Dasar Pemrograman PHP

Hands-on Labs 5

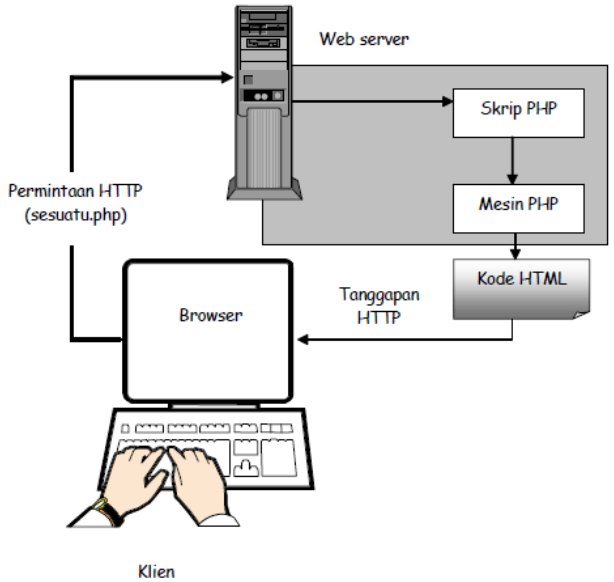
I Made Arsa Suyadnya
Program Studi Teknik Elektro, FT - UNUD

A. Pengenalan PHP

Bahasa pemrograman PHP diciptakan oleh Rasmus Lerdorf, pada tahun 1995. PHP atau merupakan singkatan rekursif dari **P**HP : **H**ypertext **P**reprocessor adalah suatu bahasa pemrograman yang termasuk kategori server side programming. Server side programming adalah jenis bahasa pemrograman yang nantinya script/program tersebut akan dijalankan oleh server. Selanjutnya hasil pengolahan script/program tersebut akan dikirim ke client sebagai output.

Secara khusus, PHP dirancang untuk membentuk web dinamis. Berbeda dengan HTML yang hanya bisa menampilkan konten statis, PHP bisa berinteraksi dengan database, file dan folder, sehingga membuat PHP bisa menampilkan konten yang dinamis dari sebuah website. PHP adalah bahasa scripting, bukan bahasa tag-based seperti HTML. PHP termasuk bahasa yang *cross-platform*, ini artinya PHP bisa berjalan pada sistem operasi yang berbeda-beda (Windows, Linux, ataupun Mac). Program PHP ditulis dalam file plain text (teks biasa) dan mempunyai akhiran/ekstensi ".php".

Untuk dapat berjalan, PHP membutuhkan **web server**, yang bertugas untuk memproses file-file php dan mengirimkan hasil pemrosesan dalam bentuk HTML untuk ditampilkan di browser client. Secara sederhana, prinsip kerja PHP dapat dilihat pada Gambar 1. Web server sendiri adalah *software* yang di-*install* pada komputer server yang berada di jaringan intranet atau internet yang berfungsi untuk melayani permintaan-permintaan web dari client. Web server yang paling banyak digunakan saat ini untuk PHP adalah "Apache" (www.apache.org). Selain Apache, PHP juga memerlukan PHP binary (www.php.net) yang bisa dikonfigurasikan sebagai modul Apache atau pun sebagai aplikasi CGI (Common Gateway Interface). Untuk media penyimpanan datanya (database server), PHP biasa menggunakan "MySQL" (www.mysql.com) walaupun PHP juga mendukung banyak DBMS (Database Management System) lainnya.



Gambar 1. Prinsip Kerja PHP

Untuk meng-*install* dan mengkonfigurasi ketiga *software* tersebut (Apache, MySQL, PHP) agar dapat berjalan dan saling terhubung, memang cukup sulit. Maka dari itu dibuatlah paket *software* LAMP, XAMPP, MAMP, WAMP, dan lain-lain untuk membantu proses instalasi dan konfigurasi. Dalam satu kali

instalasi, sudah mencakup ketiga *software* tersebut dan sudah dikonfigurasi untuk keperluan lingkungan pengembangan aplikasi web. Sehingga, programmer web hanya tinggal menulis program PHP dan langsung menjalankan atau melakukan tes terhadap program yang ditulis tersebut melalui web browser. Untuk mendapatkan paket *software* web server tersebut dapat men-*download* dari website yang bersangkutan (untuk XAMPP: http://www.apachefriends.org dan untuk WampServer: http://www.wampserver.com/en/).

B. Struktur Dasar Script PHP

Kode PHP disimpan sebagai plain text dalam format ASCII, sehingga kode PHP dapat ditulis hampir di semua editor text seperti windows notepad, windows wordpad, Notepad++, dan lainnya. Kode PHP adalah kode yang disertakan di sebuah halaman HTML dan kode tersebut dijalankan oleh server sebelum dikirim ke browser. Kode-kode PHP dituliskan diantara tanda berikut ini.

```
<?php ..... ?>
```

Blok scripting PHP selalu diawali dengan <?php dan diakhiri dengan ?>. Blok scripting PHP dapat ditempatkan dimana saja di dalam dokumen. Setiap baris kode PHP harus diakhiri dengan semikolon (;). Semikolon ini merupakan separator yang digunakan untuk membedakan satu instruksi dengan instruksi lainnya.

Script PHP yang dibuat harus disimpan dalam ekstensi **.php**. Apabila tidak, maka script tidak akan bisa dijalankan dan hanya dianggap sebagai teks biasa. Dahulu script PHP dapat juga disimpan dalam bentuk ekstensi .php3, namun hal ini berlaku untuk PHP versi 3.x saja. Sejak rilis 4.x ke atas, ekstensi tersebut tidak digunakan lagi.

