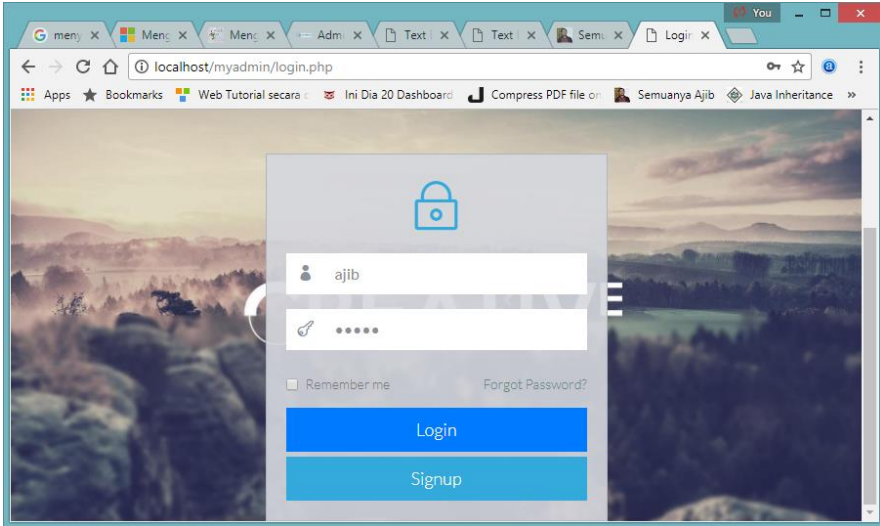




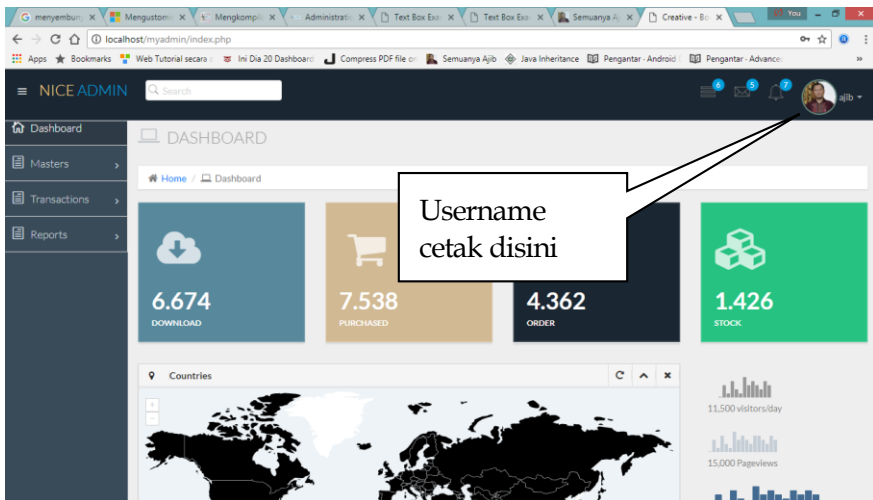
Cek kembali setelah 1 menit :



3. Buat program login untuk masuk ke sebuah sistem seperti gambar berikut ini :



Jika username dan password cocok akan menampilkan halaman utama sistem, halaman utama bebas, boleh menggunakan css template



Ket :gunakan session untuk menyimpan username



---

# Bab 5

## FILE & DIREKTORI

---

Bab ini membahas:

- File
  - Direktori
- 

### File

Dalam management file dan direktori, PHP menyediakan lebih dari fungsi. Beberapa fungsi utama yang berhubungan dengan management file(create, write, append, dan delete), antara lain :

Membuka dan Membuat File

```
fopen ($namafile, $mode);
```

*Keterangan :*

\$namafile merupakan nama file yang akan dibuat, sedangkan

\$mode merupakan mode akses file. Mode akses file yang bisa digunakan yaitu :

Mode	Keterangan
r	Hanya untuk baca file, pointer berada di awal file
r+	Untuk baca dan tulis file, pointer berada di awal file
w	Hanya untuk tulis file, isi file lama dihapus, jika file belum ada maka akan di-create
w+	Untuk baca dan tulis file, isi file lama dihapus, jika file belum ada maka akan di-create
a	Hanya untuk menambahkan isi file, pointer berada di akhir file, jika file belum ada maka di-create
a+	Untuk membaca dan menambahkan isi file, pointer berada di akhir file, jika file belum ada maka di-create

