

PHP? Siapa Takut!

Ivan Irawan

ivanorma@softhome.net

Lisensi Dokumen:

Copyright © 2003-2006 IlmuKomputer.Com

Seluruh dokumen di IlmuKomputer.Com dapat digunakan, dimodifikasi dan disebarkan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari IlmuKomputer.Com.

Bagian 1: Captain's Log Stardate 41153.7

Kisah Sang PHP

Web pada kisah awalnya sangat menjemukan bagi orang-orang yang dinamis. Bagaimana tidak, pemakainya hanya dicekoki oleh isi (*content*) halaman web yang meskipun bersifat saling terhubung dengan halaman web yang lain (*hyperlink*) tetap saja tidak memberikan saluran bagi pengguna yang ingin mengemukakan pendapatnya. Tidak ada demokrasi, karena pengguna hanya bersifat pasif dan tidak bisa berinteraksi secara aktif dalam web.

Ketika akhirnya ditemukan *tag* <FORM> barulah kejemuan dan kebuntuan yang ada menjadi sirna. Pengguna menjadi bisa secara aktif berinteraksi dengan halaman web, dan mulailah era aplikasi berbasis web yang dinamis. Secara tradisi, bahasa *script* Perl menjadi bahasa utama yang digunakan oleh *programmer* web untuk menangani pemrosesan form dalam berinteraksi dengan pengguna web. Tidak diragukan lagi kedigjayaan dari Perl dalam menangani urusan ini, hal ini juga didukung dengan begitu dominannya bahasa ini digunakan di situs-situs web yang ada.

Perl bisa menjadi alat bantu yang sangat hebat di tangan ahlinya, namun akan berubah menjadi mimpi paling buruk bagi seorang *programmer* web pemula yang dikejar waktu dan bosnya untuk segera menulis halaman webnya. Tidak mudah memang, mempelajari bahasa Perl, dan seringkali dibutuhkan langkah panjang dan rumit untuk sebuah maksud yang sederhana saja. Pendek kata, dibutuhkan suatu bahasa yang lebih praktis dan mudah dipelajari serta adidaya untuk memudahkan dalam membangun sebuah aplikasi yang berbasis web.

Di rimba belantara web, tersebutlah dua bahasa yang paling kondang yang mampu menggantikan tugas-tugas Perl namun dengan tingkat kesulitan belajar yang rendah, ASP (*Active Server Page*) dan PHP (*PHP: Hypertext Preprocessor*). ASP yang dijagokan oleh Pak Bill Gates tentu saja berjalan di lingkungan sistem operasi Windows dan sampai saat ini belum terlihat akan di-*porting* ke platform yang lain. Padahal dunia web saat ini masih didominasi oleh platform UNIX dan *variant*-nya termasuk sistem operasi *like UNIX* seperti Linux. Selain itu, untuk dapat menggunakan ASP yang resmi, kita juga harus merelakan sebagian uang kita untuk menambah isi kantong Pak Bill Gates.

PHP sebagai alternatif lain memberikan solusi sangat murah (karena gratis digunakan) dan dapat berjalan di berbagai jenis platform. Awalnya memang PHP berjalan di sistem UNIX dan *variant*-nya, namun kini dapat berjalan dengan mulus di lingkungan sistem operasi Windows. Suatu nilai tambah yang luar biasa karena proses *development* program berbasis web dapat dilakukan lintas sistem operasi. Pak Fulan, misalnya, bisa mencuri waktu memrogram aplikasi untuk usaha pribadinya di kantor yang menggunakan sistem operasi Windows dan meneruskannya di rumahnya dengan komputer yang menggunakan sistem operasi Linux.

Dengan luasnya cakupan sistem operasi yang mampu menjalankan PHP dan ditambah begitu lengkapnya fungsi-fungsi program (tersedia lebih dari 400 fungsi di PHP yang sangat berguna) tidak heran jika PHP ini semakin menjadi *trend* di kalangan *programmer* web. Konon, saat ini lebih dari satu juta situs web menggunakan PHP sebagai *script* pemrogramannya.

