2014

DASAR-DASAR PHP

L. Erawan, M.Kom

Sistem Informasi - Fasilkom

Contents

PHP

Disadur dari situs w3schools.com

Pengenalan PHP

PHP singkatan dari PHP:Hypertext Prepocessor. Php merupakan bahasa script yang dijalankan pada sisi server (SSS : Server Side Scripting). Database yang didukung PHP antara lain : MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC. PHP adalah software Open Source, bebas untuk diunduh dan digunakan.

File PHP dapat berisi teks, tag HTML, dan Script. File PHP dikembalikan ke browser dalam bentuk plain HTML. File PHP dapat berekstensi .php, .php3, atau .phtml.

Kelebihan PHP sebagai bahasa script adalah : dapat dijalankan pada berbagai platform (windows, linux, dll), kompatibel terhadap hampir semua server yang digunakan saat ini, bebas diunduh dari situs resmi PHP www.php.net, mudah dipelajari dan berjalan dengan efisien pada sisi server.

Untuk menggunakan PHP harus menginstall terlebih dahulu web server Apache (atau IIS) pada komputer/server yang akan digunakan, serta menginstall PHP dan MySQL. Atau dapat menggunakan layanan hosting yang menyediakan dukungan terhadap PHP dan MySQL.

Tiga Model Penggunaan PHP

Sebelum menginstall, pastikan terlebih tujuan penggunaan PHP. Terdapat 3 model penggunaan PHP, yaitu :

- Website dan Aplikasi Web (Server Side Scripting)
- Command Line Scripting
- Aplikasi Desktop (GUI)

Model pertama merupakan penggunaan paling umum, dibutuhkan 3 hal : PHP, sebuah server web, dan browser web. Untuk platform Linux dan MacOS X, dapat menggunakan server web Apache, server web IIS digunakan untuk platform Windows. Browser web biasanya sudah ada dalam setiap komputer, terutama yang menggunakan Windows, yaitu Internet Explorer (IE). Selain IE, ada Mozilla FireFox, Google Chrome, Opera, dan lain-lain. Selain itu, dapat juga menyewa space web dari suatu layanan hosting di Internet. Dengan cara ini, yang perlu dilakukan hanya menulis script PHP, mengupload ke server yang disewa, dan melihat hasilnya pada browser.

Dalam hal penggunaan PHP dengan cara menginstall sendiri, ada 2 pilihan metode menghubungkan PHP ke server. Pada sebagian server PHP mempunyai *Direct Module Interface* (SAPI). Termasuk kedalam kelompok server ini adalah Apache, Microsoft Internet Information

Server, Netscape dan server iPlanet. Sebagian server yang lain memiliki dukungan untuk ISAPI, Microsfot Module Interface (contohnya OmniHTTPd). Jika PHP tidak memiliki dukungan Modul untuk server web Anda, maka gunakan PHP sebagai processor CGI atau FastCGI. Artinya, server disetup untuk menggunakan CGI executable dari PHP untuk memproses semua file PHP yang diminta pada server.

Jika PHP akan digunakan sebagai Command Line Scripting, misalnya untuk menulis script yang membangkitkan gambar secara otomatis secara offline, atau memproses file teks yang dikendalikan oleh beberapa argumen yang dikirimkan kepada script, maka dibutuhkan Command Line Executable. Dalam kasus ini, tidak diperlukan server dan browser.

PHP juga dapat digunakan untuk menulis aplikasi dekstop GUI. Untuk itu diperlukan ekstensi PHP-GTK. Cara ini memiliki pendekatan yang berbeda dari membuat halaman-halaman web, seperti tidak ada output HTML yang dibuat, tetapi mengelola windows dan objek yang berada didalamnya. PHP-GTK tidak disertakan dalam distribusi resmi PHP.

Instalasi PHP untuk Bidang Pengembangan Website dan Aplikasi Web

Sumber: www.php.net/manual/en/install.php

Instalasi Pada Windows

Windows Installer (PHP 5.1.0 dan sebelumnya)

Installer PHP Windows dapat diunduh dari www.php.net/downloads.php. Installer ini akan menginstall versi CGI dari PHP untuk IIS, PWS, dan Xitami dan menkonfigurasi server web. Installer tidak termasuk ekstensi eksternal PHP tambahan (php_*.dll).

Untuk menggunakan Installer ini, pertama install dulu server HTTP atau web dan pastikan telah bekerja dengan baik. Lalu jalankan file Installer dan ikuti petunjuk instalasi dari installation wizard. Tersedia 2 jenis instalasi, *standard* yang menyediakan setting default, dan *advanced* yang akan memberikan sejumlah pertanyaan selama proses instalasi.

Selama proses, wizard instalasi akan mengumpulkan informasi yang cukup untuk setup file *php.ini* dan menkonfigurasi server web untuk menggunakan PHP. Salah satu server web yang tidak akan dikonfigurasi oleh installer ini adalah Apache.

Setelah proses instalasi selesai, sistem akan meminta Anda untuk merestart sistem, atau Anda dapat langsung menggunakan PHP

Setup PHP jenis ini tidak aman. Jika menginginkan setup PHP yang aman, sebaiknya Anda menggunakan cara manual, dan atur setiap option dengan hati-hati.

Windows Installer adalah cara mudah untuk membuat PHP bekerja, tetapi installer ini membatasi banyak aspek, sebagai contoh, setup ekstensi secara otomatis tidak didukung. Penggunaan cara ini tidak disarankan

Windows Installer (PHP 5.2 dan setelahnya)

Windows Installer PHP untuk versi PHP ini dibuat menggunakan teknologi MSI menggunakan Toolkit Wix (wix.sourceforge.net/). Installer ini akan menginstall dan mengkonfigurasi PHP dan semua ekstensi built-in dan PECL untuk server web Apache, IIS, dan Xitami.

Langkah-langkah instalasi sebagai berikut. Pertama, install terlebih dulu server web dan pastikan telah bekerja dengan baik, lalu pilih salah satu tipe instalasi berikut :

Instalasi Normal

Jalankan installer MSI dan ikuti petunjuk yang diberikan wizard instalasi. Ketika proses berjalan, pertama Anda akan diminta untuk memilih server web yang akan dikonfigurasi serta menentukan rincian konfigurasi yang dibutuhkan. Lalu Anda akan diminta untuk menentukan ekstensi dan fitur yang ingin diinstall dan diaktifkankan. Dengan memilih "Will be installed on local hard drive" dalam menu drop down untuk setiap item, Anda dapat menentukan apakah fitur akan diinstall atau tidak. Dengan memilih "Entire feature will be installed on local hard drive", Anda akan dapat menginstall seluruh sub fitur dari fitur-fitur yang ditawarkan.

Installer kemudian akan melakukan setup terhadap PHP untuk digunakan dalam windows serta setup file php.ini dan mengkonfigurasi server web untuk menggunakan PHP. Saat ini server web yang dapat dikonfigurasi adalah IIS, Apache, Xitami, dan Samba, selainnya harus dikonfigurasi secara manual.

Tidak disarankan untuk menginstal semua ekstensi secara default, karena banyak dari ekstensi tersebut tergantung dengan sumber daya diluar PHP untuk dapat bekerja dengan benar. Jika terlanjur, tipe instalasi "Repair Mode" dapat dijalankan dengan menu "Add/Remove Programs" dari control panel untuk mengenable atau mendisablekan ekstensi dan fitur usai proses instalasi.

Instalasi Manual

Untuk melakukan instalasi secara manual, berikut langkah-langkahnya:

1. Pilih dan unduh paket distribusi PHP

Ada beberapa versi paket distribusi, pilih yang sesuai dengan server web yang sedang digunakan.