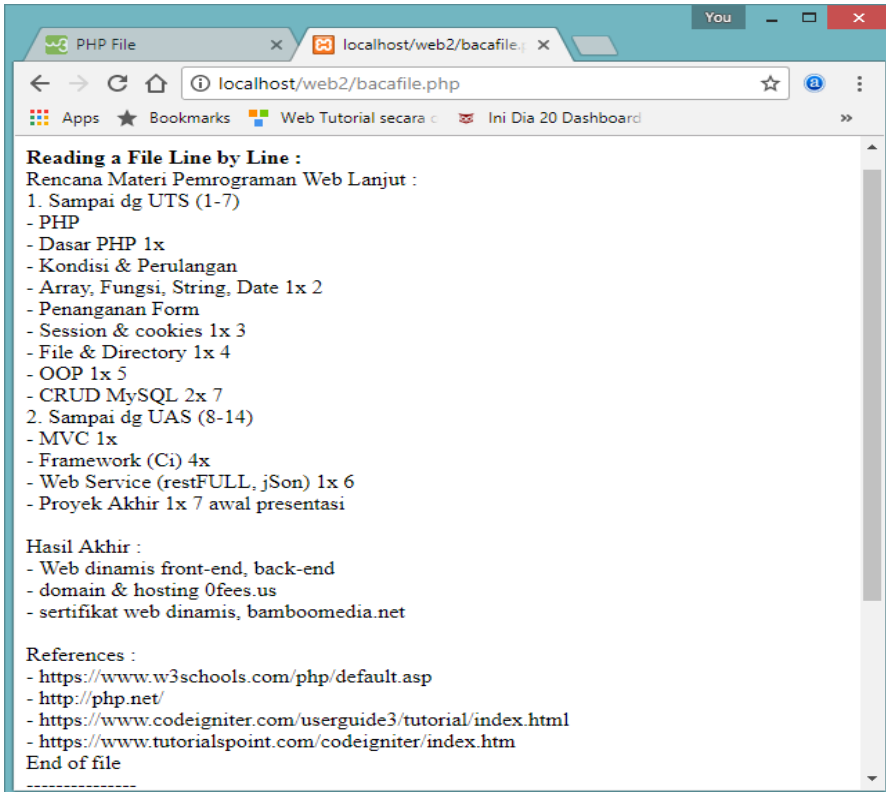
## Direktori

## Latihan

### Pertemuan 4 : File, Directory, File Uploads

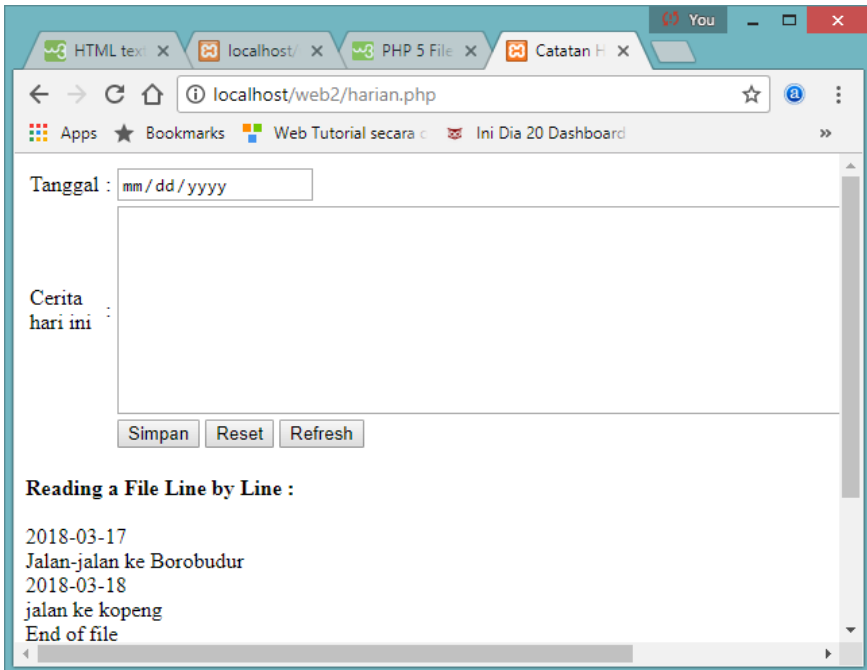
1. Buat program berikut ini :

```
<!DOCTYPE html>
<html>
<body>
<?php
$file=fopen("materiweblanjut.txt","r")           or
exit("Unable to open file!");
//Output a line of the file until the end is
reached
echo "<br><b>Reading a File Line by Line :</b>
<br>";
while(!feof($file))
{
    echo fgets($file). "<br>";
}
if (feof($file)) echo "End of file";
fclose($file);
?>
</body>
</html>
```



Buat untuk membaca file tiap character, perhatikan perbedaannya!

2. Perhatikan contoh File yang di <http://w3schools.com>, buat program seperti gambar di bawah ini :



Keterangan :

- Dapat menyimpan dan menambahkan data baru

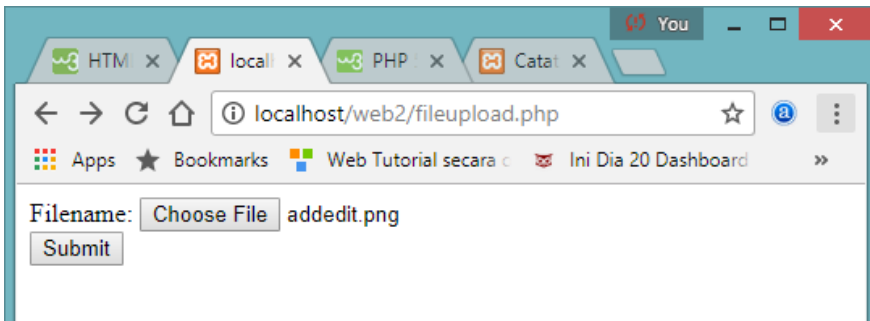
3. Buat program upload file seperti gambar berikut ini :

```

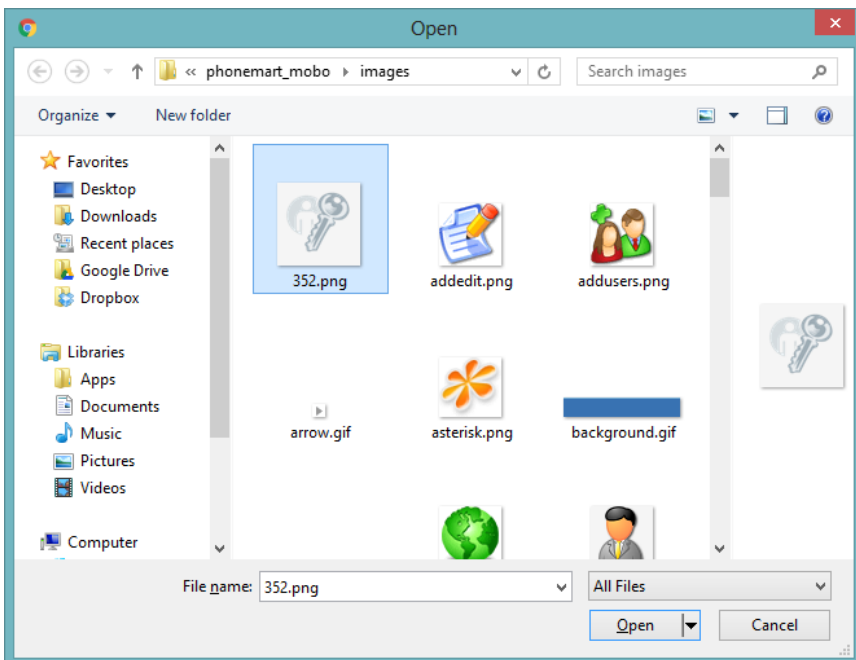
2 <body>
3
4 <form action="upload_file.php" method="post"
5 enctype="multipart/form-data">
6 <label for="file">Filename:</label>
7 <input type="file" name="file" id="file"><br>
8 <input type="submit" name="submit" value="Submit">
9 </form>
10
11 </body>
12 </html>

```

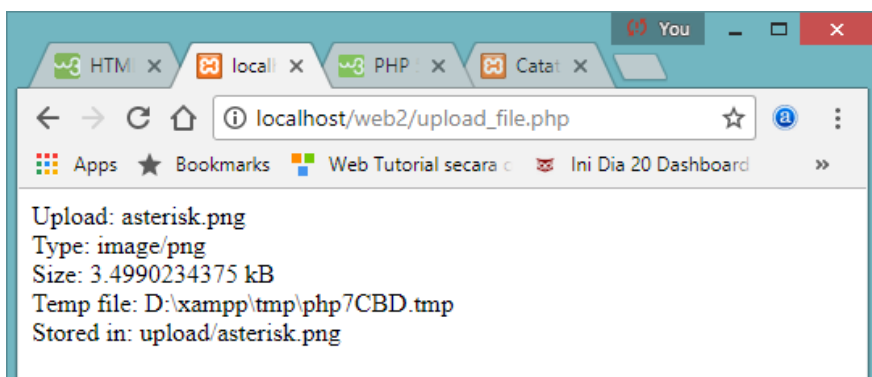
Hasil seperti berikut :