Pak Rasmus Lerdorf adalah bapak penemu awal bahasa PHP ini, yang bermula dari keinginan sederhana Pak Lerdorf untuk mempunyai alat bantu (*tools*) dalam memonitor pengunjung yang melihat situs web pribadinya. Inilah sebabnya pada awal pengembangannya, PHP merupakan singkatan dari *Personal Home Page tools*, sebelum akhirnya dipaksakan menjadi singkatan rekursif dari *PHP: Hypertext Preprocessor*. Pertengahan tahun 1995 dirilis PHP/FI (FI adalah singkatan dari *Form Interpreter*) yang memiliki kemampuan dasar membangun aplikasi web, memproses form, dan mendukung database mSQL.

Antusias komunitas internet terhadap bahasa PHP ini begitu besar, sehingga Pak Rasmus Lerdorf akhirnya menyerahkan pengembangan PHP ini kepada sebuah team pemrograman dalam kerangka gerakan *open source*. Team ini membangun kembali PHP dari awal dengan menulis ulang program *parser* PHP Hasilnya adalah PHP 3.0 yang memiliki dukungan lebih luas lagi terhadap database yang ada termasuk MySQL dan Oracle. PHP 4.0 sebagai versi lanjutan dari PHP 3.0 dirilis setelah itu dengan menggunakan mesin *scripting* Zend (akronim dari pengembangnya, Zeev Suraski dan Andi Gutmans) untuk memberikan kinerja yang lebih cepat dan lebih baik Versi terakhir ini mampu mendukung server web selain Apache dan secara *built-in* telah mampu

menangani manajemen *session*.

Nah, dongeng ini kita cukupkan di sini dulu. Singkat kata, PHP kita pilih sebagai bahasa untuk pengembangan web yang akan kita pelajari di bagian selanjutnya. Sebelum memulainya, ada baiknya Anda mengetahui kebutuhan-kebutuhan dasar yang akan membantu Anda memahami tulisan ini. Anda diasumsikan telah memiliki sebuah sistem yang telah terinstalasi dan terkonfigurasi dengan baik Apache Web Server, PHP 4, dan database MySQL. Ketiganya adalah program *open source* yang tersedia secara gratis di Internet dan dapat berjalan di berbagai platform (Windows maupun UNIX/Linux). Anda juga harus merupakan seorang pemula di dunia PHP, karena kalau tidak Anda akan mengalami keadaan mirip anak SMA yang masuk ke kelas 1 SD. Tulisan ini tidak membahas pengenalan format HTML, sehingga diharapkan Anda telah memiliki pengetahuan dasar mengenai hal ini, karena bagaimana pun Anda akan menggunakan PHP untuk membangun aplikasi web yang pasti tidak lepas dari urusan tag HTML. Satu hal lagi, Anda juga perlu memiliki rasa humor yang cukup tinggi, dan menyukai Star Trek.

Apakah Anda sudah siap dan memenuhi syarat? Baik, mari kita teruskan.

Saya Piccard, Jean Luc Piccard

Jika sistem Apache Web Server, PHP4, dan MySQL kita telah siap dan terkonfigurasi dengan benar, sekarang adalah saatnya yang paling tepat untuk memulai perjalanan ini. Program "Hello World!" yang legendaris untuk memulai belajar sebuah bahasa pemrograman, sengaja tidak dipakai karena kita tidak ingin meneruskan tradisi kuno ini. Ada hal yang lebih berguna yang dapat kita pakai sebagai contoh. Anda bisa mulai memilih *text editor* favorit Anda (yang jelas **vi** bukan sebuah program nyaman bagi pemula). Jika Anda bekerja di sistem operasi Windows, Anda bahkan dapat memilih Lingkungan Pengembangan Terpadu/IDE (*Integrated Development Environment*) khusus untuk PHP seperti :