Untuk dapat dijalankan, script PHP yang dibuat harus diletakkan dalam root direktori dari web server. Sebagai contoh, apabila menggunakan XAMPP yang di-*install* pada komputer local (localhost) dengan sistem operasi Windows, script PHP diletakkan dalam direktori **C:\xampp\htdocs.**

Contoh script PHP dalam dokumen HTML:

```
<html>
<head>
<title>Halaman PHP pertamaku</title>
</head>
<body>
<h1>Script PHP dalam HTML</h1>
<?php
echo "Hello World!";
echo "Hello World!";
echo "Hello World!";
echo "Hello World!";
echo "Hello World!";
?>
</body>
</html>
```

Pada contoh script di atas, karena dalam dokumen HTML di atas terdapat script PHP, maka file di atas nantinya juga harus disimpan menggunakan ekstensi .php, bukan .htm atau .html. Misalnya disimpan dengan nama test1.php dalam folder 'test' dan untuk memanggilnya menggunakan URL http://localhost/test/test1.php

1. Penggunaan variabel

Variabel dalam pemrograman digunakan untuk menyimpan suatu nilai dan jika suatu saat nilai tersebut diperlukan kembali untuk proses perhitungan, maka cukup memanggil nama variabel tersebut.

Dalam PHP, nama suatu variabel ditandai dengan tanda **$**. Berikut ini adalah sintaks untuk menyimpan nilai ke dalam suatu variabel (dikenal juga dengan istilah *assignment*).

```
$namaVariabel = nilai;
```

Adapun beberapa aturan penulisan nama variabel adalah sebagai berikut:

- Nama variabel harus diawali dengan huruf atau underscore (_)
- Nama variabel hanya boleh dituliskan dengan alpha numeric a-z, A-Z, 0-9 dan underscore
- Nama variabel yang terdiri lebih dari satu kata, dapat dipisahkan dengan underscore

Berikut ini beberapa contoh menyimpan nilai ke dalam variabel.

```
<?php
$teks = "Hello World!";
$sebuah_bilangan = 4;
$bilanganYangLain = 8.567;
$teks2 = $teks;
?>
```

Keterangan:

- Perintah pertama digunakan untuk menyimpan nilai berupa string ke dalam variabel bernama $teks.
- Perintah kedua digunakan untuk menyimpan nilai berupa bilangan bulat 4 ke dalam variabel bernama $sebuah_bilangan.
- Perintah ketiga digunakan untuk menyimpan nilai berupa bilangan riil 8.567 ke dalam variabel bernama $bilanganYanglain.
- Perintah keempat digunakan untuk menyimpan nilai yang tersimpan dalam variabel $teks, dalam hal ini juga "Hello World!".

Catatan Penting:

Besar kecilnya huruf dalam nama variabel sangat berpengaruh. Maksudnya misalnya membuat variabel $a dengan $A, keduanya adalah berbeda. Sifat ini dinamakan *case sensitive*.

2. Menampilkan nilai variabel

Untuk menampilkan nilai yang telah tersimpan dalam variabel, digunakan echo. Berikut ini contoh penggunaannya:

```
<?php
$teks = "Hello World!";
$sebuah_bilangan = 4;
$bilanganYangLain = 8.567;
$teks2 = $teks;
echo $teks;
echo "<br />" . $sebuah_bilangan;
echo "<br />Isi dari variabel \$bilanganYangLain : " . $bilanganYangLain;
echo "<br />Isi dari variabel \$teks2 adalah : " . $teks2;
?>
```

Keterangan:
- Tanda titik (dot) digunakan untuk menggabungkan string. Dalam hal ini, nilai yang akan ditampilkan dianggap sebagai suatu string.
- Tanda backslash (\) di depan $ pada script di atas digunakan untuk menampilkan tanda $ ke dalam browser.

3. Menyisipkan komentar dalam script PHP

Seperti halnya bahasa pemrograman yang lain, komentar dalam suatu kode PHP tidak akan dieksekusi atau diproses. Biasanya komentar digunakan untuk memberikan deskripsi tentang script secara keseluruhan atau memberikan penjelasan pada baris perintah tertentu.

Terdapat dua cara memberikan komentar dalam PHP, yaitu:
- Menggunakan tanda // di depan teks komentar. Perintah ini hanya bisa berlaku untuk komentar dalam satu baris
- Menggunakan tanda /* di depan teks komentar dan diakhiri dengan */. Perintah ini dapat digunakan untuk komentar yang terdiri lebih dari satu baris.

Contoh:

```
<?php
echo "Hello World!"; // perintah ini akan mencetak Hello World!
?>
Contoh yang lain:
<?php
/*
 Berikut ini adalah perintah
 untuk menampilkan teks Hello World
 pada browser
*/
echo "Hello World!";
?>
```

C. Operator

Operator adalah simbol yang digunakan dalam program untuk melakukan suatu operasi, misalnya penjumlahan atau perkalian, perbandingan kesamaan dua buah nilai, atau bahkan memberikan nilai ke variabel. Nilai yang dioperasikan oleh operator (disebut *operand* atau argumen) bersama-sama operator membentuk ekspresi (ungkapan). sebagai contoh:

```
2 + 3 * 4
```

disebut ekspresi. Tanda + dan * disebut **operator**, sedangkan 2, 3 dan 4 adalah *operand* atau argumen.

Ada beberapa macam operator yang bisa digunakan, diantaranya adalah operator aritmetika, operator penugasan, operator perbandingan dan operator logika.

1. Operator Aritmetika

Operator aritmetika digunakan untuk operasi perhitungan yang melibatkan nilai berupa bilangan atau dalam operasi matematika. Ada beberapa macam operator aritmetika seperti terlihat pada Tabel 1 berikut ini.

Tabel 1. Operator Aritmetika pada PHP

Operator	Makna	Contoh
+	Penjumlahan	2 + 4
-	Pengurangan	6 - 2
*	Perkalian	5 * 3
/	Pembagian	15 / 3
%	Modulus/sisa hasil bagi	43 % 10
++	Increment	x=5; x++
--	Decrement	x=5; x--

Contoh script:

```
<?php
$penjumlahan = 2 + 4;
$pengurangan = 6 - 2;
$perkalian = 5 * 3;
$pembagian = 15 / 3;
$modulus = 5 % 2;
echo "Hasil: 2 + 4 = " . $penjumlahan."<br>";
echo "Hasil: 6 - 2 = " . $pengurangan."<br>";
echo "Hasil: 5 * 3 = " . $perkalian."<br>";
echo "Hasil: 15 / 3 = " . $pembagian."<br>";
echo "Hasil: 5 % 2 = " . $modulus;
?>
```