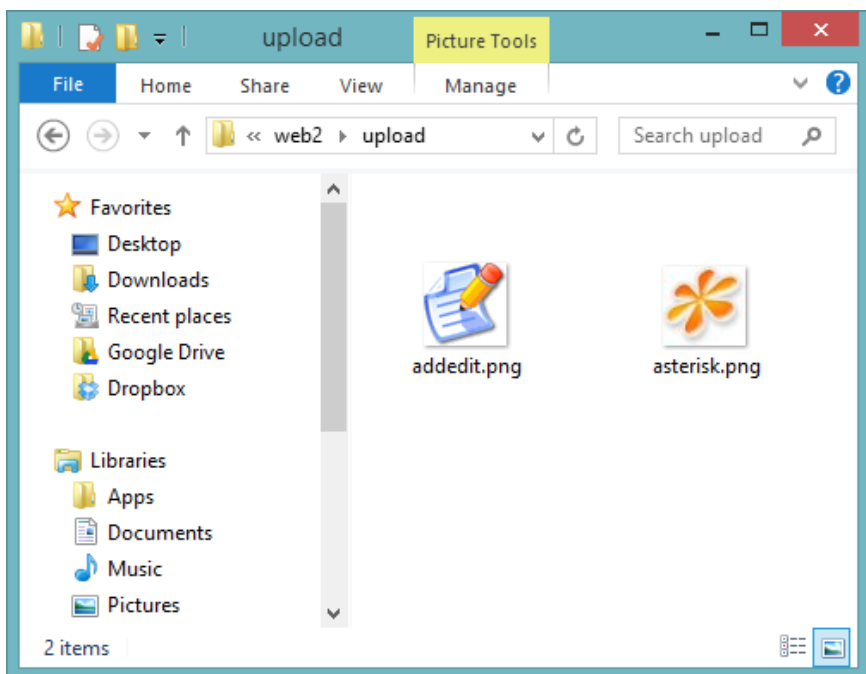
Jika klik choose file :



Jika klik Submit :



Hasil di folder upload :



Kembangkan contoh di atas untuk membuat form Data Diri Mahasiswa yang dapat meng-*upload* foto mahasiswa. Minimal inputan NIM, NAMA dan Upload Foto.

# Bab 6

## OBJECT ORIENTED PROGRAMMING DI PHP

---

**Bab ini membahas:**

- Pemrograman Berorientasi Obyek
  - Class & Object
  - Properties & Method
  - Turunan Class
- 

## Pemrograman Berorientasi Obyek

PHP pada awalnya hanyalah kumpulan script sederhana. Dalam perkembangannya, selanjutnya ditambahkan berbagai fitur pemrograman berorientasi objek. Hal ini dimulai sejak PHP 4. Dengan lahirnya PHP 5, fitur-fitur pemrograman berorientasi objek semakin mantap dan semakin cepat. Dengan PHP 5, script yang menggunakan konsep *object-oriented* akan lebih cepat dan lebih efisien.

Pemrograman berorientasi objek atau *object-oriented programming* (OOP) merupakan suatu pendekatan pemrograman yang menggunakan object dan class. Saat ini konsep OOP sudah semakin berkembang. Hampir setiap perguruan tinggi di dunia mengajarkan konsep OOP ini pada mahasiswanya. Pemrograman yang banyak dipakai dalam penerapan konsep OOP adalah Java dan C++.

OOP bukanlah sekedar cara penulisan sintaks program yang berbeda, namun lebih dari itu, OOP merupakan cara pandang dalam menganalisa system dan permasalahan pemrograman. Dalam OOP, setiap bagian



---

dari program adalah *object*. Sebuah *object* mewakili suatu bagian program yang akan diselesaikan.

Beberapa konsep OOP dasar, antara lain :

1. *Encapsulation* (Class dan *Object*)
2. *Inheritance* (Penurunan sifat), dan
3. *Polymorphisme*

## Class & Object

Bagian dasar dari sebuah program yang berorientasi objek adalah **objects**. Secara mudah kita dapat memahami mengenai *object* ini. Sebagai contoh, sebuah **mobil** adalah objek. Sebuah mobil mempunyai *properties* atau bagianbagian di dalamnya, seperti warna, mesin, roda, pintu dsb. Sebuah mobil juga dapat melakukan sesuatu (ada sesuatu yang bisa dilakukan dengan mobil), seperti mengisi bensin, menyalakan mesin, berjalan, mengerem dsb.

Biasanya *object* adalah sebuah kata benda. **Orang** adalah *object*. Demikianjuga mobil, pohon, bunga, komputer, TV, buku dsb. Namun, *object* tidak selamanya sebuah objek fisik. Bisa saja sebuah benda abstrak, seperti *account* bank, sebuah file di komputer, database, pesan email, acara TV, dsb.

**Class** merupakan penjelasan atau deskripsi dari *object*. Di dalam *class*, terdapat penjelasan tentang suatu *object* termasuk *properties* yang dimilikinya serta kelakuan atau *method* yang bisa dilakukan oleh *object*. Sebagai contoh, *class* **Orang**. *Class* Orang tentu setidaknya memiliki beberapa bagian seperti tangan, kaki, mata, telinga dsb. *Class* Orang juga setidaknya harus bisa jalan, bisa loncat, bisa lari, bisa melihat, bisa bicara dsb. Salah satu keuntungan program didefinisikan dengan konsep OOP adalah adanya pengkapsulan (*encapsulation*) program dalam *class* dan *object*, dimana programmer yang menggunakan *class* tidak perlu mengetahui isi dan jalannya *class* secara detail, hanya perlu tahu bagaimana cara menggunakannya. Sama halnya dengan sebuah **mobil** misalnya. Seorang pemilik mobil tentunya tidak perlu mengetahui bagian-bagian mobil secara menyeluruh. Dia tidak perlu mengetahui bagaimana mesin mobil melakukan pembakaran dan bagaimana mesin mobil bisa menggerakkan roda, dsb. Dia

hanya perlu tahu bagaimana cara menjalankan mobil, bagaimana menghentikan mobil, dan fungsi mobil lainnya.

