

# BAB I.

## PHP – INTRODUCTION

### Apa Itu PHP?

Berdasarkan informasi dari situs resmi PHP, “PHP.net”, PHP (**PHP: Hypertext Preprocessor**) merupakan bahasa pemrograman web yang dapat disisipkan dalam script HTML. Banyak sintaks di dalamnya yang mirip dengan bahasa C, Java dan Perl. Tujuan dari bahasa ini adalah membantu para pengembang web untuk membuat web dinamis dengan cepat.

Ketika seseorang mengunjungi web berbasis PHP, web server akan memproses code-code PHP. Beberapa perintah atau code dari PHP tersebut selanjutnya ada yang diterjemahkan ke dalam HTML dan beberapa ada yang disembunyikan (misalnya proses kalkulasi dan operasi). Setelah diterjemahkan ke dalam HTML, web server akan mengirim kembali ke web browser pengunjung tersebut.

### Apa yang Bisa Dilakukan dengan PHP?

- Mengurangi waktu untuk membuat web berskala besar
- Mampu menciptakan web interaktif
- Menciptakan berbagai tool untuk keperluan online ([http://www.hotscripts.com/PHP/Scripts\\_and\\_Programs/](http://www.hotscripts.com/PHP/Scripts_and_Programs/))
- Mendukung e-commerce (shopping carts)

### Modal Dasar Mempelajari PHP

Sebelum mempelajari PHP, Anda harus menguasai

- HTML,
- Dasar-dasar pemrograman (C/C++ lebih baik)

### Aplikasi Yang Diperlukan

Untuk dapat bekerja dengan PHP, berikut ini adalah beberapa aplikasi yang diperlukan:

- Web server (Apache, IIS, Personal Web Server/PWS)
- PHP server (dapat didownload di PHP.net)
- Database server (MySQL, Interbase, MS SQL, dll)
- Web Editor (Dreamweaver, Frontpage, dll)

Anda dapat pula menggunakan tool aplikasi yang di dalamnya sudah terdapat web server (Apache), PHP server, dan MySQL yang terintegrasi menjadi satu. Tool tersebut dapat diinstal di PC sebagai sarana belajar PHP. Beberapa contoh tool tersebut diantaranya adalah Easyphp (Easyphp.org), PHPTriad, AppServe, dll.

PHP server dapat berjalan dengan baik di beberapa OS seperti Windows, Linux, dan Macintosh.

# BAB II

## PHP - SINTAKS

Kode-kode PHP dituliskan di antara tanda berikut ini:

```
<?php
```

```
...  
...  
...
```

```
?>
```

atau

```
<?
```

```
...  
...  
...
```

```
?>
```

Apabila Anda membuat kode PHP dan berencana akan mendistribusikan ke pihak/orang lain, maka usahakan untuk menggunakan sintaks `<?php ... ?>`. Hal ini dikarenakan untuk penggunaan kode yang menggunakan `<? ?>` terkadang tidak bisa dijalankan dalam server tertentu.

## Menyimpan File PHP

Apabila Anda memiliki kode PHP yang disisipkan dalam HTML dalam suatu file dan menginginkan web server dapat menjalankannya, maka file tersebut harus disimpan dalam ekstensi `.php`. Apabila Anda menyimpannya dengan ekstensi `.html` atau `.htm`, maka kode PHP tersebut tidak akan diproses dan akan ditampilkan dalam web browser seperti apa adanya (berupa kode-kode).

### Contoh:

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

# Semicolon (;)

Apabila Anda perhatikan contoh sebelumnya, maka terdapat tanda titik koma (semicolon) pada akhir perintah echo. Tanda semicolon merupakan penanda akhir dari statement PHP dan harus ada.

### Contoh:

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

# Pindah Spasi

Seperti halnya HTML, pergantian spasi dalam PHP tidak akan mempengaruhi tampilan hasilnya. Dengan kata lain, pergantian spasi akan diabaikan oleh PHP.

Perhatikan contoh berikut ini. Pada contoh tersebut diberikan tiga bentuk penulisan kode PHP yang berbeda namun akan dihasilkan tampilan yang sama dalam web browser.

### Contoh:

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

<html>
<head>
<title>My First PHP Page</title>
</head>
<body>
<?php
echo "Hello World! "; echo "Hello World! ";
?>
```

```
</body>
</html>

<html>
<head>
<title>Halaman PHP pertamaku</title>
</head>
<body>
<?php
echo "Hello World! ";

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

## Variabel

Misalkan dalam PHP kita akan menyimpan suatu nilai berupa angka atau string dalam suatu variabel, caranya adalah membuat nama variabel terlebih dahulu kemudian diberikan suatu assignment pada nilai yang diinginkan. Perhatikan sintaks berikut ini

```
$nama_variabel = nilai;
```

**Note: jangan lupa tanda dollar (\$)**

Contoh:

```
<?php
$hello = "Hello World!";
$sebuah_bilangan = 4;
$bilanganYangLain = 8;
?>
```

Dari contoh di atas tampak bahwa dalam PHP, nama variabel tidak perlu dideklarasikan terlebih dahulu seperti halnya bahasa Pascal atau C/C++.

## Aturan Penamaan Variabel

Berikut ini adalah beberapa aturan penulisan nama variabel:

- 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

## Echo

Seperti yang Anda lihat pada contoh-contoh kode PHP sebelumnya, bahwa perintah echo digunakan untuk menampilkan teks ke dalam browser. Suatu teks atau string dapat dituliskan di browser dengan langsung dituliskan dalam echo yang diapit oleh dua tanda petik ganda (quotes)

atau menyimpan string atau teks terlebih dahulu dalam suatu variabel kemudian dituliskan dalam echo. Berikut ini adalah contohnya:

Contoh:

```
<?php
$stringKu = "Hello!";
echo $stringKu;
echo "<h5>I love using PHP!</h5>";
?>
```

### Penting!!!

Hati-hati dalam penulisan suatu string yang di dalamnya terdapat tanda petik ganda (quotes) menggunakan echo. Dalam echo, tanda quotes merupakan penanda awal dan akhir teks/string yang akan ditulis dengan echo, sehingga Anda harus memperhatikan hal-hal berikut ini

- Jangan menggunakan tanda quotes di dalam teks yang akan ditulis dengan echo
- Apabila Anda tetap ingin menuliskan tanda quotes dalam teks yang akan ditulis dengan echo, maka berikan tanda slash “\” di depan quotes tersebut.
- Selain itu, dapat pula Anda gunakan tanda petik tunggal (apostrophes) untuk menggantikan tanda quotes pada teks.

Contoh:

```
<?php
echo "<font face='verdana' size='4'>I love using PHP!</font>";
?>

<?php
echo "<font face=\"verdana\" size=\"4\">I love using PHP!</font>";
?>

<?php
echo "<font face='verdana' size='4'>I love using PHP!</h5>";
?>
```

Pada contoh kode pertama di atas akan terjadi error karena dalam teks yang ditulis dalam echo terdapat tanda quotes. Sedangkan untuk kode kedua dan ketiga tidak terdapat error dan akan dihasilkan output yang sama di browser.

## Menampilkan Nilai Variabel dengan Echo

Nilai variabel dapat dengan mudah ditampilkan dengan menggunakan echo, baik nilai yang berupa bilangan maupun string. Berikut ini adalah contoh-contoh di antaranya:

Contoh:

```
<?php
$string_ku = "Hello.. Nama saya: ";
$bilangan_ku = 4;
$huruf_ku = "a";
echo $string_ku;
echo $bilangan_ku;
echo $huruf_ku;
```

?>

Catatan:

Untuk menampilkan nilai variabel dengan echo tanpa menggunakan tanda quotes.

Berikut ini contoh menampilkan gabungan suatu nilai dari variabel yang berupa string.

Contoh:

```
<?php
$string_ku = "Hello. Nama saya: ";
$baris_baru = "<br>";
echo $string_ku."Ari".$baris_baru;
echo "Hi, Nama saya Ari. Kamu siapa? ".$string_ku.$baris_baru;
echo "Hi, Nama saya Ari. Kamu siapa? ".$string_ku."Amalia";
?>
```

Untuk menggabungkan beberapa string menjadi satu digunakan operator dot (.)

## Komentar dalam PHP

Seperti halnya bahasa pemrograman yang lain, komentar dalam suatu kode PHP tidak akan dieksekusi. Terdapat dua cara memberikan komentar dalam PHP, yaitu

- Diberikan tanda // di depan teks komentar. Perintah ini hanya bisa berlaku untuk komentar dalam satu baris
- Diberikan 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!"; // Ini akan mencetak Hello World!
echo "<br>Psst...You can't see my PHP comments!"; // echo "nothing";
// echo "Namaku Faza!";
?>
```

```
<?php
/* Berikut ini adalah perintah
   untuk menuliskan Hello World */
echo "Hello World!";
/* echo "My name is Humperdinkle!";
echo "No way! My name is Ari-PHP Programmer!";
*/
?>
```

# BAB III

## PHP – OPERATOR

Dalam bahasa pemrograman secara umum, operator digunakan untuk memanipulasi atau melakukan proses perhitungan pada suatu nilai. Sampai saat ini, Anda telah mengenal operator “.” (menggabungkan string) dan “=” (proses assignment). Selain dua operator itu masih banyak jenis operator yang lain dalam PHP yaitu:

- Operator aritmatik
- Operator perbandingan
- Gabungan operator aritmatik dan assignment

### Operator Aritmatik

Berikut ini adalah tabel operator aritmatik, makna dan contohnya:

Operator	Makna	Contoh
+	Penjumlahan	2 + 4
-	Pengurangan	6 - 2
*	Perkalian	5 * 3
/	Pembagian	15 / 3
%	Modulus	43 % 10

Contoh:

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

### Operator Perbandingan

Perbandingan digunakan untuk menguji hubungan antara nilai dan atau variabel. Operator ini digunakan dalam suatu statement bersyarat yang selalu menghasilkan nilai TRUE atau FALSE.

Misalkan:

```
$x = 4; $y = 5;
```

berikut ini adalah beberapa contoh penggunaan operator perbandingan dan hasilnya.

Operator	Makna	Contoh	Hasil
==	Sama dengan	<code>\$x == \$y</code>	FALSE
!=	Tidak sama dengan	<code>\$x != \$y</code>	TRUE
<	Lebih kecil dari	<code>\$x &lt; \$y</code>	TRUE
>	Lebih besar dari	<code>\$x &gt; \$y</code>	FALSE
<=	Lebih kecil atau sama dengan dari	<code>\$x &lt;= \$y</code>	TRUE
>=	Lebih besar atau sama dengan dari	<code>\$x &gt;= \$y</code>	FALSE

## Kombinasi Operator Aritmatik dan Assignment

Dalam pemrograman seringkali dijumpai proses yang melibatkan proses increment. Misalkan kita menginginkan proses increment dengan tingkat kenaikan 1, maka perintah yang dituliskan dapat berupa

```
$counter = $counter + 1;
```

dalam PHP, perintah di atas dapat ditulis dalam satu perintah singkat sebagai

```
$counter += 1;
```

Dari contoh di atas tampak bahwa operator yang digunakan (+=) merupakan gabungan dari operator aritmatik dan assignment. Berikut ini adalah bentuk-bentuk operator lain jenis ini.

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

## Operator Pre/Post Increment dan Decrement

Operator jenis ini merupakan pengembangan dari operator jenis sebelumnya. 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 "$x;  
$x = 4;  
$x--;  
echo "$x;  
?>
```

# BAB IV

## MODULARITAS

Suatu pemrograman yang baik seharusnya program yang besar dipecah menjadi program-program yang kecil yang selanjutnya disebut modul. Modul-modul kecil tersebut dapat dipanggil sewaktu-waktu diperlukan. Dalam PHP juga mendukung konsep tersebut yang selanjutnya diberinama modularitas. Kita dapat menyisipkan isi suatu file/modul lain ke dalam file/modul tertentu.

Terdapat 2 perintah/function untuk hal tersebut dalam PHP yaitu menggunakan `include` dan `require`.

### Include()

Untuk memudahkan pemahaman, diberikan contoh berikut. Misalkan kita akan membuat menu link sejumlah 4 buah yaitu index, about, links, dan contact pada setiap halaman web yang kita buat. Teknik yang digunakan adalah membuat menu link dalam suatu modul tersendiri kemudian modul tersebut dipanggil pada setiap halaman web yang diinginkan terdapat menu link di dalamnya.

#### menu.php

```
<html>
<body>
<a href="index.php">Home</a> -
<a href="about.php">About Us</a> -
<a href="links.php">Links</a> -
<a href="contact.php">Contact Us</a> <br>
```

#### index.php

```
<?php
include( "menu.php" );
?>

<p>Ini adalah halaman index</p>
</body>
</html>
```

#### about.php

```
<?php
include( "menu.php" );
?>

<p>Ini adalah halaman about</p>
</body>
</html>
```

Dari teknik di atas tampak adanya kemudahan dalam membuat halaman web. Dalam hal ini, kita tidak perlu membuat menu link di setiap halaman web yang ada. Bayangkan seandainya kita

punya halaman web sejumlah 100 buah yang kesemuanya ingin diberi menu link tanpa menggunakan teknik di atas, tentu hal tersebut sangat merepotkan.

Meskipun secara teknis, kode pembangun web dipecah dalam modul-modul, namun ketika di browser akan terlihat menyatu. Berikut ini adalah kode HTML yang dihasilkan oleh browser ketika membuka halaman web index.php

```
<html>
<body>
<a href="index.php">Home</a> -
<a href="about.php">About Us</a> -
<a href="links.php">Links</a> -
<a href="contact.php">Contact Us</a> <br>
<p>Ini adalah halaman index</p>
</body>
</html>
```

## Require()

Cara penggunaan maupun fungsi dari require() sama dengan include(). Jadi apa perbedaannya? Sebaiknya mana yang kita gunakan? Perhatikan contoh berikut ini

```
<?php
include("noFileExistsHere.php");
echo "Hello World!";
?>
```

dengan asumsi bahwa file noFileExistxHere.php tidak ada.

Maka dengan menggunakan include() akan dihasilkan tampilan:

**Warning:** main(noFileExistsHere.php): failed to open stream: No such file or directory in **include.php** on line 2

**Warning:** main(): Failed opening 'noFileExistsHere.php' for inclusion (include\_path='.:usr/lib/php:/usr/local/lib/php') in **include.php** on line 2

Hello World

Selanjutnya kita akan gunakan require().

```
<?php
require("noFileExistsHere.php");
echo "Hello World!";
?>
```

dan hasilnya

**Warning:** main(noFileExistsHere.php): failed to open stream: No such file or directory in **require.php** on line 2

**Fatal error:** main(): Failed opening required 'noFileExistsHere.php' (include\_path='.:usr/lib/php:/usr/local/lib/php') in **require.php** on line 2

Bandingkan kedua hasil di atas, khususnya yang tercetak merah. Pada `include()`, error yang dihasilkan hanya berupa `Warning` saja dan statement berikutnya masih dapat dijalankan. Hal ini terlihat bahwa teks `Hello World!` Masih ditampilkan di browser. Sedangkan pada `require()`, error yang dihasilkan berupa `Fatal Error`. Dengan demikian statement selanjutnya tidak akan dijalankan.

Disarankan agar Anda menggunakan `require()` dengan harapan bahwa kode PHP yang Anda buat tidak akan diproses apabila terdapat file yang hilang atau tidak ada.

# BAB V

## ARRAY

Dalam PHP, indeks untuk array dapat berupa numerik dan dapat pula berupa suatu nilai atau yang sering disebut dengan array asosiatif.

### Array Berindeks Numerik

Sintaks umum untuk menyatakan suatu array berindeks numerik beserta nilainya adalah

```
$nama_array[$key] = value;
```

Dalam hal ini \$key berupa bilangan bulat mulai dari 0, 1, 2, ...

Contoh:

```
$karyawan[0] = "Bob";  
$karyawan[1] = "Sally";  
$karyawan[2] = "Charlie";  
$karyawan[3] = "Clare";
```

Sedangkan berikut ini adalah contoh untuk menampilkan nilai dari suatu array berindeks numerik.

```
<?  
$karyawan[0] = "Bob";  
$karyawan[1] = "Sally";  
$karyawan[2] = "Charlie";  
$karyawan[3] = "Clare";  
  
echo "Berikut ini adalah 2 orang karyawan saya, yaitu "  
. $karyawan[0] . " & " . $karyawan[1];  
echo "<br>Dua orang karyawan saya yang lain adalah "  
. $karyawan[2] . " & " . $karyawan[3];  
?>
```

### Array Asosiatif

Untuk array asosiatif, sintaksnya sama dengan array berindeks numerik namun perbedaannya adalah pada \$key. Pada array asosiatif, \$key dapat berupa suatu string. Berikut ini adalah contohnya.

```
$gaji["Bob"] = 2000;  
$gaji["Sally"] = 4000;  
$gaji["Charlie"] = 600;  
$gaji["Clare"] = 0;
```

dan berikut ini adalah contoh kode untuk menampilkan nilai dari array asosiatif

```
<?  
$gaji["Bob"] = 2000;  
$gaji["Sally"] = 4000;
```

```
$gaji["Charlie"] = 600;  
$gaji["Clare"] = 0;  
  
echo "Bob digaji - $" . $gaji["Bob"] . "<br>";  
echo "Sally digaji - $" . $gaji["Sally"] . "<br>";  
echo "Charlie digaji - $" . $gaji["Charlie"] . "<br>";  
echo "dan Clare digaji - $" . $gaji["Clare"];  
?>
```

# BAB VI

## STATEMENT KONTROL

### Statement IF ...

Sintaks dari statement tersebut dalam PHP adalah

```
if (syarat)
{
    statement1;
    statement2;
    .
    .
}
```

Untuk menyatakan syarat, biasanya digunakan operator perbandingan seperti yang telah dibahas sebelumnya. Apabila syarat bernilai TRUE maka statement-statement yang diapit dengan tanda kurung kurawal akan dijalankan. Bentuk lain dari sintaks IF adalah

```
if (syarat)
{
    statement1;
    statement2;
    .
    .
}
else
{
    statement3;
    statement4;
    .
    .
}
```

Untuk sintaks kedua di atas, statement3, statement4, dst akan dijalankan apabila syarat bernilai FALSE.

Contoh:

```
<?
$my_name = "nada";

if ($my_name == "nada")
{
    echo "Your name is ".$my_name."!"<br>";
}
echo "Welcome to my homepage!";
?>
```

Contoh:

```
<?
$number = 3;

if ($number == 4)
{
    echo "Benar";
}
else
{
    echo "Salah";
}
?>
```

Terdapat pula bentuk sintaks berikutnya dari IF ... yaitu dengan ditambahkan `elseif`

```
if (syarat1)
{
    statement11;
    statement12;
    .
    .
}
elseif (syarat2)
{
    statement21;
    statement22;
    .
    .
}
.
.
else
{
    statement1;
    statement2;
    .
    .
}
```

Jika syarat1 bernilai TRUE, maka statement11, statement12 dst akan dijalankan. Sedangkan jika syarat1 FALSE maka selanjutnya akan dicek untuk syarat2. Jika syarat2 TRUE maka statement21, statement22, dst akan dijalankan, sedangkan jika syarat2 FALSE akan dicek syarat berikutnya (jika masih ada). Statement1, statement2, dst baru akan dijalankan apabila semua syarat sebelumnya bernilai FALSE.