- **Tingkat Presedensi**

Perlu berhati-hati dalam menggunakan operator aritmetika, terutama jika menggunakan lebih dari satu operator yang berbeda dalam satu statement perhitungan, sebagai contoh script berikut ini:

```
<?php
$a = 3 + 4 * 5 – 6;
echo $a;
?>
```

Apabila script di atas dijalankan, maka hasil yang muncul bukan 29, tapi 17. Hal ini terjadi, karena operasi aritmatik yang dikerjakan terlebih dahulu adalah perkalian (*). Perkalian memiliki tingkat presedensi yang lebih tinggi daripada + dan -. Setelah perkalian dikerjakan, maka operasi yang dikerjakan adalah + dan -. Keduanya, + dan -, memiliki tingkat presedensi yang sama. Jika kedua operator memiliki pesedensi yang sama, maka bagian yang lebih kiri akan dikerjakan lebih dahulu. Operator lainnya yang memiliki presedensi yang sama dengan perkalian adalah pembagian (/) dan modulo (%). Agar bagian operasi tertentu dari statement perhitungan dikerjakan lebih dahulu maka dapat menggunakan tanda kurung, seperti contoh berikut.

```
<?php
$a = (3 + 4) * 5 – 6;
echo $a;
?>
```

- **Operator Pre/Post Increment dan Decrement**

Operator ini hanya digunakan pada proses increment maupun decrement dengan tingkat 1. Berikut ini adalah operator yang termasuk jenis ini:

```
$x++;
```
ekuivalen dengan $x += 1; atau $x = $x + 1;

```
$x--;
```
ekuivalen dengan $x -= 1; atau $x = $x – 1;

Contoh:
```
<?php
$x = 4;
$x++;
echo "Nilai x yang baru : ". $x;

$x = 4;
$x--;
echo "Nilai x yang baru : ". $x;
?>
```

2. Operasi Penugasan (Assignment Operators)

Salah satu operator penugasan yang sering ditemui adalah operator =, yang digunakan untuk memberikan nilai ke suatu variabel. Selain itu, PHP menyediakan operator yang terkait dengan pemberian nilai ke suatu variabel sebagaimana diperlihatkan pada Tabel 2.

Tabel 2. Operator Penugasan pada PHP

Operator	Contoh	Operasi yang ekuivalen
+=	$x += 2;	$x = $x + 2;
-=	$x -= 4;	$x = $x - 4;
*=	$x *= 3;	$x = $x * 3;
/=	$x /= 2;	$x = $x / 2;
%=	$x %= 5;	$x = $x % 5;
.=	$my_str.="hello";	$my_str = $my_str . "hello";

3. Operasi Perbandingan (Comparison Operators)

Operator perbandingan atau dikenal juga sebagai operator relasional adalah operator yang digunakan untuk melakukan perbandingan dua buah operand dan menghasilkan nilai benar atau salah (*true* atau *false*). Yang termasuk ke dalam kelompok operator ini, dapat dilihat pada Tabel 3.

Tabel 3. Operator Perbandingan pada PHP

Operator	Keterangan	Contoh
==	sama dengan	5==8 returns false
!=	tidak sama dengan	5!=8 returns true
>	lebih besar dari	5>8 returns false
<	kurang dari	5<8 returns true
>=	kurang dari dengan	5>=8 returns false
<=	kurang dari dengan	5<=8 returns true

4. Operator Logika

Operator logika biasa digunakan untuk menggabungkan kondisi berganda dan menghasilkan sebuah ekspresi yang bernilai benar atau salah. Berikut pada Tabel 4 ditampilkan operator yang termasuk ke dalam kelompok operator logika.

Tabel Error! No text of specified style in document.**.** Operator Logika pada PHP

Operator	Keterangan	Contoh
&&	and	x=6; y=3 (x < 10 && y > 1) returns true
||	or	x=6; y=3 (x==5 || y==5) returns false
!	not	x=6; y=3 !(x==y) returns true

Operator **and** atau **&&** menghasilkan nilai benar jika kedua operand bernilai benar. Operator **or** atau || menghasilkan nilai benar apabila ada salah satu operand yang bernilai benar.

D. Statement Kontrol

Dalam dunia pemrograman umumnya, terdapat 2 jenis statement kontrol yaitu: **statement kontrol kondisional (bersyarat)** dan **statement kontrol perulangan (*looping*)**.

Statement kontrol kondisional adalah statement kontrol yang digunakan untuk mengatur kapan suatu perintah akan dijalankan. Dengan statement ini kita bisa mengatur kapan suatu perintah akan dijalankan, yaitu ketika telah dipenuhinya suatu syarat tertentu. Sedangkan statement kontrol perulangan digunakan untuk mengatur perintah yang dijalankan secara berulang-ulang.

Dalam PHP, terdapat dua buah statement kontrol yang termasuk statement kontrol kondisional, yaitu IF dan SWITCH. Sedangkan yang termasuk statement kontrol perulangan adalah: FOR, WHILE, DO WHILE dan FOREACH.

1. Statement If

Pernyataan **if** biasa digunakan untuk mengambil keputusan berdasarkan suatu kondisi. PHP memiliki tiga macam bentuk if, yaitu:

- **if**

Bentuk pertama if berupa:

```
if (condition) {
    code to be executed if condition is true;
}
```

Contoh:

```
<html>
<body>
<?php
$total_beli = 200000;
$keterangan = "Tidak dapat diskon";

if ($total_beli >= 100000){
    $keterangan = "Dapat diskon";
}
echo $keterangan;
?>
</body>
</html>
```

- **if-else**

  Bentuk ini digunakan apabila ingin menjalankan suatu tindakan tertentu bila kondisi bernilai benar dan menjalankan tindakan yang lain kalau kondisi bernilai salah.