## Properties & Method

Setiap class memiliki *properties* yang kadang disebut juga *attributes*. *Properties* dari sebuah mobil misalnya warna, ukuran, harga dsb. Di dalam class, *properties* dinyatakan dengan sebuah variabel. Misalnya \$warna, \$harga, dsb.

*Method* merupakan sesuatu yang bisa dilakukan oleh *object*. Method dalam PHP sama artinya dengan sebuah fungsi. Method yang mungkin dipunyai dari sebuah mobil misalnya, method untuk menghidupkan mobil, menjalankan mobil, menghentikan mobil, dsb.

Penamaan *properties* dan *method* memiliki aturan yang sama dengan penamaan sebuah variabel atau fungsi. Akan tetapi berdasarkan kesepakatan (*convention*), penamaan *properties* dan *method* harus menggunakan *camel Caps*, dimana tiap kata diawali dengan huruf besar kecuali kata pertama, setiap kata digabung tanpa spasi atau *under-score* (\_).

## Turunan Class

## Latihan

### Pertemuan 5 : OOP di PHP

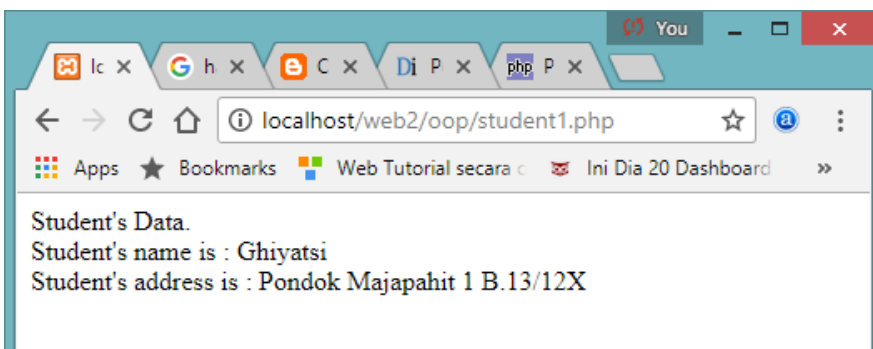
1. Buat 2 program berikut ini :

```
<?php
class student {
public $name='';
public $address='';

public function __construct()
{
```

```
        echo "Student's Data.<br />";
    }
    public function show_name()
    {echo "Student's name is : ".$this->name."<br/>";
    }
    public function show_address()
    {echo "Student's address is : ".$this->
    >address."<br/>";
    }
    }
    }
    ?>
```

```
<?php
include "student.php";
$student1 = new student;
//this create new object in the class student named
object student1
//assigning data into properties
$student1->name = "Ghiyatsi";
$student1->address = "Pondok Majapahit 1 B.13/12X";
//call the method
$student1->show_name();
$student1->show_address();
?>
```



Jalankan dan perhatikan hasilnya!

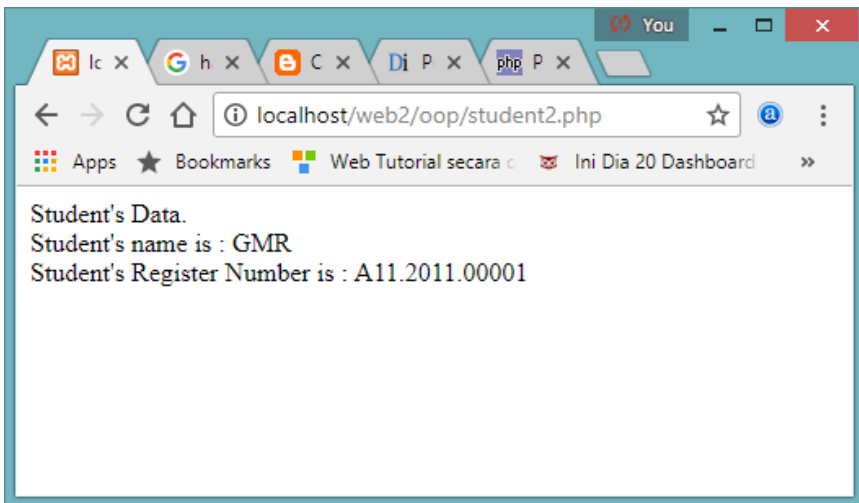
Tambahkan program di bawah ini yang merupakan turunan dari kelas sebelumnya!

```
<?php
include("student.php");
class newstudent extends student {
private $register_number;

public function set_register_number($reg)
{
    $this->register_number=$reg;
}
public function show_register_number () {
    echo "Student's Register Number is : ".$this-
>register_number."<br/>";
}
}
?>
```

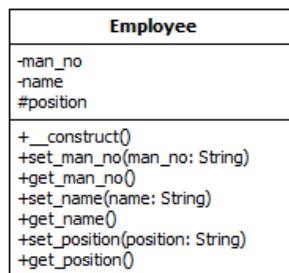
```
<?php
include('newstudent.php');
$student2 = new newstudent;
$student2->name = 'GMR';
$student2->address = 'PM 1';
$student2->set_register_number('A11.2011.00001');

//call the method
$student2->show_name();
$student2->show_register_number();
?>
```



2. Perhatikan contoh Class Diagram di bawah ini :

### Class Diagram Example



### Class Diagram Key

- The **Upper box** contains the class name
- The **middle box** contains the class variables
- The **lower box** contains the class methods
- The **minus (-)** sign means private scope
- The **plus (+)** sign means public scope
- The **hash (#)** sign means protected scope

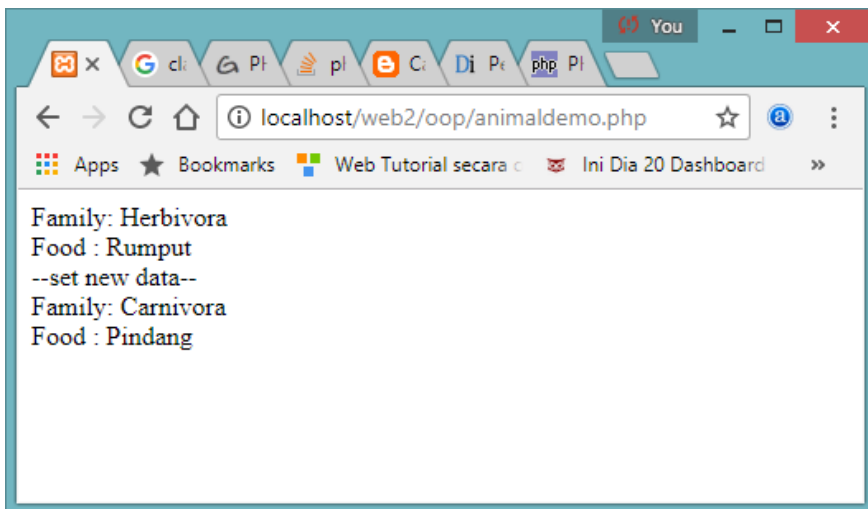
Implementasikan class diagram di atas seperti code berikut :

```
1 <?php
2 class Animal
3 {   private $family;
4     private $food;
5     public function __construct($family, $food)
6     {   $this->family = $family;
7         $this->food  = $food;
8     }
9     public function get_family()
10    {   return $this->family;}
11    public function set_family($family)
12    {   $this->family = $family;}
13    public function get_food()
14    {   return $this->food;}
15    public function set_food($food)
16    {   $this->food = $food;}
17 }
18 ?>
```