- Jika PHP akan digunakan dengan IIS, pilih PHP 5.3 VC9 Non Thread Safe atau PHP 5.2 VC6 Non Thread Safe.
- Jika PHP akan digunakan dengan IIS7 atau lebih tinggi dan PHP 5.3+, pilih PHP VC9 binari.
- Jika PHP akan digunakan dengan Apache 1 atau 2, pilih PHP 5.3 VC6 atau PHP 5.2 VC6.

Versi VC9 disusun dengan kompiler Visual Studio 2008, dan membutuhkan Microsoft 2008 C++ Runtime (x86) atau Microsoft 2008 C++ Runtime (x64)

Unpack isi dari file arsip zip kedalam direktori yang dipilih, misalnya c:\php\.

Berikut daftar modul dan file *executable* yang terdapat dalam file *zip* distribusi :

- *go-pear.bat*, script setup PEAR
- *php-cgi.exe, CGI* executable yang dapat digunakan saat menjalankan PHP pada IIS melalui CGI atau FastCGI

- *php-win.exe*, PHP executable untuk mengeksekusi script PHP tanpa menggunakan jendela command line (Sebagai contoh, aplikasi PHP yang menggunakan GUI Windows)
- *php.exe*, PHP executable yang digunakan untuk mengeksekusi script PHP didalam interface command line (CLI)
- *php5apache2_2.dll*, modul Apache 2.2.x
- *php5apache2_2_filter.dll*, filter Apache 2.2.x

2. Mengubah File php.ini

Setelah isi paket instalasi PHP diekstrak, copy file *php.ini-production* kedalam *php.ini* didalam direktori yang sama. Jika diperlukan, file php.ini juga dapat ditempatkan di lokasi lain yang diinginkan tetapi memerlukan konfigurasi tambahan.

3. File Konfigurasi

File konfigurasi (*php.ini*) dibaca ketika PHP start up. Untuk PHP sebagai modul dari server, proses ini terjadi hanya sekali ketika server web distart. Untuk versi CLI dan CGI, proses ini terjadi setiap kali ada permintaan.

File php.ini akan dicari pada lokasi-lokasi berikut secara berurutan :

- Lokasi spesifik modul SAPI (directive *PHPIniDir* in Apache 2, opsi command line *-c* dalam CGI dan CLI, parameter *php_ini* dalam NSAPI, variabel *PHP_INI_PATH*environment dalam THTTPD)
- Variabel *PHPRC* environment. Sebelum PHP 5.2.0, variabel ini dicek setelah key registry tersebut dibawah ini.
- Pada PHP 5.2.0, lokasi dari file *php.ini* dapat diset untuk versi PHP yang berbeda. Key registry berikut ini dicek dalam urutan :
    - *[HKEY_LOCAL_MACHINE\SOFTWARE\PHP\x.y.z]*
    - *[HKEY_LOCAL_MACHINE\SOFTWARE\PHP\x.y]*
    - *[HKEY_LOCAL_MACHINE\SOFTWARE\PHP\x]*
- *dimana* x, y dan z menyatakan versi major, minor, dan release PHP. Jika terdapat sebuah nilai untuk *IniFilePath* dalam kunci-kunci ini, yang pertama ditemukan akan digunakan sebagai lokasi dari file *php.ini* (Windows only).
- *[HKEY_LOCAL_MACHINE\SOFTWARE\PHP]*, nilai dari *IniFilePath* (Windows only).
- Direktori kerja saat ini (kecuali CLI).
- Direktori server web (untuk modul SAPI), atau direktori PHP(selain di Windows).
- Direktori Windows (*C:\windows* or *C:\winnt*) (for Windows), or *--with-config-file-path* compile time option.

Jika file *php-sapi.ini* ada (contohnya: php-cli.ini atau php-apache.ini), maka akan digunakan daripada file *php.ini*. Nama SAPI dapat ditentukan dengan *php_sapi_name()*.

File *php.ini* memberitahu PHP bagaimana mengkonfigurasi diri sendiri, dan bagaimana bekerja dalam lingkungan dimana PHP berjalan. Berikut ini beberapa pengaturan yang membantu PHP bekerja lebih baik dalam lingkungan windows :

Direktive yang dibutuhkan :

- extension_dir = <path to extension directory>

  Direktive ini diperlukan untuk menunjukkan direktori tempat ekstension-ekstension PHP disimpan. Path dapat absolut (misalnya "c:\php\ext") atau relatif (misalnya ".\ext"). Ekstension-ekstension yang tertulis dibagian bawah dari file php.ini perlu dijelaskan lokasinya dalam direktive ini

- extension = xxxxx.dll

  Setiap ekstension yang akan diaktifkan diberikan direktive ini untuk memberitahu PHP ekstension mana yang akan diload saat startup

- log_error = on

  fasilitas pencatatan kesalahan PHP yang dapat digunakan untuk mengirimkan kesalahan ke sebuah file atau layanan (misalnya syslog) dan bekerja bersama direktive *error_log* berikut ini.

- error_log = <path to the error log file>

  direktiv ini digunakan untuk menentukan jalur penyimpanan file log error PHP. File harus diset agar dapat ditulisi. Lokasi file ini biasanya terdapat dalam direktori TEMP

- cgi.force_redirect = 0

  Direktiv ini diperlukan jika PHP dijalankan di SO Windows. Pada banyak web server, direktiv ini diperlukan untuk mengamankan direktori. Jika direktiv ini diaktifkan pada PHP yang dijalankan di Windows, maka PHP tidak akan berjalan.

Pengaturan opsional:

- max-execution_time=##

  pengaturan ini untuk menentukan maksimal waktu yang ditetapkan dalam mengeksekusi suatu skrip, pengaturan bawaan adalah 30 detik. Jika aplikasi PHP membutuhkan waktu yang lama untuk mengeksekusi, naikkan nilainya.

- memory_limit=##M

  Jumlah memori yang disediakan untuk proses PHP dengan satuan MB. Nilai bawaannya adalah 128 yang sesuai untuk sebagian besar aplikasi PHP. Beberapa aplikasi yang kompleks mungkin memerlukan nilai yang lebih besar.

- display_error=off

  pengaturan ini untuk menentukan apakah semua pesan kesalahan akan disertakan oleh PHP ketika mengirimkan pesan kesalahan ke server web. Jika aturan ini diset *on*, maka semua kelas pesan yang telah didefinisikan pada direktiv *error_reporting* akan dikirim ke server web. Untuk alasan keamanan, sebaiknya aturan ini ditentukan *off* agar tidak mengungkapkan informasi keamanan yang sensitif yang sering disertakan pada suatu pesan kesalahan.

- open_basedir = <jalur ke direktori-direktori, dipisahkan dengan tanda titik koma>

  Untuk penjelasannya dimisalkan contoh:

  open_basedir="c:\inetpub\wwwroot;c:\inetpub\temp"

  Spesifikasi ini menentukan jalur direktori yang PHP diperbolehkan untuk melakukan operasi sistem file. Setiap operasi yang dilakukan diluar jalur direktori yang telah ditentukan akan menimbulkan kesalahan.

- upload_max_filesize = ###M dan post_max_size = ##M

  Untuk menentukan ukuran file maksimal yang dapat diupload dan maksimal data yang dapat diposting. Nilai-nilai ini sebaiknya diperbesar jika aplikasi PHP perlu mengupload file-file besar sejenis file gambar atau video.

Sintaks PHP

Skrip PHP dieksekusi di server dan hasil eksekusi yang berupa kode HTML dikirim ke komputer klien.

Simbol Pembuka dan Penutup Skrip PHP

Skrip PHP selalu diawali dengan tanda **'<?php'** dan ditutup dengan **'?>'**. Skrip PHP dapat diletakkan dimana saja dalam suatu dokumen HTML. Beberapa server yang sudah diatur konfigurasi directive '*shorthand-support*', dapat mengawali skrip dengan tanda '<?' dan diakhiri dengan '?>'. Tetapi demi kompatibilitas maksimum, disarankan menggunakan bentuk standar '<?php'.