  Bentuk pernyataan:

```
if (condition) {
    code to be executed if condition is true;
} else {
    code to be executed if condition is false;
}
```

  Contoh:

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri"){
  echo "Have a nice weekend!";
} else {
  echo "Have a nice day!";
}
?>
</body>
</html>
```

- **if-elseif**

  Pernyataan if-elseif sangat bermanfaat untuk melakukan pengambilan keputusan yang melibatkan banyak alternatif.

  Bentuk pernyataan:

```
if (condition) {
    code to be executed if condition is true;
} elseif (condition) {
    code to be executed if condition is true;
} else {
    code to be executed if condition is false;
}
```

  Contoh:

```
<html>
<body>
<?php
```

```
$d=date("D");
if ($d=="Fri"){
  echo "Have a nice weekend!";
}elseif ($d=="Sun"){
  echo "Have a nice Sunday!";
}else{
  echo "Have a nice day!";
}
?>
</body>
</html>
```

2. Statement Switch

Switch digunakan untuk melakukan suatu aksi dari beberapa aksi yang berbeda berdasarkan pada satu atau lebih kondisi yang berbeda.

Bentuk pernyataan:

```
switch (n) {
    case label1:
        code to be executed if n=label1;
        break;
    case label2:
        code to be executed if n=label2;
        break;
    case label3:
        code to be executed if n=label3;
        break;
    ...
    default:
        code to be executed if n is different from all labels;
}
```

Contoh:

```
<html>
<body>
<?php
$favcolor = "red";

switch ($favcolor) {
    case "red":
        echo "Your favorite color is red!";
        break;
    case "blue":
        echo "Your favorite color is blue!";
        break;
    case "green":
        echo "Your favorite color is green!";
        break;
    default:
        echo "Your favorite color is neither red, blue, or green!";
}
?>
</body>
</html>
```

3. Statement While

Pernyataan while merupakan salah satu pernyataan yang berguna untuk melakukan suatu perulangan. Adapun bentuk pernyataannya:

```
while (condition is true) {
    code to be executed;
}
```

Pernyataan while akan memeriksa nilai kondisi/ekspresi terlebih dahulu. jika bernilai benar maka pernyataan-pernyataan atau kode yang tedapat dalam { } akan dijalankan dan kemudian kondisi/ekspresi dievaluasi lagi. Proses ini diulang terus-menerus sampai kondisi atau ekspresi bernilai salah.

Contoh berikut ini menunjukkan penggunaan while untuk menampilkan bilangan 1 hingga 5.

```
<html>
<body>
<?php
$x = 1;

while($x <= 5) {
    echo "Bilangan: $x <br>";
    $x++;
}
?>
</body>
</html>
```

4. Statement Do While

Pernyataan do-while mempunyai kegunaan yang serupa dengan pernyataan while. Bentuk pernyataannya:

```
do {
    code to be executed;
} while (condition is true);
```

Perulangan akan berakhir jika kondisi atau ekspresi (yang diuji sesudah pernyataan atau kode dijalankan) bernilai salah. Dalam perulangan do-while, paling tidak pernyataan dalam { } akan dieksekusi sekali.

Contoh berikut ini menunjukkan penggunaan do-while untuk menampilkan bilangan 1 hingga 5.

```
<html>
<body>
<?php
$x = 1;

do{
    echo "Bilangan: $x <br>";
    $x++;
}while ($x <= 5);
?>
</body>
</html>
```

5. Statement For

Pernyataan for juga merupakan pernyataan yang biasa digunakan untuk menangani perulangan proses. Bentuk pernyataannya:

```
for (initialization; condition; increment) {
    code to be executed;
}
```

Statatemen for bekerja sebagai berikut :

1. *Inisialisasi* sebagai nilai awal
2. *Kondisi* diuji, jika bernilai true (benar), maka perulangan dilakukan dengan mengerjakan blok pernyataan jika bernilai false (salah) maka perulangan berhenti dan blok pernyataan dilewati
3. Jika blok pernyataan hanya terdiri satu baris maka tanda kurung kurawal dapat ditiadakan
4. *Increment* merupakan nilai penambahan atau pengurangan untuk mengulangi pengerjaan blok pernyataan jika kondisi masih terpenuhi.

Contoh berikut ini menunjukkan penggunaan for untuk menampilkan bilangan 0 hingga 10.

```
<html>
<body>
<?php
for ($x = 0; $x <= 10; $x++) {
    echo "Bilangan: $x <br>";
}
?>
</body>
</html>
```

E. Array

Selama ini bentuk variabel yang dibuat untuk menyimpan suatu value atau nilai adalah berbentuk tunggal. Maksudnya, satu variabel hanya bisa untuk menyimpan sebuah value saja. Misalkan sebuah value telah disimpan atau di-assign ke dalam sebuah variabel, selanjutnya variabel tersebut akan di-assign kembali dengan sebuah value yang lain, maka value yang sebelumnya akan ditimpa oleh value yang baru.

Berbeda dengan variabel, array adalah suatu wadah yang dapat menampung sejumlah nilai. Misalkan suatu variabel bernama X bertipe data array, maka X ini dapat dibayangkan seperti Gambar 2 berikut.

X

0	1	2	3	4	...	N-1

Gambar 1. Gambaran array

Dalam terminologi array, array X di atas dikatakan mempunyai beberapa buah elemen yang berhingga yakni sejumlah N. Setiap elemen dikenali dengan indeks-nya: 0,1,2,3,...,N-1. PHP secara bawaan menggunakan indeks dimulai dengan nol.

- **Menciptakan array**

Suatu array dapat diciptakan dengan menggunakan konstruksi array. Sebagai contoh, array yang berisi 4 buah kota dapat dibentuk dengan menggunakan pernyataan berikut.

```
$kota = array("Yogya","Solo","Bandung","Bogor");
```

Alternatif lainnya, dapat menggunakan sederetan pernyataan berikut.

```
$kota[] = "Yogya";
$kota[] = "Solo";
$kota[] = "Bandung";
$kota[] = "Bogor";
```

Keempat pernyataan di atas dapat juga ditulis menjadi:

```
$kota[0] = "Yogya";
$kota[1] = "Solo";
$kota[2] = "Bandung";
$kota[3] = "Bogor";
```

atau dapat ditulis menjadi:

```
$kota = array(0=>"Yogya",1=>"Solo",2=>"Bandung",3=>"Bogor");
```

Angka yang diletakkan di dalam tanda [ ] merupakan kunci atau indeks dari array. Pada praktiknya, indeks tidak harus dimulai dari nol. Bahkan dapat menciptakan indeks yang tidak urut. Misalnya:

```
$wadah[10] = 60;
$wadah[17] = 65;
$wadah[25] = 55;
```

Perlu diketahui, apabila menuliskan pernyataan seperti:

```
$wadah[] = 67;
```

maka indeks yang akan digunakan berupa indeks tertinggi dari elemen-elemen array yang sudah terbentuk ditambah dengan satu. Jadi, sekiranya menuliskan empat pernyataan berikut:

```
$wadah[10] = 60;
$wadah[3] = 65;
$wadah[8] = 55;
$wadah[] = 67;
```

maka angka 67 akan disimpan ke elemen array yang memiliki indeks berupa 11 (10 + 1).

- **Mengambil isi array**

  Untuk mengambil isi array, dapat menggunakan notasi berikut ini.