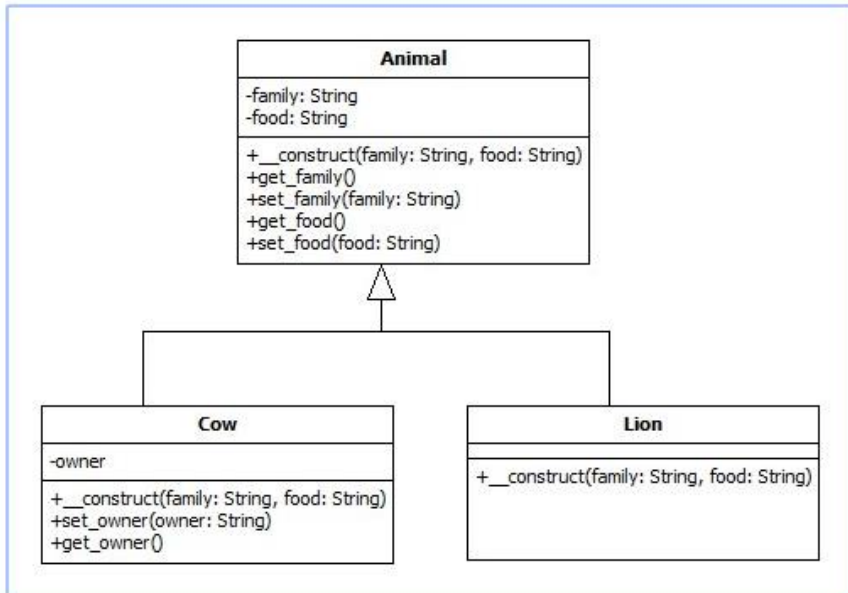
Buat program untuk mendemokan class di atas seperti berikut :

```
1 <?php
2 include "animal.php";
3 $hewan=new Animal('Herbivora', 'Rumput');
4 echo "Family: ".$hewan->get_family()."<br>";
5 echo "Food  : ".$hewan->get_food()."<br>";
6 echo "--set new data--<br>";
7 $hewan->set_family('Carnivora');
8 $hewan->set_food('Pindang');
9 echo "Family: ".$hewan->get_family()."<br>";
10 echo "Food  : ".$hewan->get_food()."<br>";
11 ?>
```

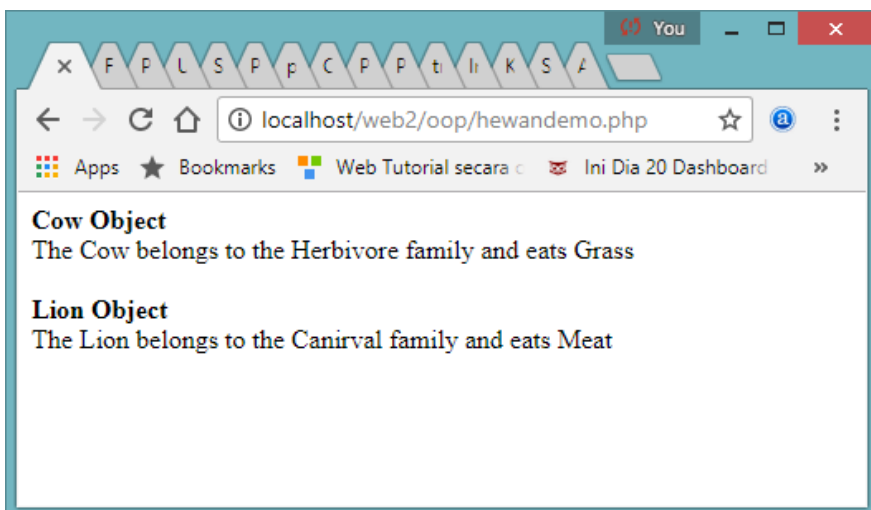
Jalankan dan perhatikan hasilnya !



3. Perhatikan class diagram berikut ini dan implementasikan ke code PHP :

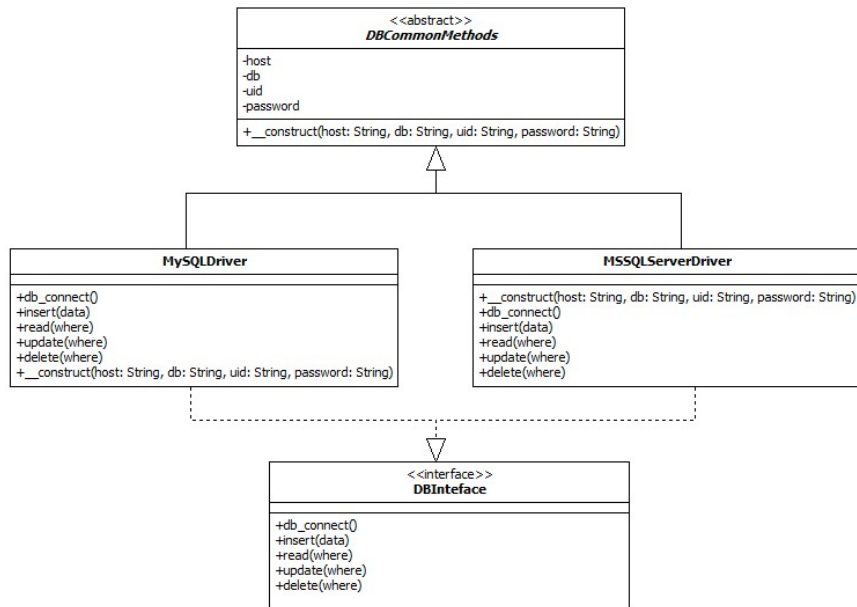


Hasil seperti berikut :





4. Perhatikan class diagram berikut ini dan implementasikan ke code PHP :



# Bab 7

## PHP & DATABASE

---

Bab ini membahas:

- Aplikasi PHPMyAdmin
  - Database MySQL
  - Koneksi Database
  - Aplikasi CRUD (Create, Read, Update, Delete)
- 