- PHP Coder buatan Jerman (<http://www.phpide.de>), yang saat ini kodenya telah diakuisisi oleh Maguma (<http://www.maguma.com>)
- PHPEd yang buatan Turki (<http://www.soysal.com/PHPEd>), terakhir kodenya dibeli oleh NuSphere (<http://www.nusphere.com>)
- PHPEdit, program *open source* yang dapat didownload dari <http://www.phpedit.net>

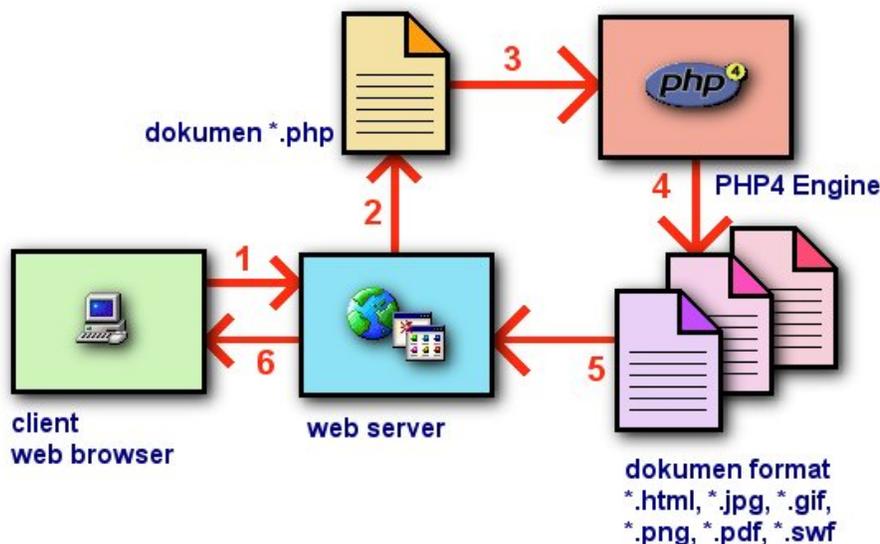
Silakan coba kode di bawah ini dan simpan sebagai file dengan ekstension `.php`, misalkan coba `.php`.

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

Untuk menjalankannya, kita bisa mulai membuka *browser* web kesukaan kita, kemudian arahkan alamat pada file coba.php yang telah kita buat, misalkan alamatnya adalah `http://localhost/coba.php`, maka Anda akan mendapatkan tampilan *browser* Anda berisikan parameter-parameter yang diset untuk PHP yang kita miliki. Anda dapat mengubah parameter ini dengan memodifikasi file `php.ini`. Jika Anda memang benar-benar masih pemula, jangan kaget melihat begitu banyaknya parameter yang harus diset untuk PHP Anda, karena akan saya beri tahu sebuah rahasia kecil bahwa nilai parameter yang *default* sebenarnya sudah sangat lebih dari cukup untuk memulai belajar pemrograman PHP 4.

Konsep pemrograman dengan PHP ini sedikit berbeda dengan pemrograman dengan menggunakan *script* CGI yang memaksa kita untuk selalu menulis kode yang menghasilkan keluaran dalam format HTML. Pada PHP, kita diberikan kebebasan untuk menyisipkan kode PHP di mana pun pada halaman HTML biasa dan menjalankan kode PHP tersebut setiap ada permintaan terhadap halaman tersebut.

Interpreter PHP dalam mengeksekusi kode PHP pada sisi server (disebut *server-side*) dan berbeda dengan mesin maya Java yang mengeksekusi program pada sisi client (*client-side*). Proses eksekusi kode PHP yang disisipkan pada halaman HTML secara diagram dapat digambar sebagai berikut.