Contoh:

```
<?
$karyawan = "Bob";
if($karyawan == "Tanner")
{
    echo "Hello Tanner!";
}
elseif($karyawan == "Bob")
{
    echo "Hello Bob!";
}
```



```
}  
else  
{  
    echo "Hello!";  
}
```

## Statement SWITCH

Sintaks dari statement ini adalah

```
switch (variabel)  
{  
    case option1:  
        statement11;  
        statement12;  
        .  
        .  
        break;  
    case option2:  
        statement21;  
        statement22;  
        .  
        .  
        break;  
    .  
    .  
    default:  
        statementdefault1;  
        statementdefault2;  
        .  
        .  
        break;  
}
```

Pada sintaks di atas, nilai dari variabel akan dicek pada setiap option yang ada (terletak di bagian case). Jika ada option yang sama dengan nilai variabel, maka statement-statement di bawah option tersebutlah yang akan dijalankan. Bagian default adalah optional (boleh ada, boleh tidak).

Contoh:

```
<?php  
$tujuan = "Tokyo";  
echo "Biaya Perjalanan Menuju $tujuan adalah ";  
switch ($tujuan){  
    case "Las Vegas":  
        echo " $500";  
        break;  
    case "Amsterdam":  
        echo " $1500";  
        break;  
    case "Egypt":  
        echo " $1750";  
        break;  
    case "Tokyo":  
        echo " $900";  
        break;  
}
```

```
        case "Caribbean Islands":
            echo " $700";
            break;
    }
?>
```

Contoh:

```
<?php
$tujuan = "New York";
echo "Biaya Perjalanan Menuju $tujuan adalah ";
switch ($tujuan){
    case "Las Vegas":
        echo " $500";
        break;
    case "Amsterdam":
        echo " $1500";
        break;
    case "Egypt":
        echo " $1750";
        break;
    case "Tokyo":
        echo " $900";
        break;
    case "Caribbean Islands":
        echo " $700";
        break;
    default:
        echo " $100";
        break;
}
<?
```

## Statement WHILE

Statement ini digunakan untuk mengerjakan suatu statement secara berulang-ulang sampai suatu syarat dipenuhi. Sintaksnya adalah

```
while (syarat)
{
    statement;
    statement;
    .
    .
}
```

Pada sintaks di atas, selama syarat bernilai TRUE maka statement-statement di dalam while akan terus dijalankan secara berulang-ulang. Perulangan baru akan berhenti apabila syarat bernilai FALSE. Sebelum statement yang diulang-ulang dilakukan, terlebih dahulu akan dicek syarat nya apakah bernilai TRUE atau FALSE. Apabila TRUE maka statement akan dijalankan. Sedangkan apabila FALSE, perulangan akan langsung berhenti. Dengan kata lain, statement dalam WHILE bisa jadi tidak akan pernah dilakukan, yaitu apabila syaratnya langsung bernilai FALSE.

Contoh:

```
<?
$harga_sikat = 1500;
$jumlah_sikat = 10;

echo "<table border=\"1\" align=\"center\">";
echo "<tr><td><b>Jumlah Sikat</b></td>";
echo "<td><b>Harga</b></td></tr></td>";
while ( $jumlah_sikat <= 100 )
{
    echo "<tr><td>";
    echo $jumlah_sikat;
    echo "</td><td>";
    echo "Rp. ".$harga_sikat * $jumlah_sikat;
    echo "</td></tr>";
    $jumlah_sikat = $jumlah_sikat + 10;
}
echo "</table>";
?>
```

Kode di atas akan menampilkan hasil di browser berupa tabel yang berisi jumlah sikat dan harganya, dengan asumsi harga sebuah sikat adalah Rp. 1.500. Jumlah sikat yang ditampilkan adalah kelipatan 10 dengan batas sampai 100 buah.

## Statement FOR

Statement FOR mirip dengan WHILE yang memiliki sintaks berikut ini

```
for (inisialisasi counter; syarat; increment/decrement counter)
{
    statement;
    .
    .
}
```

Untuk memperjelas pemahaman tentang FOR, berikut ini adalah contoh kode dengan for untuk menghasilkan tampilan yang sama dengan contoh while sebelumnya (tentang jumlah sikat dan harganya). Coba bandingkan dengan kode contoh while sebelumnya.

Contoh:

```
<?
$harga_sikat = 1500;

echo "<table border=\"1\" align=\"center\">";
echo "<tr><td><b>Jumlah Sikat</b></td>";
echo "<td><b>Harga</b></td></tr>";
for ($jumlah_sikat = 10; $jumlah_sikat <= 100; $jumlah_sikat+=10)
{
    echo "<tr><td>";
    echo $jumlah_sikat;
    echo "</td><td>";
    echo "Rp. ".$harga_sikat * $jumlah_sikat;
    echo "</td></tr>";
}
```

```
echo "</table>";  
?>
```

## Statement Foreach

Misalkan Anda punya data berupa array asosiatif yang akan diproses secara berulang-ulang, maka PHP menyediakan statement foreach yang mudah digunakan.

Sintaksnya adalah:

```
foreach(variabelarray as kunci => value)  
{  
    statement;  
    .  
    .  
}
```

Sebagai contoh, misalkan Anda memiliki 5 orang karyawan dengan usianya masing-masing yang ditulis dalam kode PHP sebagai berikut

```
$UsiaKaryawan["Lisa"] = "28";  
$UsiaKaryawan["Jack"] = "16";  
$UsiaKaryawan["Ryan"] = "35";  
$UsiaKaryawan["Rachel"] = "46";  
$UsiaKaryawan["Grace"] = "34";
```

Berikut ini adalah contoh kode PHP yang akan menampilkan semua karyawan beserta usianya dengan menggunakan foreach.

```
<?  
$UsiaKaryawan["Lisa"] = "28";  
$UsiaKaryawan["Jack"] = "16";  
$UsiaKaryawan["Ryan"] = "35";  
$UsiaKaryawan["Rachel"] = "46";  
$UsiaKaryawan["Grace"] = "34";  
  
foreach($UsiaKaryawan as $Nama => $umur)  
{  
    echo "Nama Karyawan: $Nama, Usia: $umur". " th <br>";  
}  
?>
```

## Statement DO WHILE

Statement ini merupakan bentuk modifikasi dari WHILE. Sintaksnya adalah sebagai berikut

```
do  
{  
    statement;  
    .  
    .  
}  
while (syarat);
```

Coba bandingkan dengan sintaks WHILE sebelumnya. Dilihat dari posisi statement yang diulang, posisi statement yang diulang pada DO WHILE terletak di atas syarat. Dengan demikian, sebelum syarat dicek TRUE atau FALSE nya, statement akan dikerjakan terlebih dahulu. Sedangkan pada WHILE, sebelum statement yang diulang dikerjakan, terlebih dahulu syarat akan dicek.

Prinsip kerja DO WHILE sama dengan WHILE yaitu statement akan terus dikerjakan selama syarat bernilai TRUE dan perulangan akan berhenti apabila FALSE.

Perhatikan contoh berikut ini yang membandingkan DO WHILE dengan WHILE

Contoh:

```
<?
$kue = 0;
while($kue > 1)
{
    echo "Mmmm...Aku suka kue! Nyam..nyam..nyam..";
}
?>

<?
$kue = 0;
do
{
    echo "Mmmm... Aku suka kue! Nyam..nyam..nyam..";
} while ($kue > 1);
?>
```

Pada kode WHILE, teks "Mmmm.... " dst tidak akan ditampilkan karena syaratnya langsung bernilai FALSE (perulangan berhenti). Sedangkan pada DO WHILE, teks akan ditampilkan dahulu kemudian perulangan berhenti (syarat bernilai FALSE).

# BAB VII

## BEKERJA DENGAN FORM

Setelah kita belajar dasar-dasar perintah PHP, sekarang saatnya mengaplikasikannya pada aplikasi web. Biasanya PHP digunakan sebagai pengolah data yang diinputkan melalui form yang dibuat dengan HTML. Sebagai contoh, andaikan Anda memiliki toko virtual dalam web yang menjual alat-alat tulis seperti pensil, buku tulis, dan ballpoint. Berikut ini adalah salah satu bentuk kode HTML yang digunakan untuk membuat form pemesanan pembelian barang-barang tersebut.

```
<html><body>
<h2>Toko Alat Tulis Amalia</h2>
<form action="proses.php" method="post">
<select name="barangpesanan">
<option>Pensil</option>
<option>Buku Tulis</option>
<option>Ballpoint</option>
</select>
Jumlah pesanan: <input name="jumlah" type="text">
<input type="submit" value="Submit">
</form>
</body></html>
```

Seperti yang Anda lihat pada kode HTML di atas, perintah `action="proses.php"` digunakan untuk mengarahkan ke file PHP yang digunakan untuk memproses barang pembelian ketika tombol submit ditekan. Dalam form yang dihasilkan dari kode di atas terdapat 2 buah komponen input yaitu berbentuk combobox dan textbox. Untuk combobox, diberi nama "barangpesanan" (perhatikan perintah `<select name="barangpesanan">`) dan textbox diberi nama "jumlah" (perhatikan perintah `<input name="jumlah" type="text">`).

Penjelasan mengenai `method="post"` akan dijelaskan pada bab berikutnya.

Sedangkan berikut ini adalah salah satu contoh kode PHP untuk memproses input dari form di atas. Kode PHP ini disimpan dengan nama `proses.php`.

```
<html><body>
<?php
$jumlah = $_POST['jumlah'];
$barangpesanan = $_POST['barangpesanan'];

echo "Anda memesan ". $jumlah . " " . $barangpesanan . ".<br>";
echo "Terima kasih atas kesediaan Anda memesan barang dari kami!";
?>
</body></html>
```

Kalau Anda perhatikan, terdapat keterkaitan perintah `$_POST['xxx'];` dengan "xxx" pada `name = "xxx"` (nama komponen input).