### **Aplikasi PHPMyAdmin**

### **Database MySQL**

### **Koneksi Database**

### **Aplikasi CRUD**

## **Latihan**

**Pertemuan 6 : CRUD (Create Read Update Delete) di PHP**

1. Buat program koneksi ke DB berikut ini :

---

*//procedural dengan mysqli*

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "penjualan";
// Create connection
$conn = mysqli_connect($servername, $username,
$password, $dbname);
// Check connection
if (!$conn)
{
    die("Failed to connect to MySQL: " .
mysqli_connect_error());
}
echo "Connected successfully";
?>
```

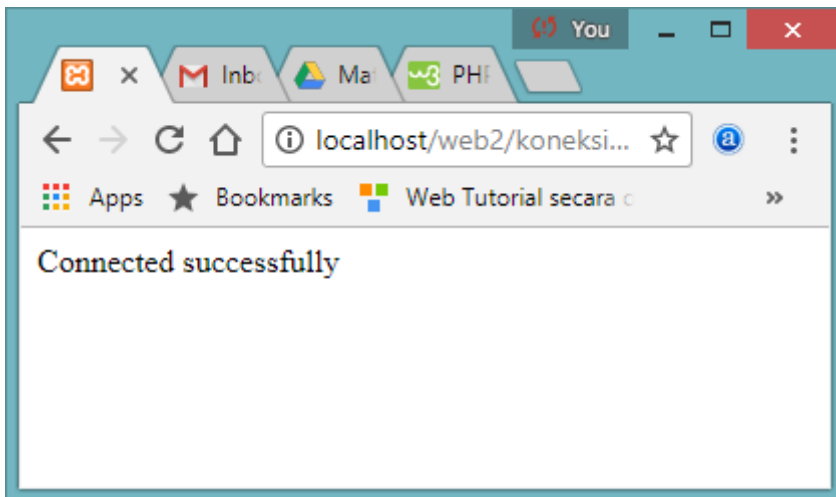
*//oop dengan mysqli*

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "penjualan";
// Create connection
$conn = new mysqli($servername, $username,
$password,$dbname);
// Check connection
if($conn->connect_error)
{
    die("Failed to connect to MySQL: " .
mysqli_connect_error());
}
echo "Connected successfully";
?>
```

*//OOP dengan PDO*

```
<?php
$servername = "localhost";
```

```
$username = "root";
$password = "";
try {
    $conn = new
PDO("mysql:host=$servername;dbname=penjualan",
$username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
    echo "Connected successfully";
}
catch(PDOException $e)
{
    echo "Connection failed: " . $e->getMessage();
}
?>
```



Jalankan dan perhatikan hasilnya!

2. Modifikasi program koneksi di atas menjadi class KoneksiDB dengan konsep OOP.
3. Buat database Penjualan, buat table barang dengan struktur sebagai berikut :

Field Name	Datatype	Len	Default	FK?	Not Null?	Unsigned?	Auto Incr?	Zerofil
kd_brg	char	6		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
nm_brg	varchar	30		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
satuan	varchar	10		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
harga	double			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
harga_beli	double			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
stok	int	4		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
stok_min	int	4		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Boleh ditambahkan satu field untuk menampung gambar produk.

4. Buat program menampilkan data berikut ini :

```

<?php
    include "koneksi_ip.php"
?>
<HTML>
<HEAD>
<TITLE>Menampilkan Daftar Barang</TITLE>

<script language="javascript">
function tanya() {
if (confirm ("Apakah Anda yakin akan menghapus
    barang ini ?")) {
return true;
} else {
return false;
}
}
</script>
</HEAD>

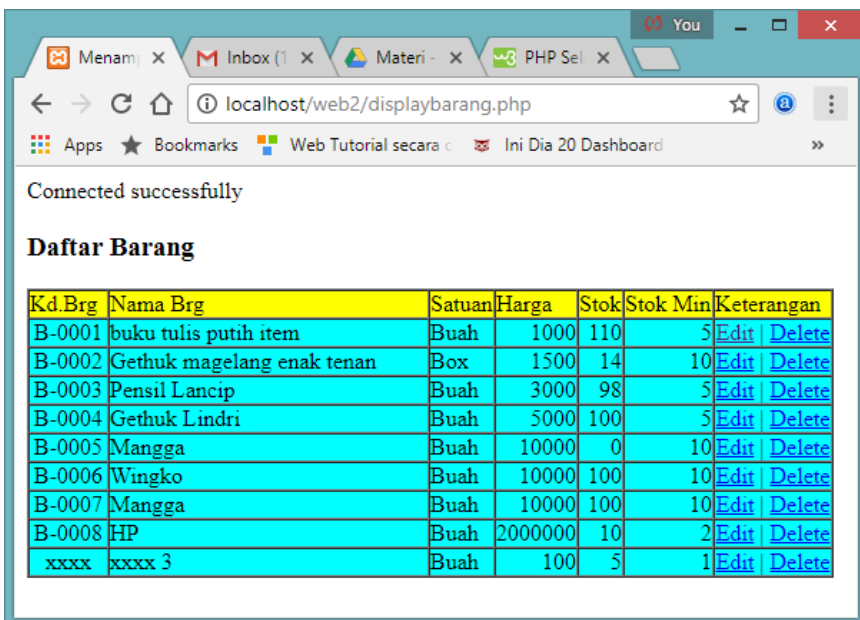
```

```

<BODY>
<?php
    $query = "SELECT * FROM barang";
    $sql = mysqli_query ($conn,$query);
    echo "<h3>Daftar Barang</h3>";
    echo "    <table border='1' cellpadding='0'
    cellpadding='0'>";
    echo "        <tr
        bgcolor='yellow'><td
        width=10%>Kd.Brg</td><td
        width=40%>Nama
        Brg</td><td>Satuan</td><td>Harga</td><td>Stok</td
        ><td>Stok Min</td><td>Keterangan</td></tr>";
    while ($hasil = mysqli_fetch_array ($sql)) {
        $kode = $hasil['kd_brg'];
        $nama = stripslashes ($hasil['nm_brg']);
        $satuan = stripslashes ($hasil['satuan']);
        $harga = $hasil['harga'];
        $stok= $hasil['stok'];
        $stok_min = $hasil['stok_min'];
        //tampilkan barang
        echo "    <tr bgcolor='cyan'>
        <td align='center' width=10%>$kode</td>
        <td align='left' width=40%>$nama</td>
        <td align='left'>$satuan</td>
        <td align='right'>$harga</td>
        <td align='right'>$stok</td>
        <td align='right'>$stok_min</td>";
        echo "        <td align='center'><a
        href='edit_barang.php?id=$kode'>Edit</a> | ";
        echo "    <a href='delete_barang.php?id=$kode'
        onClick='return tanya()'>Delete</a></td>";
    }
    echo "</table>";
?>
</BODY>
</HTML>