```
$nama_array[indeks]
```

Contoh:

```
<?php
$x = array(10, 12, 3, 44, 50, "hallo");

// akan menampilkan value pada indeks ke-0, yaitu 10
echo $x[0];

// akan menampilkan value pada indeks ke-5, yaitu 'hallo'
echo $x[5];

// mengganti value pada indeks ke-2 dengan value yang baru (-3)
$x[2] = -3;
```

```
// akan menghasilkan -3 (yang tampil bukan 3 karena sudah
// ditimpa -3)
echo $x[2];
?>
```

- **Mengetahui jumlah elemen array**

PHP menyediakan fungsi bernama **count()** yang berguna untuk mendapatkan jumlah elemen array. Fungsi ini memerlukan argumen berupa array bersangkutan. Sebagai contoh:

```
<?php
$musik = array("Jazz","Blues","Fusion");
echo "Jumlah elemen : ".count($musik);
?>
```

akan menampilkan tulisan: "Jumlah elemen = 3", yang menyatakan bahwa jumlah elemen array **musik** adalah 3.

Adapun contoh lain penggunaan fungsi count() untuk menampilkan isi atau nilai dari suatu array menggunakan mekanisme perulangan adalah sebagai berikut.

```
<?php
$kota = array("Yogya","Solo","Bandung","Bogor");
$jumlah = count($kota);
for ($i=0; $i<$jumlah; $i++){
       echo "Elemen berindeks $i : $kota[$i] <br>";
}
?>
```

- **Array dengan indeks berupa string**

Sejauh ini, contoh penggunaan array masih menggunakan indeks bertipe integer. Selain indeks bertipe integer, PHP juga memperkenankan indeks bertipe string.

Contoh:

```
<?php
$hari["Sunday"] = "Minggu";
$hari["Monday"] = "Senin";
$hari["Tuesday"] = "Selasa";
$hari["Wednesday"] = "Rabu";

echo $hari["Tuesday"];
?>
```

F. POST dan GET Request

POST dan GET merupakan metode yang digunakan untuk mengirimkan nilai variabel/data ke halaman lain untuk diproses lebih lanjut. Dalam bagian ini akan dibahas perbedaan diantara kedua metode tersebut.

1. POST Request

Untuk dapat memahami POST request perhatikan contoh berikut ini. Dalam contoh ini, akan membuat sebuah form yang digunakan untuk login.

login.php

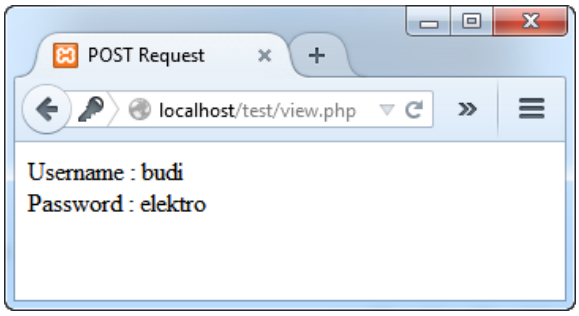
```
<html>
<title>POST Request</title>
<body>
Silakan Login dengan Username dan Password Anda
<form action="view.php" method="POST">
    <table>
        <tr>
            <td>Username</td>
            <td><input type="text" name="username"/></td>
        </tr>
        <tr>
            <td>Password</td>
            <td><input type="password" name="password"/></td>
        </tr>
        <tr>
            <td></td>
            <td><input type="submit" value="Login"/></td>
        </tr>
    </table>
</form>
</body>
</html>
```

Pada form di atas dapat dilihat bahwa terdapat dua komponen dalam form, yang masing-masing memiliki nama **'username'** dan **'password'** (perhatikan atribut **name="…"** pada komponen). Keduanya merupakan komponen form berbentuk text box. Selanjutnya, perhatikan bagian atribut **action="proses.php"**. Atribut ini memiliki makna bahwa apabila tombol Login tersebut diklik, maka data yang diisikan pada form nantinya akan dikirim dan diolah pada script bernama **view.php**.

view.php

```
<html>
<title>POST Request</title>
<body>
Username : <?php echo $_POST["username"]; ?><br/>
Password : <?php echo $_POST["password"]; ?>
</body>
</html>
```

Untuk mengetahui hasilnya, buka **login.php** melalui browser (pastikan kedua file ini: **login.php** dan **view.php** tersimpan di direktori **htdocs** pada web server). Kemudian sebagai input untuk username dan password, masing-masing adalah **budi** dan **elektro**. Setelah itu, klik tombol Login untuk mengirimkan data. Hasilnya dapat dilihat pada Gambar 3 berikut ini.



Gambar 2. Hasil proses dari POST request

2. GET Request

Untuk dapat memahami GET request maka contoh sebelumnya yakni file **login.php** akan dirubah menjadi seperti berikut ini.

login.php

```
<html>
<title>GET Request</title>
<body>
Silakan Login dengan Username dan Password Anda
<form action="view.php" method="GET">
    <table>
        <tr>
            <td>Username</td>
            <td><input type="text" name="username"/></td>
        </tr>
        <tr>
            <td>Password</td>
            <td><input type="password" name="password"/></td>
        </tr>
        <tr>
            <td></td>
            <td><input type="submit" value="Login"/></td>
        </tr>
    </table>
</form>
</body>
</html>
```

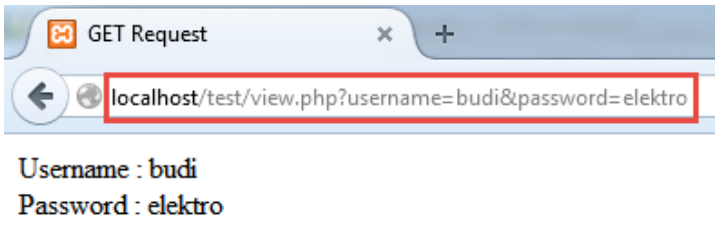
Jadi yang dirubah hanya atribut method, menjadi **method="GET"**. Dan kemudian merubah file **view.php** seperti berikut ini.

view.php