# BAB VIII

## POST AND GET METHOD

Pada contoh sebelumnya (Bab 7), kita mengirim data input dari form menuju ke file PHP untuk diproses menggunakan metode `post`. Selain metode tersebut, terdapat pula metode `get`. Lantas perbedaannya apa? Kapan kita gunakan metode `post` atau `get`? Itulah yang akan dibahas pada bab ini.

Untuk melihat perbedaan `post` dan `get`, kita akan sedikit mengubah file HTML form dan file PHP `proses.php` sebelumnya.

```
<html><body>
<h2>Toko Alat Tulis Amalia</h2>
<form action="proses.php" method="get">
<select name="barangpesanan">
<option>Pensil</option>
<option>Buku Tulis</option>
<option>Ballpoint</option>
</select>
Jumlah pesanan: <input name="jumlah" type="text">
<input type="submit" value="Submit">
</form>
</body></html>
```

dan isi `proses.php` nya adalah

```
<html><body>
<?php
$jumlah = $_GET['jumlah'];
$barangpesanan = $_GET['barangpesanan'];

echo "Anda memesan ". $jumlah . " " . $barangpesanan . "<br>";
echo "Terima kasih atas kesediaan Anda memesan barang dari kami!";
?>
</body></html>
```

Perbedaan kode HTML dan PHP di atas dengan sebelumnya adalah yang dicetak merah. Metode pengiriman data input dari form menggunakan `get`, dan dalam `proses.php` `$_POST` diganti dengan `$_GET`.

Apabila aplikasi di atas dijalankan, maka secara sekilas hasil yang tampak sama dengan ketika digunakan metode `post`. Namun, coba perhatikan URL yang tampak ketika `proses.php` muncul. Pada URL tersebut terdapat tambahan `?barangpesanan=...&jumlah=...` setelah nama file (`proses.php`). Titik-titik tersebut akan diisi dengan data sesuai dengan yang diinputkan pada form.

Coba bandingkan dengan URL ketika digunakan metode `post`. Data isian pada form tidak ditampilkan pada URL. Sehingga inilah perbedaan antara keduanya.

Dengan demikian, hendaknya kita jangan menggunakan metode `get` ketika akan memproses data input melalui form. Bayangkan seandainya form tersebut digunakan untuk login atau untuk keperluan yang menyangkut privasi. Apabila Anda gunakan metode `get`, maka semua input data

akan ditampilkan pada URL. Bisa-bisa password Anda akan kelihatan di URL (jika terdapat input password ketika login).

Untuk metode get, biasanya digunakan untuk input data melalui link (bukan melalui form). Untuk contoh aplikasinya dapat dilihat pada contoh-contoh aplikasi pada bab-bab berikutnya.



# BAB IX

## FUNCTION

Sebuah function merupakan sebuah nama yang kita berikan pada suatu blok program yang sewaktu-waktu dapat kita panggil dan gunakan. Sebuah function dapat diletakkan di bagian manapun, bisa di awal, tengah, dan akhir dari keseluruhan bagian kode PHP.

Berikut ini adalah contoh membuat sebuah function sederhana yang di dalamnya tidak ada statementnya.

Contoh:

```
<?php
function myCompanyMotto()
{
}
myCompanyMotto();
?>
```

Pada contoh di atas, `myCompanyMotto` merupakan nama function. Nama function inilah yang dapat dipanggil sewaktu-waktu diperlukan. Aturan membuat nama function sama dengan ketika membuat nama variabel (lihat di Bab II). Statement/perintah dari function dituliskan di dalam kurung kurawal `{ }`. Sedangkan perintah `myCompanyMotto();` bagian paling bawah dari kode di atas merupakan cara memanggil function. Perhatikan contoh berikutnya:

Contoh:

```
<?php
function myCompanyMotto()
{
    echo "Sabar adalah bagian dari keimanan";
}
myCompanyMotto();
?>
```

Pada contoh tersebut, terdapat perintah `echo` di dalam function. Sehingga begitu nama function dipanggil, PHP akan menampilkan teks yang di-echo-kan tersebut.

Sebuah function dapat dipanggil berulang-ulang, seperti pada contoh berikut.

```
<?php
function myCompanyMotto()
{
    echo "Sabar adalah bagian dari keimanan";
}

echo "Selamat datang di PT. Nada Corp. <br>";
myCompanyMotto();
echo "Terima kasih atas kunjungan Anda<br>";
echo "dan ingatlah selalu... <br>";
```

```
myCompanyMotto();  
?>
```

## Fungsi dengan Parameter

Contoh function sebelumnya tidak menggunakan parameter. Peran parameter adalah sebagai input untuk function yang selanjutnya diolah oleh function tersebut. Berikut ini contoh penggunaan parameter pada function.

```
<?php  
function UcapanSalam($nama)  
{  
    echo "Hallo ". $nama . "!"<br>;  
}  
?>
```

Pada contoh di atas, variabel `$nama` merupakan parameter dari function. Nilai dari variabel tersebut akan ditambahkan pada string yang di-echo-kan.

Selanjutnya akan diberikan contoh penggunaan function dengan parameter.

```
<?php  
function UcapanSalam($nama)  
{  
    echo "Hallo ". $nama . "!"<br>;  
}  
UcapanSalam("Agus");  
UcapanSalam("Ahmad");  
UcapanSalam("Budi");  
UcapanSalam("Fauzan");  
?>
```

Jumlah parameter dari function boleh lebih dari satu. Untuk memisahkan antar parameter digunakan tanda koma. Berikut ini contohnya.

```
<?php  
function UcapanSalam($kepada, $dari)  
{  
    echo $dari . " mengucapkan salam kepada ". $kepada . "<br>;  
}  
UcapanSalam("Ari", "Amalia");  
UcapanSalam("Amalia", "Nada");  
UcapanSalam("Nada", "Faza");  
UcapanSalam("Fauzan", "Ari");  
?>
```

## Pengembalian Nilai (*Return Value*)

Sebuah function juga dapat mengembalikan suatu nilai. Function hanya dapat mengembalikan sebuah nilai saja. Nilai yang dikembalikan dapat berupa suatu bilangan (bulat, real), string, maupun array, dll.

Berikut ini adalah contoh penggunaan function yang mengembalikan nilai.

```
<?php
```

```
function Jumlahkan($x, $y){  
    $hasil = $x + $y;  
    return $hasil;  
}  
$bil = 0;  
echo "Nilai bil mula-mula adalah ". $bil . "<br>";  
$bil = Jumlahkan(3, 4);  
echo "Nilai bil setelah memanggil function adalah " . $bil . "<br>";  
?>
```

Function `Jumlahkan()` di atas mengembalikan nilai dari variabel `$hasil` yang merupakan hasil penjumlahan dari nilai `$x` dan `$y`. Sedangkan perintah `$bil = Jumlahkan(3, 4);` bermakna nilai yang dikembalikan function `Jumlahkan(3, 4)` disimpan pada variabel `$bil` (dalam hal ini nilai `$bil` adalah 7).

# BAB X

## OPERASI FILE

Pada bab ini akan dipaparkan bagaimana menggunakan perintah PHP untuk melakukan operasi file mulai dari proses membuka dan menutup file. Setelah itu dilanjutkan dengan proses membaca, menulis, menambah isi, menghapus dan meng-upload file.

### Membuka File

Secara umum terdapat 3 cara membuka file, yaitu membuka file hanya untuk dibaca (read: 'r'), hanya untuk ditulis baru (write: 'w'), dan hanya untuk ditambahi isinya (append: 'a'). Selain 3 cara membuka file tersebut, terdapat pula cara lain membuka file yaitu dapat dibaca dan ditulis (read/write: 'r+'), serta dapat dibaca dan ditambahi isinya (append: 'a+').

Berikut ini adalah contoh kode PHP untuk membuka file

```
<?
$nama_file = "test.txt";
$fh = fopen($nama_file, 'X') or die("File tidak bisa dibuka");
fclose($fh);
?>
```

dengan 'X' dapat diganti dengan 'w', 'r', 'a', 'r+', 'a+'.

Apabila nama file yang akan dibuka ternyata salah, atau letak filenya yang tidak tepat, maka function `die()` yang akan dijalankan. Function `die()` akan menampilkan teks sebagai peringatan apabila proses membuka file gagal.

### Menutup File

Setelah file dibuka, hendaknya file tersebut juga ditutup ketika pemrosesan selesai. File yang tidak ditutup kemungkinan dapat terjadi kerusakan pada strukturnya. Berikut ini adalah contoh kode PHP untuk menutup file yang telah dibuka

```
$NamaFile = "testFile.txt";
$FileHandle = fopen($NamaFile, 'w') or die("File tidak bisa dibuka");
fclose($FileHandle);
```

File yang telah ditutup tidak bisa untuk dibaca, ditulis, dan ditambah. Untuk bisa melakukan hal itu kembali, file terlebih dahulu harus dibuka lagi seperti sebelumnya.

### Menulis ke File

File yang telah dibuka dapat ditulis dengan data di dalamnya. Berikut ini adalah contoh kode PHP untuk menulis suatu string ke dalam file.

```
<?
$FileKu = "testFile.txt";
```

```
$FileHandle = fopen($Fileku, 'w') or die("File gagal dibuka");
$DataString = "Hallo Amalia... \n";
fwrite($FileHandle, $DataString);
$DataString = "Hallo Faza dan Nada... \n";
fwrite($FileHandle, $DataString);
fclose($FileHandle);
?>
```

Apabila Anda menggunakan mode 'w' pada `fopen()`, dan selanjutnya Anda menuliskan data pada file, maka isi file yang lama (jika sebelumnya terdapat isi pada file tersebut) akan terhapus dan diganti dengan isi yang baru. Sedangkan apabila Anda menginginkan data yang lama pada suatu file tidak dihapus, maka Anda gunakan mode append 'a+' atau 'a'.