```

Dengan hasil seperti berikut :



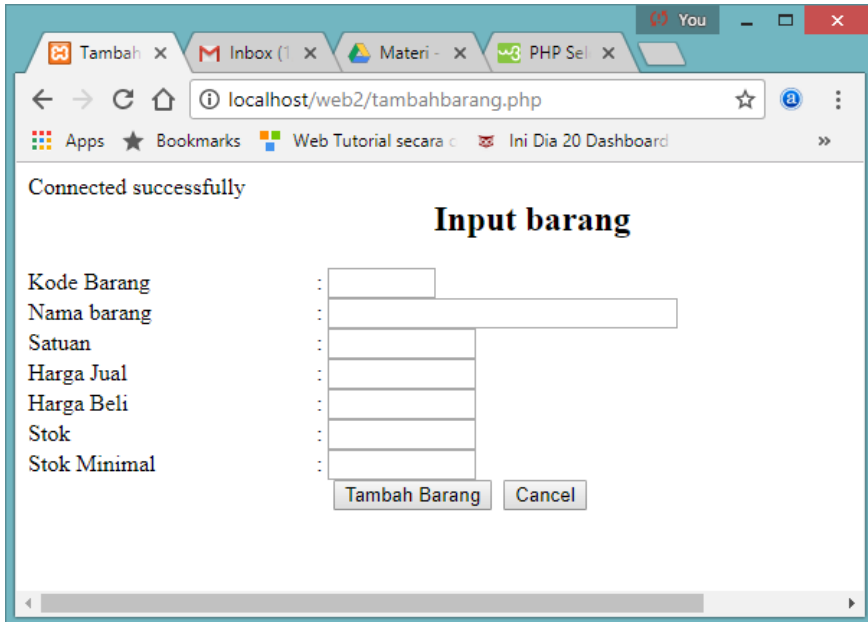
Connected successfully

### Daftar Barang

Kd.Brg	Nama Brg	Satuan	Harga	Stok	Stok Min	Keterangan
B-0001	buku tulis putih item	Buah	1000	110	5	<a href="#">Edit</a> <a href="#">Delete</a>
B-0002	Gethuk magelang enak tenan	Box	1500	14	10	<a href="#">Edit</a> <a href="#">Delete</a>
B-0003	Pensil Lancip	Buah	3000	98	5	<a href="#">Edit</a> <a href="#">Delete</a>
B-0004	Gethuk Lindri	Buah	5000	100	5	<a href="#">Edit</a> <a href="#">Delete</a>
B-0005	Mangga	Buah	10000	0	10	<a href="#">Edit</a> <a href="#">Delete</a>
B-0006	Wingko	Buah	10000	100	10	<a href="#">Edit</a> <a href="#">Delete</a>
B-0007	Mangga	Buah	10000	100	10	<a href="#">Edit</a> <a href="#">Delete</a>
B-0008	HP	Buah	2000000	10	2	<a href="#">Edit</a> <a href="#">Delete</a>
xxxx	xxxx 3	Buah	100	5	1	<a href="#">Edit</a> <a href="#">Delete</a>

Tambahkan tombol **Add** di luar Tabel, posisi bebas, digunakan untuk memanggil form tambah data.

5. Buat form tambahBarang.php untuk menambah barang, dengan tampilan seperti berikut :



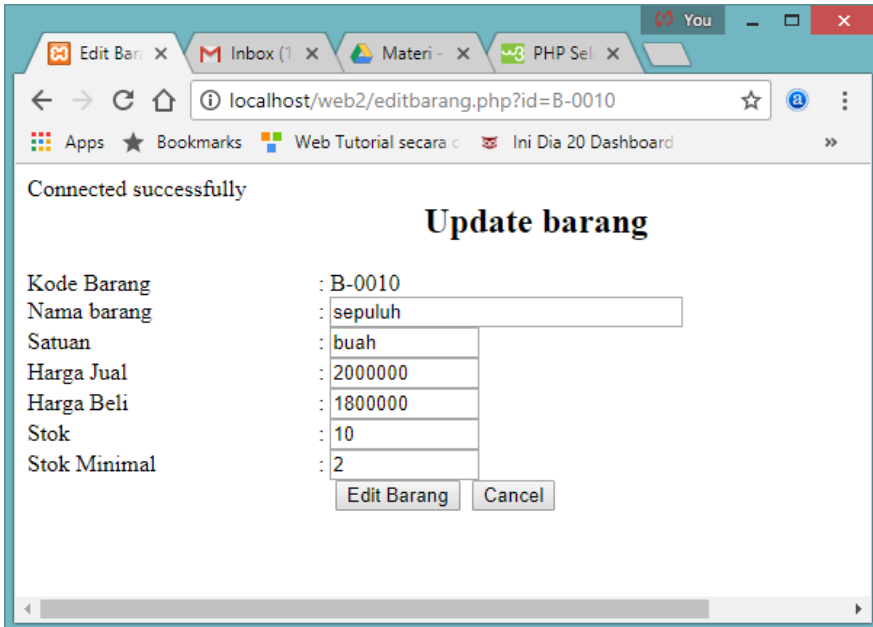
Connected successfully

### Input barang

Kode Barang	:	<input type="text"/>
Nama barang	:	<input type="text"/>
Satuan	:	<input type="text"/>
Harga Jual	:	<input type="text"/>
Harga Beli	:	<input type="text"/>
Stok	:	<input type="text"/>
Stok Minimal	:	<input type="text"/>