Perluasan File PHP

File PHP harus disimpan dengan perluasan '**.php**'. File PHP biasanya berisi tag-tag HTML dan beberapa kode skrip PHP. Contoh:

```
<html>
<body>
    <?php
    echo "Hello World";
    ?>
</body>
</html>
```

Simbol Pemisah Setiap Perintah PHP

Setiap perintah PHP harus diakhiri dengan simbol semikolon '**;**'. Tugas simbol ini untuk membedakan atau memisahkan satu perintah PHP dengan perintah PHP lainnya.

Perintah Mencetak Hasil

Ada dua jenis perintah PHP untuk mencetak keluaran atau hasil yaitu '**echo**' dan '**print**'. Contoh:

```
<html>
<body>
```

```
<?php
echo "Hello World";
print “I’m learning PHP Script Language now”;
?>
</body></html>
```

Variabel PHP

Sintaks Perintah Membuat Variabel

Suatu variabel digunakan untuk menyimpan suatu nilai, dapat berupa teks, angka, atau array. Variabel dalam PHP menggunakan simbol '$' di awal namanya. Sintaks perintah membuat variabel:

```
$nama_var = nilai;
```

Tipe Data dari Variabel

Tipe data variabel tidak perlu didekalarasikan, PHP akan otomatis mengkonversi atau menentukan tipe data variabel berdasarkan nilai yang disimpannya. Contoh:

```
<?php

$nama='Alvina Khansa';

$nilai=90;

?>
```

Variabel *nama* diatas otomatis akan bertipe *string*, variabel *nilai* akan bertipe *integer*.

Aturan Pemberian Nama Variabel

- Harus dimulai dengan huruf atau garis bawah (*underscore*) '_'.
- Hanya dapat menggunakan karakter *alphanumeric* dan *underscore*
- (A-Z / a-z / 0 - 9, dan _)
- Sebaiknya tidak menggunakan spasi, jika nama variabel terdiri lebih dari satu kata, pisahkan dengan underscore (*$nama_depan*, *$nilai_tugas*) atau kapitalisasi *($namaDepan, $nilaiTugas*)

Mendeklarasikan/Membuat Variabel

Tidak ada perintah khusus dalam PHP untuk membuat variabel. Sebuah variabel akan terbentuk pada saat sebuah nilai ditentukan kepadanya. Contoh perintah:

```
$mobil='Avanza';
```

Setelah perintah tersebut dieksekusi maka variabel *$mobil* akan berisi *Avanza*.

Catatan: untuk memberikan suatu nilai berupa teks/string kepada sebuah variabel, nilai tersebut harus diapit dengan tanda petik tunggal atau ganda.

Scope Variabel

Scope atau ruang lingkup variabel adalah bagian dari skrip yang dapat mereferensikan variabel tersebut. Ada 4 scope variabel dalam PHP:

Scope Local

Suatu variabel yang dibuat pada suatu fungsi akan menjadi variabel lokal (memiliki scope local) dan hanya bisa diakses di dalam fungsi. Nama variabel yang sama dapat dibuat dalam fungsi yang berbeda, sebab variabel lokal hanya dikenali oleh fungsi yang membentuk variabel tersebut. Variabel lokal akan dihapus setelah fungsi usai dieksekusi. Contoh:

```
<?php

function dicoba(){

        $lokal='Saya hanya bisa diakses dari fungsi dimana saya berada';

        echo $lokal;      //mencetak var lokal

}

echo $lokal;      //akan terjadi error

?>
```

Scope Global

Scope global dimiliki oleh variabel yang dibuat diluar fungsi. Variabel dengan scope global dapat diakses dari bagian manapun dari program selama perintah tersebut ditulis diluar suatu fungsi. Variabel global dapat diakses dari dalam suatu fungsi dengan menggunakan kata kunci '***global***'. Contoh:

```
<?php

$a = 10; // scope global

function myTest()

{

        // mengacu ke variabel scope global

        echo global $a;

}

myTest();
?>
```

PHP menyimpan semua variabel global dalam array bernama **$GLOBAL[]**. Indeks dari array adalah nama dari variabel-variabel tersebut. Array ini dapat diakses dari dalam fungsi, tetapi juga dapat digunakan untuk mengupdate variabel global secara langsung. Contoh:

```
<?php

        global $a;
        $a="abc";

        // adalah sama dengan menggunakan perintah:


        $GLOBALS["a"]="abc";

?>
```

Scope Statik

Ketika suatu fungsi selesai digunakan, secara normal semua variabelnya akan dihapus. Jika diinginkan variabel-variabel tersebut tidak dihapus ketika fungsi selesai dipakai, gunakan kata kunci '***static***' saat membuat variabel. Contoh pembuatan variabel statik :

```
<?php

        static $varStatik

<?
```

variabel *$varStatik* sekarang menjadi variabel statik.

Operator

Operator digunakan untuk mengolah nilai. PHP memiliki beberapa kategori operator sebagai berikut:

Operator Hitung

Operator	Penjelasan	Contoh	Hasil
+	Pertambahan	x=2 y=x+2	y=4
-	Pengurangan	x=2 y=5-x	y=3
*****	Perkalian	x=4 y=x*5	y=20
/	Pembagian	y=15/5	y=3
%	Sisa hasil bagi	x=10%5 y=10%8 z=5%2	x=0 y=2 z=1
++	Inkremen	x=5 x++	x=6
--	dekremen	x=5 x--	x=4

Contoh 1:

```
<?php
        //menghitung penjualan bersih
        $jual=100000;
```

```
        $potongan=0.1;
        $net=$jual-($jual*$potongan);
        echo "Penjualan bersih = Rp $net,00";
?>
```

Contoh 2:

```
<?php
        //penggunaan operator modulus/sisa hasil bagi
        $hal=10;
        if ($hal % 2 == 0)
        echo 'Halaman genap';
        else
        echo 'Halaman ganjil';
?>
```

Contoh 3:

```
<?php
        //penggunaan inkremen
        for($i=1;$i<=100;$i++){
        echo "$i ";
        }
?>
```

Operator Penugasan

Operator	Contoh	Sama dengan
=	x=y	x=y
+=	x+=y	x=x+y

-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
.=	x.=y	x=x.y
%=	x%=y	x=x%y

Operator Perbandingan

Operator	Penjelasan	Contoh
==	sama dengan	5==8 mengembalikan nilai *false*
!=	tidak sama dengan	5!=8 mengembalikan nilai *true*
<>	tidak sama dengan	5<>8 mengembalikan nilai *true*
>	lebih besar dari	5>8 mengembalikan nilai *false*
<	lebih kecil dari	5<8 mengembalikan nilai *true*
>=	lebih besar dari atau sama dengan	5>=8 mengembalikan nilai *false*
<=	lebih kecil dari atau sama dengan	5<=8 mengembalikan nilai *true*

Operator Logika

Operator	Penjelasan	Contoh
&&	and	x=6; y=3; (x < 10 && y > 1) mengembalikan *true*
||	or	x=6; y=3;

		(x==5 || y==5) mengembalikan *false*
!	not	x=6; y=3; !(x==y) mengembalikan true

Struktur Kendali

Seringkali alur eksekusi suatu program tidak berjalan lurus dari baris kode pertama sampai baris terakhir. Kadang-kadang pada suatu baris tertentu alur program bercabang (struktur keputusan). Di lain baris alur program akan kembali ke baris kode sebelumnya untuk mengulangi sekelompok perintah (struktur perulangan).

Struktur keputusan dalam PHP mengenal struktur *if* dan *switch*. Untuk menerapkan suatu perulangan baris-baris kode program menggunakan struktur *for*, *while*, *do…while*, dan *foreach*. Berikut ini penjelasan masing-masing struktur tersebut.