## Membaca Isi File

Misalkan kita memiliki file dengan nama `test.txt` yang isinya adalah sbb:

```
Selamat berjumpa lagi kawan!!
Senang bertemu Anda.
```

Kita dapat menggunakan kode PHP untuk membaca file tersebut dan selanjutnya hasil pembacaan dapat ditampilkan di browser atau diproses lebih lanjut. Berikut ini adalah contoh kode pembacaannya.

```
<?
$FileKu = "test.txt";
$FileHandle = fopen($FileKu, 'r');
$Data = fread($FileHandle, 5);
fclose($FileHandle);
echo $Data;
?>
```

Apabila kode di atas dijalankan, maka pada browser hanya akan menampilkan teks

```
Selam
```

Hal ini dikarenakan adanya nilai 5 pada `fread()`. Nilai tersebut menyatakan jumlah karakter awal yang dibaca dari file. Supaya seluruh isi file dapat dibaca, gunakan function `filesize()`.

```
<?
$FileKu = "test.txt";
$FileHandle = fopen($FileKu, 'r');
$Data = fread($FileHandle, filesize($FileKu));
fclose($FileHandle);
echo $Data;
?>
```

Setelah Anda lihat di browser, ternyata ganti baris pada isi file `test.txt` diabaikan. Meskipun teks dalam file tersebut ditulis dalam 2 baris, oleh perintah `fread()` akan dibaca dalam satu baris. Hal ini dikarenakan dalam file `test.txt` tidak terdapat tag html `<br>` untuk pindah baris.

Anda juga dapat menggunakan function `fgetc()` untuk membaca file. Function ini membaca isi file karakter demi karakter. Berikut ini adalah contoh penggunaannya.

```
<?
$FileKu = "test.txt";
```

```
$FileHandle = fopen($FileKu, 'r');
while(!feof($FileHandle))
{
    $Data = fgetc($FileHandle);
    echo $Data;
}
fclose($FileHandle);
?>
```

Keterangan:

Function `fgetc()` pada kode di atas diletakkan dalam perulangan. Function `feof()` digunakan untuk menyelidiki apakah pointer sudah berada di akhir dari file (end of file) atau belum. Selama belum end of file, proses pembacaan karakter akan berjalan terus. Setiap kali pembacaan, karakter yang dibaca akan ditampilkan di browser.

## Menghapus File

Dalam PHP, function untuk menghapus file adalah `unlink()`. Berikut ini contohnya.

```
<?
$myFile = "testFile.txt";
unlink($myFile);
?>
```

## Menambah Isi File

Maksud dari menambah isi file di sini adalah, menambah data baru pada file (diasumsikan data sudah ada sebelumnya). Untuk menambah isi file dalam PHP, mode pembukaan file nya menggunakan 'a' atau 'a+' dan `fwrite()` untuk menulis data ke dalam file. Berikut ini adalah contohnya.

Misalkan kita mempunyai file `buah.txt` yang isinya adalah sbb:

```
Jeruk
Apel
Mangga
```

Selanjutnya kita punya kode PHP sbb:

```
<?
$myFile = "buah.txt";
$fh = fopen($myFile, 'a') or die("File tidak bisa dibuka");
$buah1 = "Anggur\n";
fwrite($fh, $buah1);
$buah2 = "Nanas\n";
fwrite($fh, $buah2);
fclose($fh);
?>
```

Kode di atas akan menambahkan 2 buah data baru ke file `buah.txt`. Perintah `\n` (new line) digunakan untuk ganti baris pada file.

## Upload File

Di beberapa aplikasi web, sering kita menjumpai proses upload file ke server. Berikut ini akan dibahas cara melakukan hal itu.

Langkah pertama untuk membuat aplikasi web guna upload file adalah membuat formnya terlebih dahulu. Berikut ini adalah salah satu contoh form dalam bentuk HTML

```
<form enctype="multipart/form-data" action="upload.php" method="post">
  <input type="hidden" name="MAX_FILE_SIZE" value="30000" />
  Nama File : <input name="userfile" type="file" />
  <input type="submit" value="Upload" />
</form>
```

Pada form di atas, kita membatasi ukuran file yang dapat diupload adalah 30 Kb.

Selanjutnya kita membuat file `upload.php` untuk proses uploadnya.

```
<?php

$uploadaddir = 'uploads/';
$uploadfile = $uploadaddir . $_FILES['userfile']['name'];

if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)) {
    echo "File telah berhasil diupload";
} else {
    echo "File gagal diupload";
}

?>
```

# BAB XI

## OPERASI STRING

### strpos

Function strpos() digunakan untuk menentukan posisi suatu substring dari sederetan string. Function ini akan mereturn bilangan integer yang merupakan urutan posisi substring tersebut.

Contoh:

```
<?
$stringku = "1234567890";

$posisi = strpos($stringku, "5");
echo "Posisi dari karakter 5 dalam string adalah $posisi";
?>
```

Script di atas akan menghasilkan posisi karakter '5' adalah di 4.

NB:

Ingat bahwa index dari array dalam PHP dimulai dari index ke – 0.

Kelemahan dari strpos() adalah bahwa function tersebut hanya dapat menentukan posisi suatu substring pada pemunculan pertama. Pada pemunculan substring pada posisi yang berikutnya tidak akan terdeteksi. Meskipun demikian, Anda masih tetap menggunakannya untuk mencari posisi yang lain dari suatu substring dengan sedikit melakukan pemrograman.

Contoh:

```
<?
$stringku = "1234567890123456789012345678901234567890";
$offset = 0;
$counter = 1;

while($offset = strpos($numberedString, "5", $offset + 1)){
    $counter++;
    echo "<br>Karakter 5 ke-$counter ada posisi - $offset";
}
?>
```

### str\_replace

Function ini memiliki peran yang sama seperti ketika kita menggunakan fasilitas Replace All pada MS. Word. Function ini akan menggantikan suatu string dengan string yang lain.

Contoh:

```
$stringawal = "selamat datang di halaman web ini";
$ubahstring = str_replace("web", "website", $stringawal);
echo "$ubahstring";
```



Function `str_replace()` juga dapat menggantikan beberapa string secara simultan sekaligus. Berikut ini contohnya:

```
<?
$rawstring = "Welcome Birmingham parent! <br>
    Your offspring is a pleasure to have!
    We believe pronoun is learning a lot.<br>
    The faculty simple adores pronoun2 and you can often hear
    them say \"Attah sex!\"<br>";

$placeholders = array('offspring', 'pronoun', 'pronoun2', 'sex');
$malevals = array('son', 'he', 'him', 'boy');
$femalevals = array('daughter', 'she', 'her', 'girl');

$malestr = str_replace($placeholders, $malevals, $rawstring);
$femalestr = str_replace($placeholders, $femalevals, $rawstring);

echo "Son: ". $malestr . "<br>";
echo "Daughter: ". $femalestr;
?>
```

Pada script di atas, array `$placeholder` berisi string-string dari `$rawstring` yang akan diganti.

Dan tampilan script di atas adalah sbb:

```
Son: Welcome Birmingham parent!
Your son is a pleasure to have! We believe he is learning a lot.
The faculty simple adores he2 and you can often hear them say "Attah
boy!"
```

```
Daughter: Welcome Birmingham parent!
Your daughter is a pleasure to have! We believe she is learning a lot.
The faculty simple adores she2 and you can often hear them say "Attah
girl!"
```

Perhatikan string yang dicetak merah pada tampilan script tersebut. Seharusnya string `pronoun2` akan diganti dengan `him` dan `her`. Hal ini dikarenakan efek `str_replace` dari string `pronoun` yang merupakan substring dari `pronoun2`. Untuk menghindari efek kesalahan seperti itu, hindari peletakan substring di depan string lain dalam `str_replace`.

Berikut ini hasil modifikasinya:

```
<?
$rawstring = "Welcome Birmingham parent! <br>
    Your offspring is a pleasure to have!
    We believe pronoun is learning a lot.<br>
    The faculty simple adores pronoun2 and you can often hear
    them say \"Attah sex!\"<br>";

$placeholders = array('offspring', 'pronoun2', 'pronoun', 'sex');
$malevals = array('son', 'him', 'he', 'boy');
$femalevals = array('daughter', 'her', 'she', 'girl');
```

```
$malestr = str_replace($placeholders, $malevals, $rawstring);  
  
$femalestr = str_replace($placeholders, $femalevals, $rawstring);  
  
echo "Son: ". $malestr . "<br>";  
echo "Daughter: ". $femalestr;  
?>
```

dan tampilannya adalah

```
Son: Welcome Birmingham parent!  
Your son is a pleasure to have! We believe he is learning a lot.  
The faculty simple adores him and you can often hear them say "Attah  
boy!"
```

```
Daughter: Welcome Birmingham parent!  
Your daughter is a pleasure to have! We believe she is learning a lot.  
The faculty simple adores her and you can often hear them say "Attah  
girl!"
```

## strtoupper

Function ini digunakan untuk mengubah semua karakter huruf dari suatu string menjadi kapital.

Contoh:

```
<?  
$originalString = "String Capitalization 1234";  
  
$upperCase = strtoupper($originalString);  
echo "Old string - $originalString <br>";  
echo "New String - $upperCase";  
?>
```

Hasilnya adalah

```
Old string - String Capitalization 1234  
New String - STRING CAPITALIZATION 1234
```

## strtolower

Kebalikan dari strtoupper(), function ini mengubah semua karakter huruf dari string menjadi huruf kecil.

Contoh:

```
<?  
$originalString = "String Capitalization 1234";  
  
$lowerCase = strtolower($originalString);  
echo "Old string - $originalString <br>";
```

```
echo "New String - $lowerCase";  
?>
```

Hasilnya adalah

```
Old string - String Capitalization 1234  
New String - string capitalization 1234
```

## ucwords

Karakter huruf pertama dari suatu kata dalam string juga dapat diubah menjadi huruf kapital menggunakan function ini.