Berikut ini adalah cara menyisipkan kode PHP pada halaman HTML biasa.

```
<script language="php">  
    . . . . kode PHP . . . .  
</script>
```

Cara yang lebih singkat adalah:

```
<?php  
    . . . . kode PHP . . . .  
?>
```

Atau bisa juga

```
<?  
    . . . . kode PHP . . . .  
?>
```

Bahkan jika Anda memiliki waktu yang cukup mengubah parameter pada php.ini, Anda bisa membuat kode penyisipan PHP menjadi mirip seperti pada ASP yaitu dengan:

```
<%  
    . . . . kode PHP . . . .  
%>
```

Sampai di sini ada keraguan? Jika tidak, mari kita lanjutkan dengan contoh nyata cara mengkombinasikan kode PHP dengan file HTML biasa. Cobalah Anda ketik kode di bawah ini, lalu simpan dengan nama misalnya `coba1.php`. Panggil melalui *browser* dan amati hasilnya.

```
<html>
<head>
<title>Test Penyisipan PHP Pada HTML</title>
</head>
<body>
Kapal Asing, Silakan identifikasikan diri Anda! <br>
<?php
// Berikut ini adalah kode PHP yang disisipkan
echo "<b>Ini adalah kapal Federasi Planet USS
Enterprise.<br>";
echo "Saya Piccard, Jean Luc Piccard, kapten kapal.</b>";
?>
</body>
</html>
```

Setelah Anda panggil file ini lewat *browser*, Anda dapat mencoba melihat kode asal dokumen HTML yang kurang lebih akan nampak seperti ini.

```
<html>
<head>
<title>Test Penyisipan PHP Pada HTML</title>
</head>
<body>
Kapal Asing, Silakan identifikasikan diri Anda! <br>
<b>Ini adalah kapal Federasi Planet USS Enterprise.<br>
Saya Piccard, Jean Luc Piccard, kapten kapal.</b>
</body>
</html>
```

Terlihat bahwa dokumen yang tampil di *browser* pengguna adalah murni HTML tanpa kode PHP satu pun. Pengguna tidak dapat melihat kode PHP yang ditulis oleh *programmer* karena kode tersebut telah diproses menjadi format HTML oleh interpreter PHP pada server asal kode PHP.

Pada setiap akhir perintah PHP selalu diakhiri dengan tanda titik-koma (";"), seperti juga Perl dan C. Bagi seorang pemula, keharusan ini seringkali dilupakan dan menjadi sebuah kesalahan umum terjadi. Programmer PHP dapat menyisipkan komentar yang tidak akan dieksekusi oleh mesin PHP dengan dua cara seperti pada contoh dibawah ini.

```
<?php

// Ini adalah komentar dalam satu baris

/* Kalau yang ini, komentar
dalam banyak baris, yang baru
akan selesai setelah diakhiri
dengan */

?>
```

Pencarian Jati Diri

Variabel/Peubah adalah bagaikan garam dan sayur dalam bahasa pemrograman, dan kabar baiknya adalah PHP pun memilikinya. Variabel dapat dibayangkan sebagai sebuah tempat penyimpanan data bagi nilai numeris maupun non numeris, agar dapat digunakan pada bagian lain dari *script* program PHP.

PHP mendukung berbagai jenis variabel yaitu:

- integer(bilangan bulat),
- bilangan *floating point* (presisi tunggal, ganda)
- *boolean*
- *null* (untuk variabel yang belum diset).
- *string*
- *array*
- *object*
- *resource*
- *unknown*.

Jika Anda terbiasa menggunakan C atau Pascal, maka Anda harus bersiap-siap kehilangan sebuah kewajiban, karena pada PHP Anda tidak perlu mendefinisikan terlebih dahulu jenis variabel sebelum menggunakannya. PHP memiliki kemampuan untuk membedakan jenis variabel secara otomatis berdasarkan konteks yang sedang berlaku bagi variabel tersebut.

Setiap variabel dalam PHP selalu dimulai dengan tanda dolar ("\$") dan harus dimulai dengan huruf dan dapat diikuti oleh huruf dan angka. Dengan demikian, \$warpspeed, \$impuls_speed, \$LCAR dan \$Dilithium1 adalah contoh penamaan variabel PHP yang valid.

Setiap variabel dalam PHP peka terhadap perbedaan huruf kapital dan non kapital, sehingga \$subspace, \$SubSpace, dan \$SUBSPACE adalah tiga buah variabel yang berbeda.