Struktur Keputusan

Ketika Anda menulis kode program, suatu saat Anda perlu menggunakan mekanisme percabangan sehingga berdasarkan kondisi program Anda akan melakukan aksi yang berbeda. Misalnya saja Anda ingin membuat program yang dapat menampilkan salam yang berbeda tergantung jenis hari pada saat itu, maka kode programnya akan seperti ini:

```
<?php
$hari=date("D");
if ($hari=="Mon")
        echo "Selamat berlibur";
else
        echo "Selamat bekerja dan berkarya";
?>
```

Jika program diatas dijalankan pada hari Minggu, maka Anda akan memperoleh ucapan 'Selamat berlibur', tapi jika dijalankan selain hari Minggu akan memperoleh ucapan 'Selamat bekerja dan berkarya'.

Ya, contoh diatas adalah salah satu penggunaan struktur kondisi (if) yang dimiliki PHP. Struktur *if* mempunyai beberapa format, berikut penjelasannya.

If

Struktur ini memiliki 3 jenis format pemakaian, yaitu:

- ***Pertama***, bila struktur ini hanya akan mengeksekusi beberapa kode program hanya jika nilai kondisi true, maka format yang digunakan:

if (kondisi) kode_yang_akan dieksekusi_bila_nilai_kondisi_true

Contoh:

```
<?php
$hari=date("D");
if ($hari=="Mon") echo "Selamat berlibur";
?>
```

- ***Kedua***, baik nilai kondisi *if true* atau *false* akan sama-sama mengeksekusi suatu kode program, maka format yang digunakan:

if (kondisi)

kode_yang_akan dieksekusi_bila_nilai_kondisi_true;

else

kode_yang_akan dieksekusi_bila_nilai_kondisi_false;

Contoh:

```
<?php
$hari=date("D");
if ($hari=="Mon")
    echo "Selamat berlibur";
else
    echo "Selamat bekerja dan berkarya";
?>
```

- ***Ketiga***, bila ada beberapa kondisi yang perlu dievaluasi, maka format yang digunakan:

if (kondisi_1)

*kode_yang_akan dieksekusi_bila_nilai_**kondisi_1_true**;*

elseif (kondisi_2)

*kode_yang_akan dieksekusi_bila_nilai_**kondisi_2_true**;*

else

*kode_yang_akan dieksekusi_bila_nilai_**kondisi_1_dan_2_false**;*

Contoh:

```
<?php
$hari=date("D");
if ($hari=="Mon")
        echo "Selamat berlibur";
elseif ($hari=="Sat")
        echo "Selamat libur panjang";
else
        echo "Selamat bekerja dan berkarya";
?>
```

Jika kode program yang akan dieksekusi dalam *if* lebih dari satu, maka kode-kode program tersebut harus dikurung dengan tanda kurung kurawal buka dan tutup seperti ini:

```
<?php
$hari=date("D");
if ($hari=="Mon"){
        echo "Hai coy,<br />";
        echo "Selamat berlibur<br />";
        echo "Jangan lupa pesenanku ya";
}
?>
```

Switch

Jika dalam program ada banyak pilihan, maka lebih baik menggunakan *switch*. Sintaksnya:

```
switch(n) {
        case label_1        : pernyataan_yg_dieksekusi_jika_n=label_1;
                               break;
        case label_2        : pernyataan_yg_dieksekusi_jika_n=label_2;
                              break;
        default             : dieksekusi_bila_nilai_n_bukan_label2_atau_label_1;
```

```
}
```

Contoh:

```
<html>
<body>
<?php
$x=1;
switch ($x)
{
case 1:
    echo "Number 1";
    break;
case 2:
    echo "Number 2";
    break;
case 3:
    echo "Number 3";
    break;
default:
    echo "No number between 1 and 3";
}
?>
</body>
</html>
```

Struktur Perulangan (Loop)

PHP mempunyai beberapa struktur perulangan:

for

Struktur perulangan *for* sering digunakan dalam pemrograman. Sintaksnya:

```
for (inisialisasi; kondisi; inkremen)
{
        kode yang akan ndieksekusi;
}
```

Tiga parameternya:

inisialisasi	:	nilai awal konter
kondisi	:	parameter yang akan dievaluasi pada setiap iterasi perulangan. Jika hasil evaluasi true, perulangan akan dilanjutkan, bila false perulangan akan dihentikan
inkremen	:	nilai pertambahan konter setiap satu iterasi perulangan diselesaikan

Setiap parameter di atas bersifat opsional.

Contoh:

```
<html>
<body>
<?php
for ($i=1; $i<=5; $i++)
{
        echo "Bilangan ke-$i:  " . $i . "<br />";
}
?>
</body>
</html>
```

Contoh *for* tanpa parameter *kondisi*:

```
<html>
<body>
<?php
for ($i=1;; $i++)
```

```
{
        echo "The number is " . $i . "<br />";
        if ($i>=5) exit;
}
?>
</body>
</html>
```

foreach

Perulangan ini disediakan PHP untuk memudahkan kita mengakses elemen-elemen suatu array. Sintaks:

```
foreach ($array as $value)
{
        kode yang akan dieksekusi;
}
```

Pada setiap iterasi perulangan, nilai dari elemen array yang sedang ditunjuk oleh pointer akan diletakkan ke variabel *$value* kemudian pointer akan bergerak menunjuk elemen array berikutnya sehingga pada iterasi selanjutnya, elemen array berikutnya yang akan diolah.

Contoh berikut akan mencetak nilai "satu", "dua", dan "tiga" secara berturut-turut:

```
<html>
<body>
<?php
$x=array("satu","dua","tiga");
foreach ($x as $value)
{
        echo $value . "<br />";
}
?>
</body>
```

```
</html>
```

While

Perulangan while akan menjalankan suatu blok kode (sekelompok kode) selama kondisinya bernilai true. Sintaks perulangan ini:

```
while (condition)
{
    blok kode yang akan dieksekusi;
}
```

Contoh:

```
<html>
<body>
<?php
$i=1;
while($i<=5)
{
    echo "Bilangan ke-$i:  " . $i . "<br />";
    $i++;
}
?>
</body>
</html>
```

Contoh *while* di atas akan terus menjalankan blok kodenya selama nilai dari variabel *$i* lebih kecil atau sama dengan 5. Dan pada setiap iterasinya, nilai variabel *$i* akan bertambah 1 (*$i*++)

Do…while

Satu yang unik dari perulangan ini adalah, dia pasti akan melakukan iterasi, minimal satu kali, meskipun nilai kondisinya tidak pernah true. Hal ini disebabkan struktur perulangan yang menyebabkan blok kode akan dijalankan lebih dulu, lalu kondisinya dievaluasi belakangan. Sintaks perulangan ini:

```
do
```

```
{
        kode yang akan dieksekusi;
}
while (kondisi);
```

Contoh:

```
<html>
<body>
<?php
$i=1;
do
{
        $i++;
        echo "Bilangan ke-$i:  " . $i . "<br />";
}
while ($i<=5);
?>
</body>
</html>
```

Contoh di atas akan menjalankan kode *$i++* dan *echo "Bilangan ke-$i: " . $i . "<br />"; terlebih dahulu*, setelah itu baru kondisinya dievaluasi.

Fungsi

Fungsi merupakan salah satu teknik pemrograman modular. Sebuah aplikasi besar disusun dari modul-modul yang berupa sebuah fungsi atau prosedur. Fungsi berisi sekelompok kode dengan tugas dan tujuan spesifik. Fungsi tidak akan dieksekusi ketika program dijalankan. Fungsi hanya akan dieksekusi jika dilakukan pemanggilan terhadapnya. Pemanggilan dapat dilakukan dari mana saja dalam program. Keuntungan teknik ini, modul-modul yang dibuat dapat digunakan berkali-kali (reuse).