Contoh:

```
<?  
$titleString = "a title that could use some hELP";  
  
$ucTitleString = ucwords($titleString);  
echo "Old title - $titleString <br>";  
echo "New title - $ucTitleString";  
?>
```

Hasilnya adalah:

```
Old title - a title that could use some hELP  
New title - A Title That Could Use Some HELP
```

Bagaimana cara mengubah HELP menjadi Help (pada tampilan outputnya)? Perhatikan script modifikasi berikut ini

```
<?  
$titleString = "a title that could use some hELP";  
  
$lowercaseTitle = strtolower($titleString);  
$ucTitleString = ucwords($lowercaseTitle);  
echo "Old title - $titleString <br />";  
echo "New title - $ucTitleString";  
?>
```

yaitu dengan cara mengubahnya ("hELP") ke huruf kecil semua terlebih dahulu ("help"), kemudian huruf pertama dari "help" diberikan perintah `ucwords()`.

## explode

Sesuai namanya "explode", fungsi ini digunakan untuk meledakkan/memecah suatu string menjadi potongan-potongan string yang kecil. Selanjutnya potongan-potongan string ini akan disimpan dalam suatu array. Perhatikan contoh berikut ini

```
<?
$PhoneNumber = "800-555-5555";

$hasil = explode("-", $PhoneNumber);
echo " Phone Number = $PhoneNumber <br>";
echo "Pecahan 1 = $hasil[0]<br>";
echo "Pecahan 2 = $hasil[1]<br>";
echo "Pecahan 3 = $hasil[2]";
?>
```

Hasilnya adalah:

```
Phone Number = 800-555-5555
Pecahan 1 = 800
Pecahan 2 = 555
Pecahan 3 = 5555
```

Pada perintah `explode("-", $PhoneNumber);` di atas, karakter "-" dapat diibaratkan sebagai dinamitnya. Dinamit ini apabila diledakkan akan memecah string dari `$PhoneNumber`. Karakter yang akan digunakan sebagai dinamit dapat ditentukan sendiri oleh programmer, dapat berupa spasi, koma dsb.

Jumlah pecahan string dari hasil ledakan dapat dibatasi dengan menambahkan jumlah batas ledakan sebagai paramater ketiga dari function `explode()`.

Contoh:

```
<?
$PhoneNumber = "800-555-5555";

$hasil = explode("-", $PhoneNumber, 2);
echo " Phone Number = $PhoneNumber <br>";
echo "Pecahan 1 = $hasil[0]<br>";
echo "Pecahan 2 = $hasil[1]<br>";
echo "Pecahan 3 = $hasil[2]";
?>
```

Hasilnya adalah:

```
Phone Number = 800-555-5555
Pecahan 1 = 800
Pecahan 2 = 555-5555
Pecahan 3 =
```

Perintah `explode("-", $PhoneNumber, 2);` di atas membatasi 2 buah pecahan string dari hasil ledakan. Dengan demikian, tampak pada hasil bahwa pecahan ketiga tidak ada.

## implode

Kebalikan dari `explode()`, function `implode()` digunakan untuk menyatukan pecahan-pecahan string menjadi satu kesatuan string.

Contoh:

```
<?
$pecahan = array("Hello", "World,", "I", "am", "Here!");

$disatukandenganspasi = implode(" ", $pecahan);
$disatukandengandash = implode("-", $pecahan);

echo "$disatukandenganspasi <br>";
echo "$disatukandengandash ";
?>
```

Hasilnya adalah:

```
Hello World, I am Here!
Hello-World,-I-am-Here!
```

## BAB XII

# FUNCTION DATE

Pada bab ini akan dibahas mengenai cara menampilkan tanggal ke dalam halaman web. Tanggal yang akan ditampilkan menyesuaikan waktu server, bukan waktu yang ada di client. Hal ini disebabkan PHP merupakan *server side programming*.

Contoh:

```
<?
echo date("m/d/y");
?>
```

Function `date()` digunakan untuk menampilkan tanggal pada saat itu (sesuai waktu server). Misalkan pada saat itu adalah tanggal 12 Nopember 2005, maka tampilan dari script di atas adalah

11/12/05

Kita dapat mengubah format tanggal dalam bentuk 12-11-05 dengan perintah

```
<?
echo date("d-m-y");
?>
```

Lantas bagaimana kalau kita ingin menampilkan tanggal pada 2 hari mendatang? Berikut ini scriptnya.

```
<?
$duaharilagi = mktime(0, 0, 0, date("m"), date("d")+2, date("y"));
echo "Dua hari lagi adalah tanggal ". date("d/m/y", $duaharilagi);
?>
```

Function `mktime()` digunakan untuk membuat timestamp, dengan sintaks:

`mktime(hour, minute, second, month, day, year)`

Contoh:

```
<?php
$tomorrow = mktime(0, 0, 0, date("m"), date("d")+1, date("Y"));
$lastmonth = mktime(0, 0, 0, date("m")-1, date("d"), date("Y"));
$nextyear = mktime(0, 0, 0, date("m"), date("d"), date("Y")+1);

echo "Besok adalah tanggal ". date("d/m/y", $tomorrow) . "<br>";
echo "Sebulan lalu adalah tanggal ". date("d/m/y", $lastmonth) . "<br>";
echo "Setahun lagi adalah tanggal ". date("d/m/y", $nextyear) . "<br>";
?>
```

Beberapa ini beberapa jenis timestamp yang dapat digunakan untuk mengatur format tampilan tanggal dan waktu

Time:

a : am atau pm  
A : AM atau PM  
g : Jam tanpa 0 di depan. Bernilai antara 1- 12.  
G : Jam tanpa 0 di depan (format 24 jam). Bernilai antara 0- 23.  
h : Jam dengan 0 di depan. Bernilai antara 01- 12.  
H : Jam dengan 0 di depan (format 24 jam). Bernilai antara 00- 23.  
i : Menit dengan 0 di depan. Bernilai antara 00-59.  
s : Detik dengan 0 di depan. Bernilai antara 00-59.

Day:

d : Hari dalam bulan (tanggal) dengan 0 di depan. Bernilai antara 01-31.  
j : Hari dalam bulan (tanggal) tanpa 0 di depan. Bernilai antara 1-31.  
D : Hari dalam mingguan (disingkat). Bernilai antara Sun-Sat  
l : Hari dalam mingguan. Bernilai antara Sunday-Saturday  
w : Hari dalam mingguan tanpa 0 di depan. Bernilai antara 0-6.  
z : Hari dalam tahunan tanpa 0 di depan. Bernilai antara 0-365.

Month:

m : Nomor bulan dengan 0 di depan (01-12)  
n : Nomor bulan tanpa 0 di depan (1-12)  
M : Singkatan dari bulan. (Jan-Dec)  
F : Nama bulan lengkap. (January-December)  
t : Jumlah hari dalam sebulan. (28-31)

Year:

L : 1 jika melompati tahun dan 0 jika tidak.  
Y : Format tahun 4 digit  
y : Format tahun 2 digit. (00-99)

Contoh:

```
<?php

// Misalkan hari ini adalah: 10 Maret 2001, 5:16:18 pm

$today = date("F j, Y, g:ia");
echo "$today";
$today = date("m.d.y");
echo "$today";
$today = date("j, n, Y");
echo "$today";
$today = date("Ymd");
echo "$today";
$today = date("H:i:s");
echo "$today";
?>
```

Hasilnya adalah:

```
March 10, 2001, 5:16 pm
03.10.01
```

10, 3, 2001  
20010310  
17:16:17



# BAB XIII

## BEKERJA DENGAN SESSION

Session digunakan untuk mengirim data ke beberapa halaman web. Sebuah halaman web, secara normal tidak akan mengirim suatu data dari halaman yang satu ke halaman yang lain. Dengan kata lain, semua informasi tentang data tersebut akan hilang begitu halaman web direload.

Berikut ini akan diberikan contoh tentang penjelasan di atas, mengenai perlunya menggunakan session.

Contoh:

Diberikan source code sbb.

### Form.php

```
<form action=submit.php method=post>
Username <input type=text name=username><input type=submit name=submit
value=Submit>
</form>
```

### Submit.php

```
<?
$username = $_POST['username'];
echo "Nama user Anda adalah: $username<br><br>";
echo "<a href=hal1.php>Hal 1</a> <a href=hal2.php>Hal 2</a> <a
href=hal3.php>Hal 3</a>";
?>
```

### Hal1.php

```
<?
echo "Ini adalah halaman 1<br>";
echo "Nama user Anda adalah: $username<br><br>";
echo "<a href=hal1.php>Hal 1</a> <a href=hal2.php>Hal 2</a> <a
href=hal3.php>Hal 3</a>";
?>
```

### Hal2.php

```
<?
echo "Ini adalah halaman 2<br>";
echo "Nama user Anda adalah: $username<br><br>";
echo "<a href=hal1.php>Hal 1</a> <a href=hal2.php>Hal 2</a> <a
href=hal3.php>Hal 3</a>";
?>
```

### Hal3.php

```
<?
echo "Ini adalah halaman 3<br>";
echo "Nama user Anda adalah: $username<br><br>";
echo "<a href=hal1.php>Hal 1</a> <a href=hal2.php>Hal 2</a> <a
href=hal3.php>Hal 3</a>";
?>
```

Dengan file-file di atas, user diminta melakukan login terlebih dahulu (memasukkan nama user) melalui form.php. Selanjutnya akan muncul submit.php yang menampilkan nama user yang dimasukkan tadi dan beberapa link ke halaman lain. Begitu user mengakses 3 buah halaman web yang ada tersebut, diharapkan nama user yang telah dimasukkan dalam form sebelumnya tetap ditampilkan dalam setiap halaman web yang diakses. Akan tetapi apa yang terjadi? Ternyata untuk ketiga halaman web yang diakses tersebut tidak menampilkan nama user. Hal ini dikarenakan hilangnya data/informasi dari nama user yang dimasukkan sebelumnya.

Nah... di sinilah perlunya session. Dengan session, data dapat disimpan dan selanjutnya dapat diakses di beberapa halaman web.