Mari kita coba latih sedikit penggunaan variabel PHP dengan contoh di bawah ini yang merupakan modifikasi dari contoh `coba1.php`.

```
<html>
<head>
<title>Test Penyisipan PHP Pada HTML</title>
</head>
<body>
```

```
Kapal Asing, Silakan identifikasikan diri Anda! <br>

<?php

// Berikut ini adalah inisiasi beberapa variabel
$namad = "Jean";
$namat = "Luc";
$namab = "Piccard";

?>

<b>Ini adalah kapal Federasi Planet USS Enterprise.<br>

<?php

echo "Saya $namab, $namad $namat $namab, kapten kapal.</b>";

?>

</body>
</html>
```

Variabel `$namad`, `$namat`, `$namab` adalah variabel yang dari awal diset sebagai variabel *string* dan kemudian isinya digunakan pada pemanggilan fungsi PHP `echo()`. Fungsi `echo()` merupakan fungsi yang sangat populer di PHP dan umumnya digunakan untuk membentuk keluaran tampilan. Saudara kembar dari fungsi `echo()` ini adalah fungsi `print()`. Kode berikut ini dapat digunakan sebagai contoh penggunaan fungsi `print()` untuk menggantikan fungsi `echo()`. Anda bisa menyimpan kode ini dengan nama `coba2.php`.

```
<html>
<head>
<title>Test Penyisipan PHP Pada HTML</title>
</head>

<body>

Kapal Asing, Silakan identifikasikan diri Anda! <br>

<?php

// Berikut ini adalah inisiasi beberapa variabel

$namad = "Jean";
$namat = "Luc";
```

```
$namab = "Piccard";  
  
?>  
  
<b>Ini adalah kapal Federasi Planet USS Enterprise.<br>  
  
<?php  
  
print("Saya $namab, $namad $namat $namab, kapten  
kapal.</b>");  
  
?>  
  
</body>  
</html>
```

Berikut ini adalah sebuah contoh mengenai betapa *luwesnya* penggunaan variable dalam PHP. Simpan kode ini dalam nama `coba3.php`.

```
<?php  
// Contoh variabel $a  
$a = "Testing";  
  
// Kini $a adalah variable jenis String  
echo "Nilai a adalah $a (string)<br>";  
  
$a = 55;  
  
// Kini $a adalah variable jenis Integer  
echo "Nilai a berubah menjadi $a (Integer)<br>";  
  
$a = 7.5;  
  
// Kini $a adalah variable jenis floating point  
echo "Nilai a sekarang menjadi $a (floating point)<br>";  
  
?>
```

Daftar Peralatan Geordi LaForge Yang Akan Dibeli

Kita telah mengenal dasar-dasar penggunaan variabel dalam PHP. Sekarang kita teruskan perjalanan kita dengan mempelajari penggunaan beberapa operator matematika yang paling berguna untuk menyusun sebuah daftar, katakanlah Geordi LaForge akan

menyusun daftar belanja peralatan di markas Federasi Planet untuk ekspedisi di Deep Space 9.

Geordi berencana membeli peralatan:

- Senjata Phaser 2 buah
- Tricorder 5 buah
- Visor Cadangan 1 buah
- Analyzer Photonik 3 buah