```
<html>
<title>POST Request</title>
<body>
Username : <?php echo $_GET["username"]; ?><br/>
Password : <?php echo $_GET["password"]; ?>
```

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

Selanjutnya dengan menggunakan input yang sama seperti pada contoh sebelumnya maka hasil yang diperoleh dapat dilihat pada Gambar 4. Perhatikan hasil pada Gambar 3 dan Gambar 4, keduanya memberikan hasil yang sama. Perbedaannya, apabila menggunakan metode GET, maka nilai variabel ditampilkan pada URL sedangkan metode POST tidak menampilkan nilai variabel tersebut.



Gambar 3. Hasil proses dari GET request

Adapun beberapa perbedaan lainnya antara metode POST dan GET dapat dilihat pada Tabel 5 berikut ini.

Tabel 4. Perbedaan antara Metode POST dan GET

POST	GET
▪ Nilai variabel tidak ditampilkan di URL ▪ Lebih aman ▪ Panjang dari URL string tidak dibatasi ▪ Pengambilan variabel dengan $_POST ▪ Biasanya untuk input data melalui form ▪ Digunakan untuk mengirim data-data penting seperti password	▪ Nilai variabel ditampilkan di URL sehingga user dapat dengan mudah memasukkan nilai variabel baru ▪ Kurang aman ▪ Panjang URL string dibatasi maksimum sampai 2048 karakter ▪ Pengambilan variabel dengan $_GET ▪ Biasanya untuk input data melalui link ▪ Digunakan untuk mengirim data-data yang tidak sensitif

G. Cookie

Cookie adalah sepotong data yang disimpan pada komputer milik pengunjung dan digunakan oleh halaman web dalam mengingat suatu informasi. Dengan mengakses data yang disimpan pada komputer pengunjung inilah, suatu server bisa mendapatkan kembali informasi yang pernah dikirimkan ke client. Cookie biasanya digunakan untuk menciptakan sesi yang memungkinkan sesorang dapat masuk ke halaman-halaman lain tanpa perlu melakukan login kembali.

- **Menciptakan cookie**

Untuk menciptakan sebuah cookie, dapat menggunakan **setcookie()** dengan format sebagai berikut.

```
setcookie (nama_cookie, nilai_cookie);
```

Dalam hal ini, *nama_cookie* berupa string yang menyatakan nama cookie yang akan diciptakan dan *nilai_cookie* menyatakan nilai yang akan disimpan pada nama cookie.

Contoh:

```
<?php
setcookie("nama", "Demi Moore");
echo "Cookie telah terbentuk ";
?>
```

Script di atas digunakan untuk menciptakan cookie bernama **nama** dan isinya berupa "Demi Moore".

Perlu diketahui bahwa pemanggilan fungsi **setcookie()** perlu dilakukan sebelum teks HTML, mengingat cookie merupakan bagian dari header HTTP.

- **Mengakses cookie**

Untuk mengakses cookie, perlu menggunakan super global $_COOKIE.

```
$_COOKIE["nama_cookie"];
```

Contoh:

```
<?php
if isset($_COOKIE["mycookie"]){
	echo $_COOKIE["mycookie"];
}else{
	echo "Cookie Tidak Ada";
}
?>
```

Script di atas digunakan untuk menampilkan value dari cookie bernama "mycookie" apabila cookie tersebut ada (masih tersimpan dalam komputer), sedangkan bila sudah tidak ada maka tampilkan "Cookie Tidak Ada".

- **Menambahkan tanggal kadaluarsa**

Secara default, cookie hanya diingat sampai browser berakhir. Agar data tetap diingat walaupun browser berakhir, maka perlu mengatur waktu kadaluarsanya. Hal ini bisa dilakukan dengan menggunakan **setcookie()**, yaitu dengan memberikan waktu kadaluarsa pada argumen ketiga. Susunannya seperti berikut.

```
setcookie (nama_cookie, nilai_cookie, waktu_kadaluarsa);
```

Contoh:

```
<?php
setcookie("nama", "Demi Moore", time() + 3600);
echo "Cookie telah terbentuk ";
?>
```

Script di atas digunakan untuk menciptakan cookie bernama **nama** dan isinya berupa "Demi Moore" dengan masa kadaluarsa adalah 1 jam dimulai dari sekarang. **time()** digunakan untuk memperoleh waktu sekarang, 3600 menyatakan jumlah detik dalam 1 jam.

- **Menghapus cookie**

Untuk menghapus suatu cookie, gunakan **setcookie()** dengan menyebutkan nama cookie pada argumen pertama, string kosong pada argumen kedua dan kadaluarsa diset ke waktu sebelumnya (masa lampau) pada argumen ketiga.

Contoh:

```
<?php
// set kadaluarsa ke satu jam yang lalu
setcookie("nama", "", time() - 3600);
?>
<html>
<body>

<?php
echo "Cookie 'nama' telah dihapus.";
?>

</body>
</html>
```

H. Session

PHP menyediakan pustaka yang berguna untuk membentuk sebuah sesi dengan menggunakan **session**. Berbeda dengan cookie (yang menyimpan data pada client), session diimplementasikan dengan menyimpan data pada server. Dengan demikian, tidak perlu ada komunikasi bolak-balik antara web server dan client ketika web server membutuhkan data tersebut.

Setiap kali session dibentuk, akan terdapat referensi yang menunjuk ke session bersangkutan. Referensi ini dikenal dengan sebutan SID (*Session Identifier*) atau pengenal session. Nomor referensi ini dapat dibayangkan seperti nomor transaksi ketika mendapatkan faktur pembelian barang.

- **Mengawali dan mengakhiri session**

Salah satu cara untuk mengawali sebuah session yaitu dengan memanggil fungsi bernama **session_start()**, seperti berikut.