Penggunaan session sering diterapkan pada aplikasi web yang bersifat multiuser, seperti online shopping, web based mail, e-banking, dll.

Data yang tersimpan dalam session bersifat temporary/ sementara. Biasanya akan terhapus secara otomatis begitu user menutup browser, atau melakukan logout.

## Memulai PHP – Session

Sebelum Anda menyimpan data dalam session, terlebih dahulu harus memulai session. Untuk memulai session, perintahnya adalah:

```
session_start()
```

## Menyimpan Data ke dalam Session

Untuk menyimpan data ke dalam session, digunakan perintah

```
$_SESSION['nama_session'] = data;
```

Contoh:

Berikut ini adalah code dalam file submit.php (contoh sebelumnya) yang telah dimodifikasi. Nama user akan disimpan dalam session.

```
<?
session_start();

$username = $_POST['username'];

$_SESSION['namauser'] = $username;
```

```
echo "Nama user Anda adalah: $username<br><br>";  
echo "<a href=hal1.php>Hal 1</a> <a href=hal2.php>Hal 2</a> <a  
href=hal3.php>Hal 3</a>";  
?>
```

## Memanggil Data yang Tersimpan dalam Session

Setelah data disimpan dalam session, selanjutnya dapat dipanggil kembali apabila diperlukan. Untuk memanggil data dalam session, caranya cukup menuliskan

```
$_SESSION['nama_session'].
```

Contoh :

Berikut ini adalah code dari file hal1.php, hal2.php, dan hal3.php sehingga dapat menampilkan nama user yang telah disimpan dalam session.

### Hal1.php

```
<?  
session_start();  
echo "Ini adalah halaman 1<br>";  
echo "Nama user Anda adalah: ".$_SESSION['namauser']. "<br><br>";  
echo "<a href=hal1.php>Hal 1</a> <a href=hal2.php>Hal 2</a> <a  
href=hal3.php>Hal 3</a>";  
?>
```

### Hal2.php

```
<?  
session_start();  
echo "Ini adalah halaman 1<br>";  
echo "Nama user Anda adalah: ".$_SESSION['namauser']. "<br><br>";  
echo "<a href=hal1.php>Hal 1</a> <a href=hal2.php>Hal 2</a> <a  
href=hal3.php>Hal 3</a>";  
?>
```

### Hal3.php

```
<?  
session_start();  
echo "Ini adalah halaman 1<br>";  
echo "Nama user Anda adalah: ".$_SESSION['namauser']. "<br><br>";  
echo "<a href=hal1.php>Hal 1</a> <a href=hal2.php>Hal 2</a> <a  
href=hal3.php>Hal 3</a>";  
?>
```

## Menghapus Data Session

Untuk menghapus data yang tersimpan dalam suatu session, digunakan perintah

```
unset($_SESSION['nama_session']);
```

Misalkan terdapat lebih dari satu session dan kita ingin menghapus semuanya, maka caranya dengan memberikan perintah

```
session_destroy();
```

Biasanya menghapus data session yang terkait dengan username diterapkan pada proses logout. Dengan proses logout, maka username yang telah tersimpan dalam session akan terhapus.

Contoh:

Akan dibuat script untuk proses logout dari kasus di atas.

```
<?
session_start();
unset($_SESSION['namauser']);
?>
```

## Penerapan Session untuk Security

Pada contoh kasus di atas, misalkan kita buat aturan bahwa untuk dapat mengakses halaman 1, 2, dan 3, user harus terlebih dahulu melakukan login. Dari script yang kita buat di atas, seorang user bisa saja langsung melakukan by pass ke tiga halaman tersebut tanpa login terlebih dahulu. Tentu saja hal ini bisa berbahaya untuk aplikasi multiuser yang harus menjamin keamanan data dari para user-nya.

Untuk mencegah proses by pass tersebut, dapat kita akali dengan menggunakan session. Logikanya adalah, apabila user melakukan login, maka nama user yang dia masukkan akan disimpan dalam session. Sedangkan apabila ada seorang user yang mencoba mem by pass, maka dengan kata lain username tidak akan disimpan dalam session (session untuk user masih kosong), dengan catatan bahwa user lain yang sebelumnya login harus sudah me-logout. Dari hal ini, kita dapat melakukan cek apakah seorang user sudah melakukan login atau belum dengan melihat session, masih kosong atau tidak.

Untuk melihat sebuah session masih kosong atau tidak dengan menggunakan perintah

```
isset($_SESSION['nama_session'])
```

Perintah di atas akan menghasilkan nilai TRUE apabila session sudah tidak kosong, dan akan menghasilkan nilai FALSE apabila session masih kosong.

Contoh:

Kita akan membuat script untuk mencegah by pass dalam kasus sebelumnya. Script ini selanjutnya akan disisipkan pada setiap halaman yang diinginkan, dalam hal ini adalah halaman 1, 2, dan 3.

### Cek.php

```
<?
session_start();

if (!isset($_SESSION['namauser']))
{
```

```
        echo "Anda belum login";
        exit;
    }
    ?>
```

Dan berikut adalah script hal1.php, hal2.php, dan hal3.php yang sudah dimodifikasi

### Hal1.php

```
<?
session_start();
include "cek.php";
echo "Ini adalah halaman 3<br>";
echo "Nama user Anda adalah: " . $_SESSION['namauser'] . "<br><br>";
echo "<a href=hal1.php>Hal 1</a> <a href=hal2.php>Hal 2</a> <a
href=hal3.php>Hal 3</a>";
?>
```

### Hal2.php

```
<?
session_start();
include "cek.php";
echo "Ini adalah halaman 3<br>";
echo "Nama user Anda adalah: " . $_SESSION['namauser'] . "<br><br>";
echo "<a href=hal1.php>Hal 1</a> <a href=hal2.php>Hal 2</a> <a
href=hal3.php>Hal 3</a>";
?>
```

### Hal3.php

```
<?
session_start();
include "cek.php";
echo "Ini adalah halaman 3<br>";
echo "Nama user Anda adalah: " . $_SESSION['namauser'] . "<br><br>";
echo "<a href=hal1.php>Hal 1</a> <a href=hal2.php>Hal 2</a> <a
href=hal3.php>Hal 3</a>";
?>
```

# BAB XIV

## BEKERJA DENGAN COOKIES

Cookies telah lama digunakan dalam internet yang digunakan oleh administrator web untuk menyimpan informasi user atau pengunjung di komputer user tersebut.

### Membuat Cookies

Untuk membuat cookies, caranya dengan menggunakan perintah `setcookie(name, value, expiration)`. Perintah ini memiliki 3 buah argumen, yaitu

1. **name**, merupakan nama dari cookie. Nama cookie ini dapat dipanggil sewaktu-waktu untuk mendapatkan informasi. Jadi jangan sampai nama cookie ini lupa.
2. **value**, merupakan informasi atau data yang disimpan dalam cookie. Biasanya value ini berupa username atau tanggal pengaksesan suatu halaman web.
3. **expiration**, merupakan batas waktu penyimpanan cookie (dalam detik timestamp). Apabila lama penyimpanan sebuah cookie melebihi batas waktu ini, maka secara otomatis cookie tersebut akan terhapus.

Contoh:

Pada contoh ini akan dibuat perintah PHP untuk membuat cookie yang digunakan untuk menyimpan data waktu kunjungan terakhir seorang user yang mengakses suatu halaman web. Cookie ini diberi batas waktu sampai 2 bulan (60 hari) penyimpanan.

```
<?php
$duabulanlagi = time() + 60 * 24 * 3600;
setcookie(KunjunganTerakhir, date("G:i - m/d/y"), $duabulanlagi);
?>
```

### Mengambil Informasi dari Cookie

Apabila cookie belum terhapus, maka kita dapat mengambil informasi dari cookie. Untuk mengecek apakah suatu cookie sudah terhapus atau belum menggunakan perintah

```
isset($_COOKIE['nama_cookie'])
```

Apabila cookie masih ada, maka perintah di atas menghasilkan TRUE. Sedangkan apabila sudah terhapus, akan menghasilkan nilai FALSE.

Sedangkan untuk mengambil informasi dari cookie, menggunakan perintah

```
$_COOKIE['nama_cookie']
```

Contoh:

Berikut ini contoh perintah PHP untuk menampilkan tanggal kunjungan terakhir user yang mengunjungi halaman web.

```
<?php
if(isset($_COOKIE['KunjunganTerakhir']))
{
    $visit = $_COOKIE['KunjunganTerakhir'];
    echo "Kunjungan Anda terakhir pada - ". $visit;
}
else
    echo "Anda sudah 2 bulan lebih tidak mengunjungi web ini";

?>
```

# BAB XV

## PHP- MySQL

MySQL merupakan salah satu DBMS open source yang paling populer pada saat ini. Meskipun dahulu MySQL pernah dikritisi karena tidak memiliki beberapa fitur yang ada dalam DBMS pada umumnya, namun saat ini MySQL sudah banyak dikembangkan.

### Koneksi PHP ke MySQL

Sebelum kita melakukan koneksi ke MySQL ada beberapa parameter yang harus kita ketahui terlebih dahulu. Untuk melakukan koneksi, dibutuhkan:

- Server name, merupakan nama server atau no. IP server dimana MySQL tersebut diinstall
- Username, merupakan nama user yang diberikan wewenang untuk mengakses database dalam MySQL
- Password, merupakan password yang dimiliki username dalam rangka autentifikasi.
- Database name, merupakan nama database dalam MySQL yang ingin kita akses.

Untuk memperoleh informasi parameter di atas, dapat menghubungi server administrator.

Sedangkan perintah PHP untuk melakukan koneksi ke MySQL adalah

```
<?php
mysql_connect("nama server", "username", "password") or
die(mysql_error());
echo "Koneksi ke MySQL Sukses<br>";
?>
```

Perintah di atas akan menampilkan Koneksi ke MySQL sukses apabila koneksi telah berhasil, sedangkan apabila gagal akan menampilkan pesan kesalahan.