6. Buat form `editBarang.php` untuk meng-update barang, dengan tampilan seperti berikut :



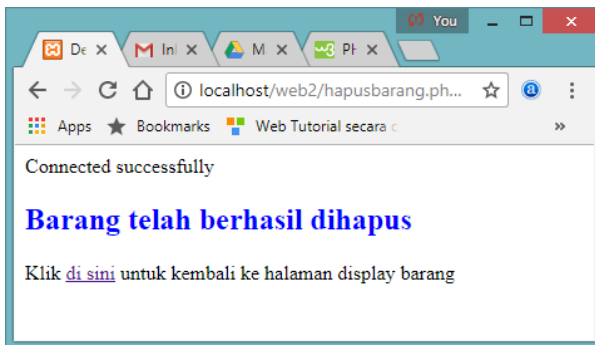


7. Buat program `hapusBarang.php` berikut ini untuk meng-hapus data barang:

```
<?php
include "koneksi_ip.php";
if (isset($_GET['id'])) {
    $kode = $_GET['id'];
} else {
    die ("Error. NO Id Selected! ");
}
?>
<html>
<head><title>Delete Barang</title>
</head>
<body>

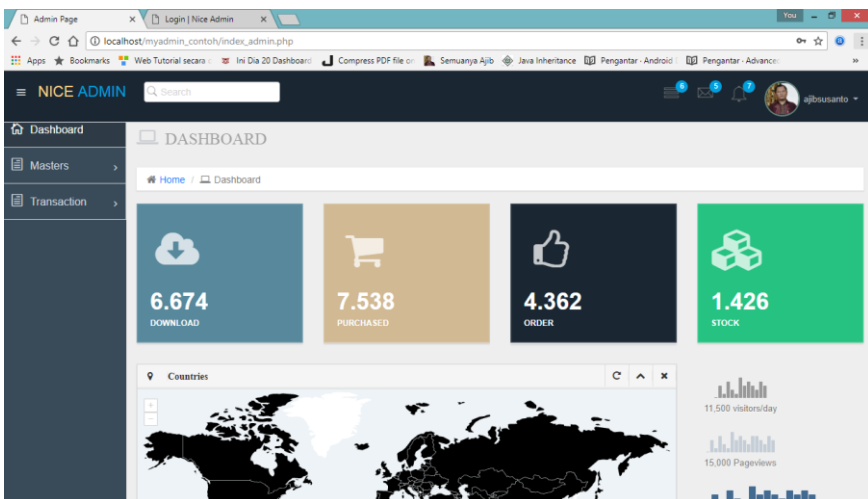
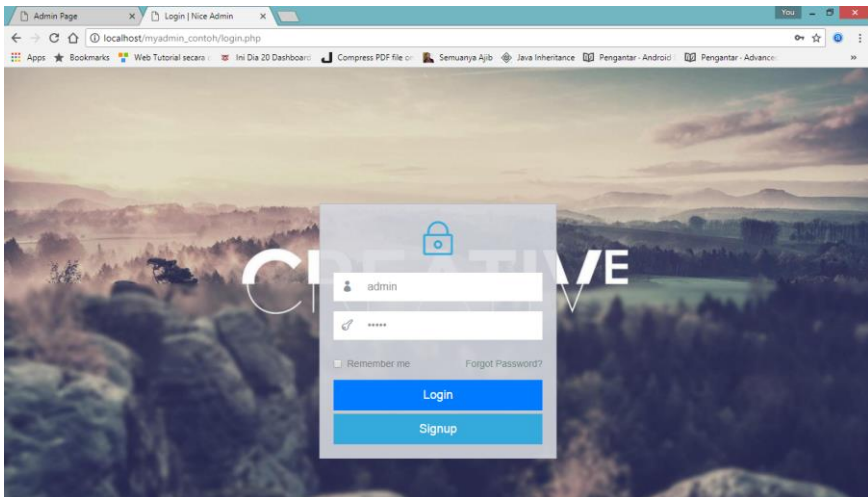
<?php
//proses delete barang
if (!empty($kode) && $kode != "") {
    $query = "DELETE FROM barang WHERE kd_brg='$kode'";
```

```
$sql = mysqli_query ($conn,$query);
if ($sql) {
echo "<h2><font color=blue>Barang telah berhasil
dihapus</font></h2>";
} else {
echo "<h2><font color=red>Barang gagal
dihapus</font></h2>";
}
echo "Klik <a href='displaybarang.php'>di sini</a>
untuk kembali ke halaman display barang";
} else {
die ("Access Denied");
}
?>
</body>
</html>
```

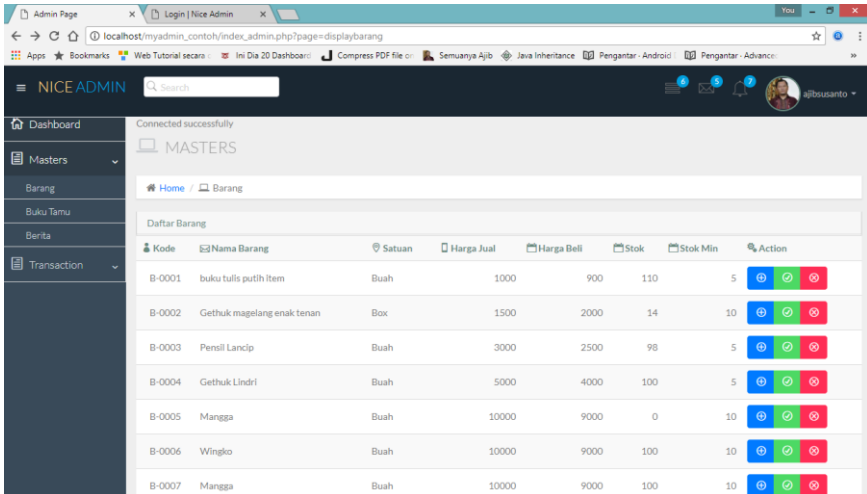


## Pertemuan 7 : CRUD (Create Read Update Delete) di PHP

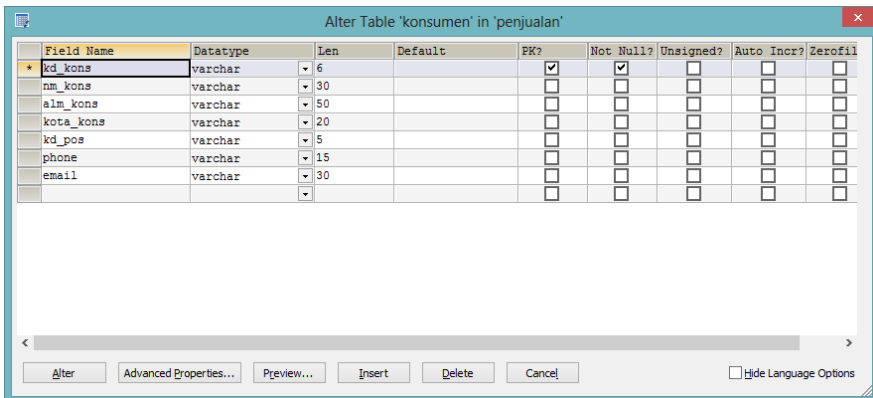
1. Buka kembali latihan Session pertemuan sebelumnya :



2. Modifikasi menu Masters, tambahkan sub menu Barang untuk memanggil latihan CRUD sebelumnya.



3. Buat table konsumen, tambahkan operasi CRUD kemudian tambahkan di menu Master sub menu Konsumen !



# Referensi