Membuat Fungsi

Sebuah fungsi dibuat dengan aturan sintaks:

```
function namaFungsi() {
    kode-kode yang akan dieksekusi;
}
```

Beberapa petunjuk dalam membuat sebuah fungsi:

- Namai fungsi yang menggambarkan fungsinya
- Nama fungsi dimulai dengan huruf atau garis bawah (*underscore*), tidak boleh angka.

Pemanggilan Fungsi

Ketika fungsi sudah dibuat, dia dapat dijalankan dengan cara dipanggil. Pemanggilan suatu fungsi mengikuti pola: nama fungsi lalu diikuti tanda kurung dan nilai parameter jika ada. Contoh:

```
tambah(10,20);
```

Memanggil sebuah fungsi bernama *tambah* dengan nilai parameter 10 dan 20. Jika tidak ada nilai parameter, maka pemanggilan fungsi:

```
cetak();
```

Parameter Fungsi

Untuk menambah daya guna fungsi dapat ditambahkan parameter fungsi yang tidak lain adalah serupa variabel. Parameter ini dituliskan sesudah nama fungsi didalam tanda kurung. Dengan parameter ini, hasil dari fungsi dapat diatur sesuai dengan keinginan.

Contoh:

```
<html>
```

```
<body>
<?php
function namaProg ($fprogram) {
        echo $fname . "<br />"
}
echo "Bahasa membuat struktur dan konten adalah ".namaProg("HTML");
echo "Unsur interaktif diberikan oleh ".namaProg("Javascript");
echo "Memperindah tampilan fungsi dari ".namaProg("CSS");
echo "Aplikasi web pengolahan data menggunakan ".namaProg("PHP");
?>
</body>
</html>
```

Hasil dari fungsi diatas (nama program) dapat diubah-ubah dengan memberi nilai berbeda pada parameter *$fprogram* saat pemanggilan fungsi.

Nilai Balik Fungsi

Fungsi dapat diatur agar mengembalikan hasil berupa nilai dengan cara menggunakan kata kunci *return*.

Contoh:

```
<html>
<body>
<?php
function tambah(){
        $total = $x + $y;
        return $total;
}
echo "$x + $y = " . tambah(5,20);
?>
</body>
</html>
```

Nilai yang dikembalikan pada fungsi diatas adalah jumlah dari nilai variabel $x dan $y yang ada didalam variabel $total. Hasil di layar browser adalah tampilan *5 + 20 = 25*.

Array

Array adalah jenis tipe data khusus yang menyimpan sejumlah nilai data dalam sebuah variabel. Nilai-nilai data tersebut, yang disebut elemen array, memiliki indeks yang menunjukkan urutannya dalam array. Elemen array pertama akan memiliki indeks 0, elemen kedua berindeks 1, dan seterusnya.

Dengan array, proses pencarian data tertentu akan mudah dilakukan. Misalkan Anda memiliki data nama barang elektronik yang disimpan dalam variabel tunggal seperti ini:

```
$brg1="DVD";

$brg2="Televisi";

$brg3="Lemari es";
```

Bagaimana jika kita ingin mencari barang elektronik tertentu dengan menggunakan perulangan terhadap kelompok variabel tersebut? Tentu akan repot dengan berbagai kode yang harus kita buat untuk menentukan nomor urut variabel tersebut terlebih dahulu dan sebagainya. Lain halnya jika menggunakan array, kita hanya perlu menggunakan indeks array untuk menemukan barang elektronik yang kita cari.

Membuat Array

Membuat array dalam PHP menggunakan fungsi array().

Contoh:

```
$brg = array ("DVD","Televisi","Lemari es");
```

Untuk mengakses elemen array tersebut dengan menggunakan indeksnya. Nilai data "Televisi" dapat diakses dengan kode $brg[1]. Misalnya dicetak, maka perintahnya adalah:

```
echo "Elemen array kedua adalah : " . $brg[1];
```

Array juga dapat dibuat dengan cara menentukan indeksnya langsung seperti:

```
$brg[0]="DVD";

$brg[1]="Televisi";

$brg[2]="Lemari es";
```

Jenis Array

Dalam PHP terdapat 3 jenis array, yaitu array numerik (*indexed arrays*), array assosiatif (*associative array*), dan array multidimensi (*multidimensional array*).

Array numerik

Jenis array yang berindeks numeris seperti contoh sebelumnya.

Array assosiatif

Jenis array yang memiliki indeks berupa string.

Array multidimensi

Jenis arary yang indeksnya juga array. Artinya, elemen dari array jenis ini berupa suatu array juga.

Contoh:

```
$mhs = array (
        array ('A12.2010.04567', 'Anita Larasati', 3.5);
        array ('A12.2010.05678', 'Dude Harmono', 3);
        array ('A12.2010.06789', 'Ernawati Listyani', 2.75);
)
```

Untuk mengakses elemen array multidimensi, misalkan akan dicetak, maka kodenya:

```
echo "NIM : " . $mhs[0][0]. "Nama : " . $mhs[0][1] . "IPK : " . $mhs[0][2] . "<br />";
echo "NIM : " . $mhs[1][0]. "Nama : " . $mhs[1][1] . "IPK : " . $mhs[1][2]; . "<br />";
echo "NIM : " . $mhs[2][0]. "Nama : " . $mhs[2][1] . "IPK : " . $mhs[2][2];
```

Mencari Panjang Suatu Array

Seperti diketahui array terdiri dari sejumlah elemen array. Untuk mengetahui jumlah elemen dalam suatu array atau panjang suatu array dapat menggunakan fungsi count().

Contoh:

```
$mobil = array ("Volvo","Jaguar", "Mercedez");
echo "Panjang array adalah : " . count($mobil);
```

Fungsi count() diatas akan mengembalikan nilai panjang array yaitu 3.

Mencetak Seluruh Elemen Array

Untuk mencetak seluruh elemen array dapat digunakan suatu perulangan *for* sebagai berikut:

```
$mobil = array ("Volvo","Jaguar", "Mercedez");
```

```
$jum = count($mobil);

for ($i=0; $i<$jum; $i++) {

        echo $mobil[$i] . '<br />';

}
```

Array Assosiatif

Penulisan indeks untuk jenis array ini bagi Anda yang tidak terbiasa akan terbilang ganjil. Tetapi jika sudah terbiasa tidak akan menjadi masalah. Kode pembuatan array jenis ini jika menggunakan fungsi array() tampak seperti berikut ini:

```
$umur = array ("Joni"=>"17", "Indra"=>"18", "Susi"=>"16");

echo "Joni berusia " . $umur("Joni") . " tahun";
```

Tampak dalam menentukan elemen-elemen array diatas menggunakan simbol "=>" yang dibuat dengan simbol sama dengan ("=") dikombinasikan dengan simbol lebih besar (">").

Untuk mencetak seluruh nilai elemen array jenis ini dapat menggunakan perulangan *foreach*, seperti ini:

```
$umur = array ("Joni"=>"17", "Indra"=>"18", "Susi"=>"16");

foreach ($umur as $x=>$nilaiX) {

        echo "Indeks " . $x . " bernilai " . $nilaiX . "<br />";

}
```

Menyorting Array

Elemen-elemen array dapat diurutkan baik secara alfabet maupun numeris, menaik (*ascending*) atau menurun (*descending*). Berikutnya penjelasan tentang jenis-jenis fungsi pengurutan array dalam PHP.