### Memilih Database

Setelah koneksi berhasil, selanjutnya kita dapat memilih database yang kita inginkan. Adapun perintahnya menggunakan `mysql_select_db()`.

Contoh:

```
<?php
mysql_connect("localhost", "admin", "ladmin") or die(mysql_error());
echo "Connected to MySQL<br />";
mysql_select_db("test") or die(mysql_error());
echo "Connected to Database";
?>
```



Contoh di atas menggambarkan bagaimana cara melakukan koneksi ke MySQL dengan nama servernya localhost, username : admin, dan password: ladmin. Selanjutnya memilih database test.

## Memberikan Query ke MySQL

Setelah kita memilih database dimana kita akan bekerja, selanjutnya kita dapat memberikan perintah query seperti SELECT, DELETE, CREATE, UPDATE. Berikut ini contoh script PHP untuk membuat tabel example dalam database test.

```
<?php
mysql_connect("localhost", "admin", "ladmin") or die(mysql_error());
mysql_select_db("test") or die(mysql_error());

mysql_query("CREATE TABLE example(
            id INT NOT NULL AUTO_INCREMENT,
            nama VARCHAR(30),
            umur INT)
            PRIMARY KEY(id)")
or die(mysql_error());

echo "Tabel sudah dibuat";

?>
```

Pada contoh di atas, dapat dilihat bahwa perintah PHP untuk menuliskan query ke MySQL adalah

```
mysql_query("query");
```

Sedangkan berikut ini contoh script untuk menyisipkan 2 buah record/data ke tabel example.

```
<?php
mysql_connect("localhost", "admin", "ladmin") or die(mysql_error());
mysql_select_db("test") or die(mysql_error());

mysql_query("INSERT INTO example(nama, umur)
            VALUES('budiman', 20)");

mysql_query("INSERT INTO example(nama, umur)
            VALUES('surti', 30)");

echo "Data sudah dimasukkan";

?>
```

## Mengambil Data dari MySQL

Mengambil data di sini terkait dengan penggunaan query SELECT. Berikut ini contoh untuk menampilkan record pertama dari tabel example.

```
<?php
mysql_connect("localhost", "admin", "ladmin") or die(mysql_error());
mysql_select_db("test") or die(mysql_error());

$result = mysql_query("SELECT * FROM example")
or die(mysql_error());

// menyimpan record ke dalam variabel $data
$record = mysql_fetch_array( $result );

// menampilkan data dari $record untuk setiap field
echo "Namanya: ".$record['nama']. "<br>";
echo "Umurnya: ".$record['umur'];

?>
```

Output script di atas adalah:

```
Namanya: budiman
Umurnya: 20
```

Perintah di atas hanya akan menampilkan record pertama dari tabel example. Lantas, bagaimana caranya untuk menampilkan record yang lebih dari satu?

Untuk menampilkan record yang lebih dari satu, kita gunakan looping. Perhatikan contoh berikut ini.

```
<?php
mysql_connect("localhost", "admin", "ladmin") or die(mysql_error());
mysql_select_db("test") or die(mysql_error());

$hasil = mysql_query("SELECT * FROM example")
or die(mysql_error());

while ($record = mysql_fetch_array($hasil))
{
    echo "Namanya: ".$record['nama']. "<br>";
    echo "Umurnya: ".$record['umur']. "<br><br>";
}

?>
```

Looping while di atas akan terus berjalan selama record masih ada untuk dibaca. Hasil dari script di atas adalah

```
Namanya: budiman
Umurnya: 20
```

```
Namanya: surti
Umurnya: 30
```

### Catatan:

Untuk lebih memudahkan Anda dalam administrasi dan mengatur database dalam MySQL, sangat dianjurkan untuk menginstall phpMyAdmin yang dapat diunduh melalui situs resminya di <http://phpmyadmin.sourceforge.net/>

# BAB XVI

## SQL INJECTION

Pada bab ini akan dibahas mengenai apa itu SQL injection, cara melakukannya dan cara pencegahannya. Meskipun dalam bab ini dijelaskan cara melakukan teknik ini, akan tetapi penulis bukan bertujuan untuk mendidik orang untuk menjadi penjahat dalam dunia virtual, melainkan memberikan pemahaman supaya orang yang membaca menjadi *eling lan waspodo*.

### Apa itu SQL Injection

SQL injection merupakan perbuatan orang yang memberikan perintah SQL untuk dijalankan di mesin server SQL tanpa sepengetahuan Anda sebagai administrator. Proses injection biasanya dilakukan orang ketika memasukkan input melalui form dengan perintah atau kode tertentu. Berikut ini adalah contohnya:

```
<?
$nama = "agus";
$query = "SELECT * FROM customers WHERE username = '$nama'";
echo "Query Normal: " . $query . "<br>";

// input user yang menggunakan SQL injection
$name = "' OR 1 OR '1' = '1'";

$query = "SELECT * FROM customers WHERE username = '$name'";

echo "Query Injection: " . $query;
?>
```

Tampilan script di atas adalah

```
Query Normal: SELECT * FROM customers WHERE username = 'agus'
Query Injection: SELECT * FROM customers WHERE username = '' OR 1' OR
'1' = '1'
```

Apabila query normal dijalankan di server SQL tentu saja tidak ada masalah karena akan mencari record dari tabel `customers` yang terkait dengan username `agus`. Query tersebut akan mendapatkan data yang diinginkan apabila memang terdapat user `agus` dan tidak akan mendapatkan data yang diinginkan apabila tidak terdapat user tersebut.

Sedangkan untuk query injection pasti akan selalu mendapatkan paling sedikit satu record/ data dari tabel `customers`. Mengapa hal ini bisa terjadi? Hal ini dikarenakan dalam query injection terdapat perintah `OR 1` yang selalu bernilai `TRUE`.

Berikut ini adalah contoh mekanisme SQL injection yang dapat terjadi pada proses login.

Misalkan terdapat tabel `user` yang digunakan untuk menyimpan data user, yang di dalamnya terdapat 2 field yaitu `username` dan `password`. Misalkan juga terdapat 2 user yaitu `JOKO` (`password : JOKO`) dan `AMIR` (`password : AMIR`).

Diberikan form login sbb.

### Login.php

```
<form method=post action=submit.php>
Nama user <input type=text name=username><br>
Password <input type=text name=password>
<input type=submit name=submit value=Submit>
</form>
```

Dan file **submit.php** sbb.

```
<?

mysql_connect("localhost","root","root");
mysql_select_db("test");

$username = $_POST['username'];
$password = $_POST['password'];

$query = "SELECT * FROM user WHERE username = '$username'
        AND password = '$password'";

echo "$query<br>";

$hasil = mysql_query($query);
$jmldata = mysql_num_rows($hasil);

if ($jmldata != 0) echo "<h1>Login SUKSES</h1>";
else echo "<h1>Login GAGAL</h1>";

?>
```

Ide dari proses login di atas adalah mencari namauser dan password dalam tabel user berdasarkan username dan password yang dimasukkan melalui form. Perintah `mysql_num_rows()` akan menghasilkan jumlah record hasil pencarian. Apabila ditemukan namauser dan password yang sesuai maka `mysql_num_rows()` akan menghasilkan jumlah baris record yang tidak sama dengan 0. Sedangkan apabila tidak ditemukan, jumlah baris record adalah 0. Selanjutnya jumlah baris record yang muncul tersebut dicek. Login akan berhasil apabila ada record yang ditemukan atau jumlah baris recordnya tidak sama dengan 0, dan akan gagal jika jumlah baris recordnya 0.

Sekarang, perhatikan apa yang terjadi apabila dalam form dimasukkan input sebarang nama user dan password berbentuk ' OR 1 OR '1' = '1. Pastilah login akan sukses karena perintah SQL yang dijalankan adalah

```
SELECT * FROM user WHERE username = 'xxx'
        AND password = '' OR 1 OR '1' = '1'
```

yang akan menghasilkan TRUE

## Cara Pencegahan SQL Injection

Beruntunglah ... sekarang PHP sudah dilengkapi dengan security (PHP rilis terbaru) dengan mengubah karakter single quote (') menjadi (\'). Dengan kata lain karakter single quote yang diinputkan melalui form secara otomatis akan diubah menjadi backslash-single quote (\').

Sehingga untuk kasus login di atas perintah SQL yang dijalankan adalah

```
SELECT * FROM user WHERE username = 'xxx'
      AND password = '\ ' OR 1 OR \'1\' = \'1'
```

sehingga akan menghasilkan FALSE karena password yang dicari adalah `\ ' OR 1 OR \'1\' = \'1'`. Dengan demikian login akan gagal.

**Catatan:**

Dalam MySQL, tanda single quote digunakan sebagai pengapit data/value yang berupa string. Sedangkan backslash-single quote merupakan tanda yang digunakan untuk menyatakan karakter single quote.

Apabila PHP Anda tidak mensupport metode tersebut (biasanya PHP rilis lama), cara lain adalah menggunakan perintah `mysql_real_escape_string()`. Peran perintah ini sama dengan metode di atas yaitu mengubah single quote menjadi backslash-single quote.

Script berikut ini adalah modifikasi dari `submit.php` pada kasus login di atas yang ditambahkan perintah `mysql_real_escape_string()`.

```
<?
```

```
mysql_connect("localhost","root","root");
mysql_select_db("test");

$username = mysql_real_escape_string($_POST['username']);
$password = mysql_real_escape_string($_POST['password']);

$query = "SELECT * FROM user WHERE username = '$username'
        AND password = '$password'";

echo "$query<br>";

$hasil = mysql_query($query);
$jmldata = mysql_num_rows($hasil);

if ($jmldata != 0) echo "<h1>Login SUKSES</h1>";
else echo "<h1>Login GAGAL</h1>";

?>
```