Senjata phaser berharga 7.500 dolar, Tricorder 12.500 dolar, Visor 16.000 dolar dan Analyzer Photonik berharga 2.300 dolar. Sebagai langganan, Geordi mendapatkan diskon 5% dari seluruh jenis peralatan yang dibeli. Bagaimanakah kita menyusun halaman HTML untuk menampilkan tabel daftar peralatan Geordi beserta harga dan total harganya? Anda bisa mencoba mempelajari kode di bawah ini yang dapat Anda simpan dengan nama `coba4.php`.

```
<?php

// inisiasi variable yang digunakan

// nama peralatan
$alat_geordi1 = "Phaser";
$alat_geordi2 = "Tricorder";
$alat_geordi3 = "Visor";
$alat_geordi4 = "Analyzer Photonik";

// harga per unit peralatan
$harga_alat_geordi1 = 7500;
$harga_alat_geordi2 = 12500;
$harga_alat_geordi3 = 16000;
$harga_alat_geordi4 = 2300;

// jumlah peralatan yang ada
$jumlah_alat_geordi1 = 2;
$jumlah_alat_geordi2 = 5;
$jumlah_alat_geordi3 = 1;
$jumlah_alat_geordi4 = 3;

// total harga per jenis peralatan
$total_alat_geordi1 = $jumlah_alat_geordi1 *
$harga_alat_geordi1;
$total_alat_geordi2 = $jumlah_alat_geordi2 *
$harga_alat_geordi2;
```

```
$total_alat_geordi3      =      $jumlah_alat_geordi3      *  
$harga_alat_geordi3;  
$total_alat_geordi4      =      $jumlah_alat_geordi4      *  
$harga_alat_geordi4;  
  
// hitung grand total nilai peralatan Geordi  
$total_harga = $total_alat_geordi1 + $total_alat_geordi2  
+ $total_alat_geordi3 + $total_alat_geordi4;  
  
// besar diskon untuk Geordi  
$diskon = 5;  
  
// jumlah total diskon yang diberikan kepada Geordi  
$nilai_diskon = ($diskon * $total_harga)/100;  
  
// jumlah yang harus dibayar Geordi  
$total_harga_dibayar = $total_harga - $nilai_diskon;  
  
?>  
  
<html>  
<head>  
<title>Geordi dan Daftar Peralatan Yang Dibeli</title>  
</head>  
<body>  
<center>  
  
<table border="1" cellspacing="0" cellpadding="3">  
  
<tr>  
<td colspan="4" align="center" valign="middle">  
<b>Daftar Pemesanan Peralatan Geordi La Forge - NCC1701D</b>  
</td>  
</tr>  
  
<tr>  
<td><b>Nama Peralatan</b></td>  
<td><b>Jumlah</b></td>  
<td><b>Harga Satuan</b></td>  
<td><b>Jumlah Harga</b></td>  
</tr>  
  
<?php  
  
// Mulai untuk mengisi tabel daftar dengan data yang ada  
  
?>
```

```
<tr>
<td align="left"><?php echo $alat_geordi1; ?></td>
<td align="right"><?php echo $jumlah_alat_geordi1; ?></td>
<td align="right"><?php echo $harga_alat_geordi1; ?></td>
<td align="right"><?php echo $total_alat_geordi1; ?></td>
</tr>

<tr>
<td align="left"><?php echo $alat_geordi2; ?></td>
<td align="right"><?php echo $jumlah_alat_geordi2; ?></td>
<td align="right"><?php echo $harga_alat_geordi2; ?></td>
<td align="right"><?php echo $total_alat_geordi2; ?></td>
</tr>

<tr>
<td align="left"><?php echo $alat_geordi3; ?></td>
<td align="right"><?php echo $jumlah_alat_geordi3; ?></td>
<td align="right"><?php echo $harga_alat_geordi3; ?></td>
<td align="right"><?php echo $total_alat_geordi3; ?></td>
</tr>

<tr>
<td align="left"><?php echo $alat_geordi4; ?></td>
<td align="right"><?php echo $jumlah_alat_geordi4; ?></td>
<td align="right"><?php echo $harga_alat_geordi4; ?></td>
<td align="right"><?php echo $total_alat_geordi4; ?></td>
</tr>

<tr>
<td colspan="3" align="right">Total Harga</td>
<td align="right"><?php echo $total_harga; ?></td>
</tr>

<tr>
<td colspan="3" align="right">
Diskon <?php echo "( $diskon % )"; ?></td>
<td align="right"><?php echo $nilai_diskon; ?></td>
</tr>

<tr>
<td colspan="3" align="right">Jumlah harus dibayar</td>
<td align="right"><?php echo $total_harga_dibayar; ?></td>
</tr>

</table>

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

Kelihatan seperti program yang panjang dan kompleks? Sebenarnya program ini sederhana saja, hanya karena kita belum menginjak pada cara mengatur aliran dan pengulangan program maka program ini menjadi panjang. Pada saatnya nanti kita akan mampu membuat program dengan maksud yang sama namun dengan cara yang lebih singkat. Sampai di sini, terpaksa Anda relakan saja harus mengetik kode yang cukup panjang terlebih dahulu. Hitung-hitung latihan membiasakan diri dalam struktur kalimat dan penyisipan kode PHP pada format HTML.

Tampilan hasil kode di atas adalah seperti di bawah ini.

Daftar Pemesanan Peralatan Geordi La Forge - NCC1701D			
Nama Peralatan	Jumlah	Harga Satuan	Jumlah Harga
Phaser	2	7500	15000
Tricorder	5	12500	62500
Visor	1	16000	16000
Analyzer Photonik	3	2300	6900
Total Harga			100400
Diskon (5 %)			5020
Jumlah harus dibayar			95380

Dalam contoh di atas, kita telah belajar cara melakukan operasi matematis untuk bilangan. Kita telah mencoba mengalikan, menjumlahkan, membagi dan mengurangi. Untuk jenis data string, kita dapat menggabungkan/menyambung dua buah string dengan operator titik ("."). Contoh berikut ini akan membantu kita untuk lebih mengerti.