Fungsi sort()

Fungsi ini akan mengurutkan elemen secara menaik atau *ascending*. Jika elemen-elemen array berupa string, maka akan diurutkan menurut urutan alfabet (a, b, c, dan seterusnya). Dan jika berupa nilai numerik, maka elemen array akan diurutkan secara numeris (1, 2, 3, dan seterusnya). Berikut ini contoh keduanya:

Pengurutan terhadap data string:

```
$mobil = array ("Volvo","Jaguar", "Mercedez");

sort($mobil);

$pjg = count($mobil);
```

```
for ($i=0; $i<$pjg; $i++) {
        echo $mobil[$i] . "<br />";
}
```

Pengurutan terhadap data numerik:

```
$bilangan = array (5, 7, 3, 4, 2, 1, 6);
sort($bilangan);
$pjg = count($bilangan);
for ($i=0; $i<$pjg; $i++) {
        echo $bilangan[$i] . "<br />";
}
```

Fungsi rsort()

Fungsi ini akan mengurutkan elemen array secara menurun atau descending. Contoh untuk data string dan numerik:

Pengurutan terhadao data string:

```
$mobil = array ("Volvo","Jaguar", "Mercedez");
rsort($mobil);
$pjg = count($mobil);
for ($i=0; $i<$pjg; $i++) {
        echo $mobil[$i] . "<br />";
}
```

Pengurutan terhadap data numeris:

```
$bilangan = array (5, 7, 3, 4, 2, 1, 6);
rsort($bilangan);
$pjg = count($bilangan);
for ($i=0; $i<$pjg; $i++) {
        echo $bilangan[$i] . "<br />";
}
```

Fungsi asort()

Fungsi ini digunakan untuk jenis array assosiatif yang akan mengurutkan array secara menaik berdasarkan nilai data elemen array.

Contoh:

```
$umur = array ("Joni"=>"17", "Indra"=>"18", "Susi"=>"16");

asort($umur);

foreach ($umur as $x=>$nilaiX) {

        echo "Indeks " . $x . " bernilai " . $nilaiX . "<br />";

}
```

Karena elemen ketiga memiliki nilai data terkecil (16) maka akan dicetak sebagai yang pertama.

Fungsi ksort()

Fungsi akan mengurutkan secara menaik array assosiatif berdasarkan nilai indeks elemen array.

Contoh:

```
$umur = array ("Joni"=>"17", "Indra"=>"18", "Susi"=>"16");

ksort($umur);

foreach ($umur as $x=>$nilaiX) {

        echo "Indeks " . $x . " bernilai " . $nilaiX . "<br />";

}
```

Karena elemen kedua memiliki indeks berupa "Indra" yang memiliki urutan alfabetis lebih dahulu ("I' lebih dahulu daripada "J" atau "S") maka akan dicetak sebagai yang pertama.

Fungsi arsort()

Fungsi ini merupakan gabungan sifat dari fungsi asort() dan rsort() yang akan menghasilkan pengurutan array assosiatif secara menurun (*descending*).

Contoh:

```
$umur = array ("Joni"=>"17", "Indra"=>"18", "Susi"=>"16");

arsort($umur);

foreach ($umur as $x=>$nilaiX) {

        echo "Indeks " . $x . " bernilai " . $nilaiX . "<br />";

}
```

Akan menghasilkan elemen kedua (“18”) dicetak sebagai yang pertama.

Fungsi krsort()

Fungsi ini gabungan sifat dari fungsi ksort() dan rsort() yang akan menghasilkan pengurutan array assosiatif secara menurun (*descending*) berdasarkan nilai indeks elemen array.

Contoh:

```
$umur = array (“Joni”=>”17”, “Indra”=>”18”, “Susi”=>”16”);

krsort($umur);

foreach ($umur as $x=>$nilaiX) {

        echo “Indeks “ . $x . “ bernilai “ . $nilaiX . “<br />”;

}
```

Akan menghasilkan elemen ketiga (“Susi”) dicetak sebagai yang pertama.

Mengelola Data Waktu dan Tanggal

Nilai Timestamp

Sebelum mempelajari fungsi-fungsi yang dapat digunakan, pertamakali yang harus kita ketahui adalah informasi waktu dasar yang digunakan dalam memanipulasi data waktu dan tanggal. Informasi ini disebut epoch timestamp atau UNIX timestamp atau timestamp saja. Istilah ini merujuk pada jumlah detik yang dihitung mulai pukul 0:0:0 tanggal 1 Januari 1970 GMT sampai dengan waktu yang kita tentukan. Nilai inilah yang menjadi dasar perhitungan informasi waktu dan tanggal dalam fungsi-fungsi terkait di PHP. Misalnya ingin mencari nilai timestamp pada tanggal 2 Januari 1970 maka nilai timestamp yang kita peroleh adalah 86400 yang berasal dari perhitungan 60 * 60 * 24. Bagaimana dengan tanggal saat ini, berapa nilai timestampnya?

Fungsi-fungsi Waktu dan Tanggal

Nilai timestamp tersebut dapat dimanipulasi untuk menghasilkan informasi tertentu dari waktu dan tanggal. PHP menyediakan banyak fungsi untuk hal ini. Salah satu fungsi dasar yang perlu diketahui adalah date().

Fungsi date()

Fungsi date() ini akan mengembalikan nilai waktu dan tanggal saat ini dari komputer yang digunakan. Sintaks fungsi:

date (format, timestamp)

Parameter *format* wajib digunakan yang akan menentukan format waktu yang ingin dihasilkan. Parameter *timestamp* bersifat opsional yang menentukan waktu timestamp yang akan dimanipulasi. Nilai default parameter ini adalah waktu timestamp saat ini.

Karakter yang dapat digunakan dalam parameter *format* untuk menghasilkan informasi waktu dan tanggal yang kita inginkan antara lain:

d : menghasilkan nilai hari dalam sebulan (01 s/d 31)

m : menghasilkan nilai bulan dalam setahun (01 s/d 12)

Y : menghasilkan nilai tahun dengan panjang 4 digit.

Karakter lain yang dapat digunakan dalam parameter format antara lain "/", ".", atau "-".

Contoh:

```
echo date("Y/m/d");
```

```
echo date("d-m-Y");

echo date("m.d.Y");
```

Fungsi mktime()

Nilai timestamp dapat diperoleh dengan fungsi mktime(). Sintaks fungsi:

mktime(jam, menit, detik, bulan, tanggal, tahun);

Contoh penggunaan dari fungsi mktime() ini misalkan kita ingin menampilkan informasi tanggal untuk hari besok. Maka kodenya akan seperti berikut ini:

```
$tomorrow = mktime(0, 0, 0, date("m"), date("d")+1, date("Y"));

echo "Besok adalah tanggal " . date("Y/m/d", $tomorrow);
```

Dalam kode diatas pertama ditentukan terlebih dahulu nilai timestamp untuk hari besok. Pada baris berikutnya, nilai tersebut digunakan sebagai parameter kedua dari fungsi date().

Sebagai catatan, untuk menggunakan fungsi ini Anda perlu melihat versi PHP yang Anda gunakan sebab fungsi ini telah dihapus pada PHP versi 5.1.0 keatas. Sebagai gantinya, gunakan fungsi time() berikut ini.

Fungsi time()

Fungsi ini menggantikan fungsi mktime() untuk PHP versi 5.1.0 keatas. Seperti fungsi mktime(), fungsi ini juga akan mengembalikan nilai timestamp. Fungsi time tidak memiliki parameter. Sebagai contoh, dimisalkan akan ditampilkan informasi tanggal seminggu dari saat ini, maka kode programnya:

```
$mgdpn= time() + (7 * 24 * 60 * 60);

echo "Hari ini tanggal : " . date('d-m-Y') . "<br />";

echo "Seminggu lagi tanggal : " . date('d-m-Y', $mgdpn);
```

Fungsi getdate()

Fungsi ini akan mengembalikan nilai dalam bentuk array assosiatif berisi informasi tanggal dari nilai timestamp yang diberikan. Jika tidak menyertakan nilai timestamp, maka waktu lokal (waktu komputer atau server) yang akan digunakan.