```
session_start();
```

Adapun utnuk mengakhiri sebuah session, dapat memanggil fungsi bernama **session_destroy()**, seperti berikut.

```
session_destroy();
```

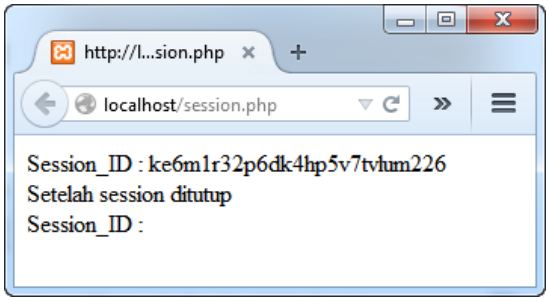
Session juga berakhir ketika browser ditutup. Untuk mengetahui pengenal session, dapat memanggil fungsi bernama **session_id()**. Fungsi ini akan memberikan string yang menyatakan pengenal session.

Contoh:

```
<?php
session_start();
echo "Session_ID : ".session_id()."<br>";
session_destroy();
```

```
echo "Setelah session ditutup <br>";
echo "Session_ID : ".session_id();
?>
```

Hasil pemanggilan script di atas, dapat dilihat pada Gambar 5 berikut ini.



Gambar 4. Mengawali dan mengakhiri session

- **Mendaftarkan variabel session**

Untuk mendaftarkan variabel session, cukup menggunakan superglobal **$_SESSION**. Misalnya variabel yang ingin digunakan bernama **var_x**, maka dapat dituliskan sebagai berikut.

varsession.php

```
<?php
session_start();

$_SESSION["var_x"] = "123456";

echo "Session_ID : ".session_id()."<br>";
echo "Isi Variabel Session : ".$_SESSION["var_x"];
?>
```

- **Mengakses variabel session**

Seandainya suatu variabel session telah dibentuk oleh suatu script, variabel ini dapat diakses oleh script lain dengan mula-mula melakukan pemanggilan fungsi **session_start()** kemudian variabel session dapat diperoleh melalui **$_SESSION**.

Contoh:

bacasession.php

```
<?php
session_start();
echo "Session_ID : ".session_id()."<br>";
echo "Variabel session var_x diakses dari bacasession.php: ";
if (isset($_SESSION["var_x"])){
        echo $_SESSION["var_x"];
}else{
        echo "Tidak ada";
}
?>
```

Perhatikan bahwa variabel **var_x** diakses melalui **$_SESSION**. Untuk mengetahui keberadaan variabel session tersebut, dapat mengujinya dengan fungsi **isset()**.

- **Membuang variabel session**

Untuk menghilangkan suatu variabel session, cukup menggunakan **unset()** terhadap variabel tersebut melalui **$_SESSION**.

Contoh:

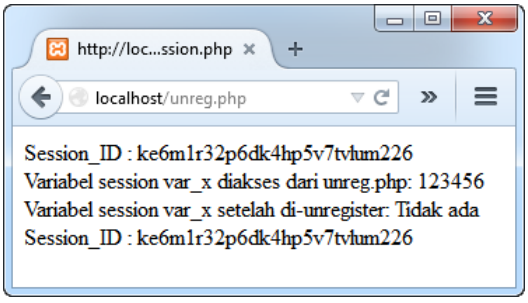
unreg.php

```
<?php
session_start();
echo "Session_ID : ".session_id()."<br>";
echo "Variabel session var_x diakses dari unreg.php: ".$_SESSION["var_x"]."<br>";

unset($_SESSION["var_x"]);

echo "Variabel session var_x setelah di-unregister: ";
if (isset($_SESSION["var_x"])){
	echo $_SESSION["var_x"]."<br>";
}else{
	echo "Tidak ada <br>";
}
echo "Session_ID : ".session_id();
?>
```

Hasil pemanggilan script di atas, dapat dilihat pada Gambar 6 berikut ini.



Gambar 5. Hasil pemanggilan unreg.php

I. Function

Script yang berukuran besar uumnya melibatkan fungsi-fungsi yang diciptakan sendiri oleh pemrogram. Fungsi adalah blok kode yang ditujukan untuk melaksanakan suatu tugas tertentu. Dengan menciptakan fungsi, yang dibuat sekali, fungsi dapat dipanggil berkali-kali di bagian mana pun dalam script.

Fungsi dapat dideklarasikan dengan menggunakan pernyataan **function**. Bentuk pendeklarasiannya adalah sebagai berikut.

```
function nama_function(parameter)
{
	..

	..

	return variabel;
}
```

Setiap function pasti dan harus memiliki nama function. Nama function ini nantinya akan dipanggil oleh program utama bila akan digunakan. Parameter di sini sifatnya optional (boleh ada, boleh tidak). Parameter ini ibaratnya input yang akan diolah oleh function. Sedangkan **return variabel** merupakan perintah untuk memberikan hasil setelah dikerjakan oleh function. Dalam hal ini perintah **return variabel** ini juga bersifat optional (boleh ada, boleh tidak). **return variabel** ini perlu digunakan bila hasil dari pengolahan function ini akan digunakan untuk proses yang lain dalam program. Sedangkan bila hasil dari function tidak akan digunakan oleh program, maka tidak perlu diberikan perintah ini.

Contoh:

Memerlukan return variabel:

```
<?php
function jumlah($a, $b){
    $c = $a + $b;
    return $c;
}
echo "Hasil penjumlahannya = ".jumlah(3, 5);
?>
```

Tidak memerlukan return variabel:

```
<?php
function tulis($x){
    echo "Anda menampilkan ". $x . "<br>";
}
tulis("Hello World..");
tulis("Apa kabar?");
?>
```

J. Teknik Modularitas

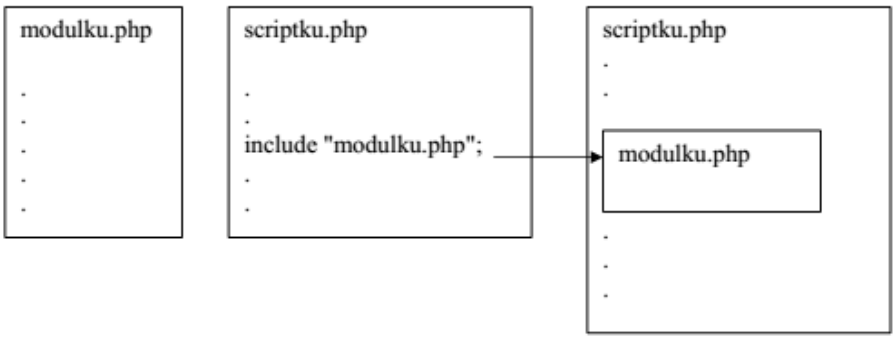
Konsep modularitas dalam pemrograman diperlukan untuk mempermudah dalam pengorganisasian script/program. Adapun prinsip dari konsep ini adalah meletakkan beberapa perintah yang menjalankan suatu tugas khusus ke dalam modul atau file script tersendiri. Setiap kali modul tersebut dibutuhkan, maka hanya menyisipkan modul tersebut ke dalam script yang sedang dibuat dengan cara memanggilnya.

Adapun cara memanggil modul dengan menggunakan perintah:

```
include "namafilemodul.php";
```

Perhatikan gambaran penggunaan konsep modularitas berikut ini:

Misalkan kita membuat modul dan disimpan dalam script bernama **modulku.php**. Selanjutnya kita juga membuat script tertentu katakanlah diberi nama file **scriptku.php**. Kebetulan dalam scriptku.php kita membutuhkan modulku.php, maka skema yang terjadi dapat dilihat pada Gambar 7 berikut ini.



Gambar 6. Gambaran penggunaan konsep modularitas

Dari skema di atas, jelas tampak bahwa bila kita memberikan perintah include "modulku.php"; dalam scriptku.php, maka akan identik dengan bila kita menyisipkan script yang ada dalam modulku.php ke dalam scriptku.php secara manual.

Contoh:

Dalam contoh ini, akan digunakan konsep modularitas untuk membuat script operasi aritmatika penjumlahan dari 2 buah bilangan. Untuk input bilangannya disimpan dalam modul tersendiri, misalkan dinamakan bilangan.php. Selanjutnya modul ini akan di-include-kan ke dalam script penjumlahan.

bilangan.php

```
<?php
$bil1 = 10;
$bil2 = 5;
?>
```

jumlah.php

```
<?php
include "bilangan.php";
$hasil = $bil1 + $bil2;
echo "Hasil penjumlahannya adalah : ".$hasil;
?>
```

Bentuk script jumlahkan.php di atas akan identik dengan script berikut ini.

```
<?php
$bil1 = 10;
$bil2 = 5;
$hasil = $bil1 + $bil2;
echo "Hasil penjumlahannya adalah : ".$hasil;
?>
```

Latihan:

1. Cobalah kode PHP yang ada dalam modul ini, kemudian perhatikan dan pahami maksud dari kode PHP tersebut.

Tugas:

1. Ibu ingin mengambil uang tabungan sejumlah Rp. 1.575.250,- yang dimilikinya di sebuah bank. Misalkan pada saat itu uang pecahan yang berlaku adalah Rp. 100.000,-; Rp. 50.000,-; Rp. 20.000,-; Rp. 5.000,-; Rp. 100,- dan Rp. 50. Dengan menggunakan script PHP, tentukan banyaknya masing-masing uang pecahan yang diperoleh ibu tadi!

   Lengkapi script berikut:

```
<?php
$jumlahUang = 1575250;
.
.
.
echo "Jumlah Rp. 100.000 : ".$a. "<br />";
echo "Jumlah Rp. 50.000 : ".$b. "<br />";
echo "Jumlah Rp. 20.000 : ".$c. "<br />";
echo "Jumlah Rp. 5.000 : ".$d. "<br />";
echo "Jumlah Rp. 100 : ".$e. "<br />";
echo "Jumlah Rp. 50 : ".$f. "<br />";
?>
```

   Keterangan:

   $a adalah variabel yang menyatakan jumlah pecahan Rp. 100.000,-
   $b adalah variabel yang menyatakan jumlah pecahan Rp. 50.000,-
   $c adalah variabel yang menyatakan jumlah pecahan Rp. 20.000,-
   $d adalah variabel yang menyatakan jumlah pecahan Rp. 5.000,-
   $e adalah variabel yang menyatakan jumlah pecahan Rp. 100,-
   $f adalah variabel yang menyatakan jumlah pecahan Rp. 50,-

2. Rancanglah sebuah form untuk pendaftaran online mahasiswa baru dalam universitas X. Data yang nantinya dimasukkan dalam form pendaftaran adalah: Nama Lengkap (text box), Tempat Lahir (text box), Tanggal Lahir (text box, format: dd-mm-yyyy), Alamat Rumah (text area), Jenis Kelamin (gunakan radio button: pria/wanita), Asal Sekolah (text box), Nilai UAN (text box). Tambahkan pula sebuah button submit dan button reset.

   Apabila data sudah diisi dan selanjutnya diklik tombol submit, maka kirim data ke sebuah script PHP (menggunakan method POST) untuk menampilkan apa yang telah diisikan dalam form tersebut. Berikut contoh output script apabila nama yang diisikan adalah "Joko".

```
Terimakasih Joko sudah mengisi form pendaftaran.

Nama Lengkap      : Joko
Tempat Lahir      : XXX
```

Tanggal Lahir	: TGL-BLN-THN
Alamat Rumah	: XXX
Jenis Kelamin	: Pria
Asal Sekolah	: XXX
Nilai UAN	: XXX

3. Terdapat bilangan-bilangan seperti: 283, 182, 381, 119, 391, 591, 123, 124, 284, 215, 312. Buatlah script dalam PHP untuk menampilkan bilangan terbesar, bilangan terkecil dan menghitung rata-rata dari bilangan-bilangan tersebut. Untuk menyelesaikan kasus ini, gunakan konsep array.

Petunjuk:

Uploadlah tugas tersebut pada folder “modul-5” pada web hosting masing-masing. Buatlah laporan pembahasan singkat mengenai kode yang telah dibuat dalam format pdf dan laporan pembahasan tersebut diupload pula pada folder “modul-5”.