Array assosiatif yang dihasilkan oleh fungsi berisi sekumpulan informasi tentang waktu yang tersimpan didalam masing-masing elemen array:

"seconds"	berisi nilai numeris menunjukkan detik	*0* s/d *59*
"minutes"	berisi nilai numeris menunjukkan menit	*0* s/d *59*
"hours"	berisi nilai numeris menunjukkan jam	*0* s/d *23*
"mday"	berisi nilai numeris menunjukkan hari dalam	*1* s/d *31*

	sebulan	
"wday"	berisi nilai numeris menunjukkan hari dalam seminggu	*0* (sunday) s/d *6* (saturday)
"mon"	berisi nilai numeris menunjukkan bulan	*1* through *12*
"year"	berisi nilai numeris menunjukkan tahun dengan panjang 4 digit	contoh: *1999* atau 2014
"yday"	berisi nilai numeris menunjukkan hari dalam setahun	*0* s/d *365*
"weekday"	berisi teks menunjukkan nama hari	*Sunday* s/d *Saturday*
"month"	berisi teks menunjukkan nama bulan	*January* s/d *December*
0	Jumlah detik sejak dari UNIX epoch, sama seperti nilai yang dikembalikan oleh fungsi time() dan yang digunakan dalam fungsi date()	Tergantung sistem yang digunakan, biasanya *2147483648 s/d 2147483647*.

Elemen array pertama berisi informasi tentang detik dari waktu saat ini yang indeksnya bernama "seconds", lalu informasi menit pada eleman array kedua dengan indeks "minutes", demikian dan seterusnya. Jadi, untuk menampilkan informasi waktu dan tanggal dapat menggunakan kode seperti ini:

```
$tgl = getdate();
echo "Waktu saat ini menunjukkan pukul " . $tgl['hours'] . ':' . $tgl['minutes'] . ':' .
    $tgl['seconds'];
```

Hasilnya jika ditampilkan pada layar browser akan menampilkan teks 'Waktu saat ini menunjukkan pukul 14:7:14'.

Fungsi checkdate()

Fungsi ini dapat digunakan untuk memeriksa kebenaran dari data waktu yang diberikan. Sintaks fungsi:

checkdate(bulan, tanggal, tahun)

Parameter bulan dapat berisi nilai 1-12 yang menunjukkan bulan, parameter tanggal dapat berisi 1-31 tergantung bulannya, sedangkan parameter tahun dapat berisi nilai 1-32767.

Fungsi akan mengembalikan nilai boolean true (angka 1) jika tanggal valid dan false jika ternyata tanggal tidak valid (misalnya tanggal 30 Pebruari jelas invalid).

Contoh:

```
if (checkdate(2,29,2014)){
        echo "Tanggal valid";
} else {
        echo "Tanggal invalid";
}
```

Kode diatas akan menghasilkan teks 'Tanggal invalid' karena tanggal 29 Pebruari 2014 tidak ada.

Fungsi date_default_timezone_set()

Seperti telah dijelaskan diatas, informasi tanggal dan waktu berhubungan dengan berbagai zona waktu dan format tanggal yang berbeda. Jika program manipulasi tanggal yang kita buat tidak menghasilkan informasi waktu yang tepat, misalnya menghasilkan informasi waktu pukul 1 padahal waktu saat itu menunjukkan pukul 9, maka kita perlu mengatur zona waktu yang akan digunakan terlebih dahulu agar sesuai dengan zona waktu tempat kita berada. Fungsi yang dapat digunakan untuk mengatur zona waktu lokal ini adalah date_default_timezone_set(). Sintaks fungsi ini:

```
date_default_time_zone_set('nama_zona_waktu')
```

Parameter nama zona waktu merupakan data string yang berbentuk 'nama kawasan/nama ibukota'. Untuk Indonesia nama zona waktunya adalah 'Asia/Jakarta'. Contoh penggunaan:

```
$today = date('h:i:s, d-m-Y');
echo $today;
```

Contoh diatas akan menghasilkan informasi waktu '03:28:20 18-02-2014' sementara pada saat program ini dijalankan waktu menunjukkan pukul 10:30:00 dan tanggal 18-02-2014. Disini ada perbedaan informasi waktu yang dihasilkan. Maka agar informasi waktu sesuai dengan waktu saat ini dimana program ini dijalankan (dijalankan di Indonesia), fungsi date_default_timezone_set() perlu digunakan, sehingga kode sebelumnya akan menjadi:

```
date_default_timezone_set('Asia/Jakarta');
$today = date('h:i:s, d-m-Y');
echo $today;
```

Kode sekarang akan menghasilkan informasi waktu yang sesuai dengan waktu dimana program dijalankan (Dalam hal ini waktu yang sesuai dengan waktu komputer atau server yang digunakan).

Variabel Superglobal

Variabel super global adalah variabel terpasang siap pakai yang selalu tersedia dalam semua lingkup aplikasi web yang dibuat. Tidak seperti variabel biasa yang hanya bisa digunakan pada sebuah file program (misalnya program tambah data mahasiswa) dimana variabel tersebut dibuat yang artinya lingkupnya hanya sebatas program tersebut.

Beberapa variabel yang telah ditetapkan dalam PHP adalah superglobal, yang berarti bahwa variabel tersebut selalu dapat diakses, terlepas dari lingkupnya, dan kita dapat mengaksesnya dari fungsi, kelas atau file program tanpa harus melakukan sesuatu yang khusus.

Variabel-variabel superglobal adalah:

- $_GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_ENV
- $_COOKIE
- $_SESSION

Variabel $GLOBAL

$GLOBAL adalah variabel super global PHP yang digunakan untuk mengakses variabel global dari mana saja didalam script PHP (juga dari dalam fungsi atau metode). PHP menyimpan semua variabel global dalam sebuah array yang disebut *$GLOBALS[index].* Indeks berupa nama variabel.

Contoh berikut memperlihatkan bagaimana penggunaan $GLOBAL ini:

```
<?php
$x = 75;

$y = 25;

 function addition()

{

        $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];

}
```

```
addition();

echo $z;

?>
```

Dalam contoh di atas, karena z adalah variabel dalam array $GLOBALS, maka variabel ini dapat diakses dari luar fungsi!

Variabel $_SERVER

$_SERVER adalah variabel super global PHP yang menyimpan informasi tentang header, path, dan lokasi file program atau script.

Berikut contoh penggunaan $_SERVER:

```
<?php

echo $_SERVER['PHP_SELF'];

echo "<br>";

echo $_SERVER['SERVER_NAME'];

echo "<br>";

echo $_SERVER['HTTP_HOST'];

echo "<br>";

echo $_SERVER['HTTP_REFERER'];

echo "<br>";

echo $_SERVER['HTTP_USER_AGENT'];

echo "<br>";

echo $_SERVER['SCRIPT_NAME'];

?>
```

Tabel dibawah ini berisi daftar elemen-elemen yang paling sering digunakan didalam $_SERVER

Elemen	Deskripsi
$_SERVER['PHP_SELF']	mengembalikan nama file script yang sedang berjalan
$_SERVER['GATEWAY_INTERFACE']	mengembalikan versi CGI yang digunakan server

$_SERVER['SERVER_ADDR']	mengembalikan alamat IP server host
$_SERVER['SERVER_NAME']	mengembalikan nama dari server host (misalnya www.w3schools.com)
$_SERVER['SERVER_SOFTWARE']	mengembalikan string yang mengidentifikasikan server (misalnya Apache/2.2.24)
$_SERVER['SERVER_PROTOCOL']	mengembalikan nama dan versi dari protokol informasi (misalnya HTTP/1.1)
$_SERVER['REQUEST_METHOD']	mengembalikan metode request/permintaan yang digunakan untuk mengakses halaman (misalnya POST)
$_SERVER['REQUEST_TIME']	mengembalikan timestamp dari awal permintaan (misalnya 1377687496)
$_SERVER['QUERY_STRING']	mengembalikan string query jika halaman diakses melalui sebuah string query
$_SERVER['HTTP_ACCEPT']	mengembalikan header Accept dari request saat ini
$_SERVER['HTTP_ACCEPT_CHARSET']	mengembalikan header Accept_Charset dari request saat ini (misalnya utf-8,ISO-8859-1)
$_SERVER['HTTP_HOST']	mengembalikan header Host dari request saat ini
$_SERVER['HTTP_REFERER']	mengembalikan URL lengkap dari halaman saat ini (tidak dapat diandalkan karena tidak semua user agent (browser) mendukungnya)
$_SERVER['HTTPS']	adalah script yang dikirimkan melalui protokol HTTP yang aman
$_SERVER['REMOTE_ADDR']	mengembalikan alamat IP dari perangkat yang digunakan pengguna untuk mengakses halaman saat ini
$_SERVER['REMOTE_HOST']	mengembalikan nama host yang digunakan pengguna untuk mengakses halaman saat ini
$_SERVER['REMOTE_PORT']	mengembalikan port yang digunakan pada perangkat pengguna untuk berkomunikasi dengan server web
$_SERVER['SCRIPT_FILENAME']	mengembalikan nama path absolut dari script yang dijalankan saat ini

$_SERVER['SERVER_ADMIN']	mengembalikan nilai yang diberikan ke direktif SERVER_ADMIN dalam file konfigurasi server web (jika script anda berjalan diatas host virtual, maka akan berisi nilai yang didefinisikan untuk virtual host tersebut (misalnya someone@w3schools.com)
$_SERVER['SERVER_PORT']	mengembalikan port pada komputer server yang digunakan oleh server web untuk berkomunikasi (misalnya 80)
$_SERVER['SERVER_SIGNATURE']	mengembalikan versi server dan nama host virtual yang ditambahkan ke halaman yang digenerate oleh server
$_SERVER['PATH_TRANSLATED']	mengembalikan path berbasis file sistem yang mengarah ke script saat ini
$_SERVER['SCRIPT_NAME']	mengembalikan path script saat ini
$_SERVER['SCRIPT_URI']	mengembalikan URI dari halaman web saat ini

Variabel $_REQUEST

$_REQUEST digunakan untuk mengumpulkan data setelah pengiriman (*submitting*) sebuah form HTML terjadi.

Contoh di bawah ini memperlihatkan sebuah form dengan field input dan tombol submit. Ketika pengguna mengirimkan data dengan mengklik "Submit", maka data form dikirim ke file yang ditentukan dalam atribut *action* dari tag <form>. Dalam contoh ini, kami arahkan pengiriman ke file ini sendiri untuk mengolah data form. Jika Anda ingin menggunakan file PHP lain untuk memproses data form, ganti nilai atribut action dengan nama file pilihan Anda. Setelah pengiriman, maka kita bisa menggunakan variabel super global $_REQUEST untuk mengumpulkan nilai dari field input:

```
<html>

<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">

Name: <input type="text" name="fname">

<input type="submit">

</form>

<?php

    $name = $_REQUEST['fname'];

    echo $name;
```

```
?>

</body>

</html>
```

Variabel $_POST

$_POST secara luas digunakan untuk mengumpulkan data form setelah proses pengiriman sebuah form HTML dengan *method="post"*. $_POST Juga banyak digunakan untuk melewatkan variabel.

Contoh di bawah ini memperlihatkan sebuah form dengan field input dan tombol submit. Ketika pengguna mengirimkan data dengan mengklik "Submit", maka data form dikirim ke file yang ditentukan dalam atribut *action* dari tag <form>. Dalam contoh ini, kami arahkan pengiriman ke file ini sendiri untuk mengolah data form. Jika Anda ingin menggunakan file PHP lain untuk memproses data form, ganti nilai atribut action dengan nama file pilihan Anda. Setelah pengiriman, maka kita bisa menggunakan variabel super global $_POST untuk mengumpulkan nilai dari field input:

```
<html>

<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">

Name: <input type="text" name="fname">

<input type="submit">

</form>

<?php

        $name = $_POST['fname'];

        echo $name;

?>

</body>

</html>
```

Variabel $_GET

$_GET juga dapat digunakan untuk mengumpulkan data form yang dikirimkan oleh form HTML dengan method="get". $_GET Juga dapat digunakan untuk mengumpulkan data yang dikirim melalui sebua URL.

Anggaplah kita memiliki halaman HTML yang berisi hyperlink dengan parameter:

```
<html>
```

```
<body>

<a href="test_get.php?subject=PHP&web=W3schools.com">Test $GET</a>

</body>

</html>
```

Ketika pengguna mengklik link "Test $GET", maka parameter *subject* dan *web* dikirim ke file *test_get.php*, kemudian nilai-nilai parameter tersebut di file *test_get.*php dapat diakses dengan $ _GET.

Berikut ini kode file *test_get.php* yang digunakan:

```
<html>

<body>

<?php

echo "Belajar " . $_GET['subject'] . " di " . $_GET['web'];

?>

</body></html>
```

Form

Seandainya saja tidak ada form HTML, tentulah programmer web akan kelimpungan ketika berusaha mengumpulkan data dari pemakai/pengunjung melalui aplikasinya. Form adalah alat pengumpul data yang utama bagi seorang programmer web. Inilah alasannya mengapa Anda harus menguasai alat ini dengan baik jika ingin menjadi seorang programmer web.

Script PHP bekerjasama secara baik dengan form HTML. Elemen-elemen form pada sebuah halaman web secara otomatis akan tersedia pada script PHP.

Berikut contoh sebuah form (***form.php***):

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="nama" />
Age: <input type="text" name="usia" />
<input type="submit" />
</form>

</body>
</html>
```

Ketika seorang user mengisi isian data pada form diatas lalu menekan tombol submit, maka data yang diisikan kedalam form akan dikirim ke file PHP (*welcome.php*) yang ditentukan.

Berikut isi file ***welcome.php***:

```
<html>
<body>

Welcome <?php echo $_POST["nama"]; ?>!<br />
You are <?php echo $_POST["usia"]; ?> years old.

</body>
</html>
```

Variabel $_POST digunakan untuk mengambil data dari form. Ada satu variabel jenis ini, yaitu $_GET, yang akan digunakan jika *method* form menggunakan *get.*

Catatan: Variabel-variabel $_POST dan $_GET adalah variabel super global yang disediakan oleh PHP untuk keperluan pengiriman data form. Artinya, ketika tombol submit diklik maka data form akan

diletakkan dalam variabel ini secara otomatis. Maka ketika PHP akan mengambil data yang dikirim oleh form tersebut, akan menggunakan variabel-variabel ini untuk mengaksesnya.

Validasi FORM

Input user sebaiknya divalidasi pada sisi browser menggunakan script klien, misalnya javascript, bila memungkinkan. Ini akan mempercepat proses validasi dan mengurangi komunikasi dengan server.

Jika input user akan disimpan kedalam database, perlu dipertimbangkan melakukan proses validasi di server. Cara yang disarankan dalam melakukan proses validasi di server adalah mengirimkan data form ke form itu sendiri. Maka isi dari atribut *action* form berupa nama file dari form itu sendiri.

Dalam contoh diatas, isi dari atribut *action*, yaitu berisi *welcome.php*, diganti *form.php*. Kemudian isi file *welcome.php* ditulis dibawah kode yang ada pada file *form.php* sehingga isi dari file *form.php* akan menjadi:

```
 <html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="nama" />
Age: <input type="text" name="usia" />
<input type="submit" />
</form>
Welcome <?php echo $_POST["nama"]; ?>!<br />
You are <?php echo $_POST["usia"]; ?> years old.
</body>
</html>
```

Bagian yang dicetak tebal adalah kode yang semula ditulis di file *welcome.php*.