```
<?php

// inisiasi variabel
$a = "USS Enterprise";
$b = "Menurut catatan kapten";
$c = "Mengunjungi Planet Vulcan;

// alternatif pertama
$salt1 = $a . " " . $c . ", " . $b . ".";

// alternatif kedua
$salt2 = $b . ", " . $a . " " . $c . ".";
```

```
?>

<html>
<head>
<title>Menggabungkan String</title>
</head>

<body>

String yang pertama adalah: <br>

<?php echo $alt1; ?>

<br><br>

String yang kedua adalah: <br>

<?php echo $alt2; ?>

</body>
</html>
```

Simpan kode di atas sebagai `coba5.php`, dan cobalah menjalankannya dari *browser*. Maka di layar akan muncul hasil seperti ini:

```
String yang pertama adalah:
USS Enterprise Mengunjungi Planet Vulcan, Menurut catatan kapten.

String yang kedua adalah:
Menurut catatan kapten, USS Enterprise Mengunjungi Planet Vulcan.
```

USS Enterprise menghadapi Romulan Warbird

Jika Anda berpengalaman dalam bahasa pemrograman C, Anda pasti familiar dengan perintah `include` yang hampir selalu muncul di awal kode program C. PHP memiliki dua macam fungsi untuk maksud yang sama, namun dengan karakteristik yang khas untuk masing masing fungsi. Fungsi yang pertama adalah fungsi `include()` dan yang kedua adalah fungsi `require()`. Anda dapat mencoba contoh berikut ini sebagai gambarannya.

```
<html>
<head>
<title>Persenjataan      dan      Perlengkapan      Perang
Enterprise</title>
</head>
```

```
<?php

// Standar Senjata Kapal Perang Kelas Galaxy
require("torpedo.php");
require("laser.php");

// Standar Perisai Kapal Perang Kelas Galaxy
include("shielding.php");

// Standar Mesin Penggerak Kapal Perang Kelas Galaxy
include("impuls.php");
include("warp.php");

?>

<body>
LCAR: Cek kesiapan perlengkapan perang USS Enterprise
NCC-1701-D
<ol type="1">
<li> Torpedo : <?php echo $torpedo; ?>
<li> Laser : <?php echo $laser; ?>
<li> Perisai : <?php echo $shielding; ?>
<li> Mesin Impuls : <?php echo $impuls; ?>
<li> Mesin Warp : <?php echo $warp; ?>
</ol>

<br>
Commander La Forge, segera laporkan semua sistem persenjataan
telah dicek dan berfungsi dengan baik.
USS Enterprise siap menghadapi Kapal Romulan. <br>

</body>
</html>
```

Simpan kode di atas dengan nama `coba6.php` dan panggil melalui *browser*. Hopla, akan kita dapatkan begitu banyak pesan kesalahan (*error*). Tentu saja, karena kita belum membuat file-file `torpedo.php`, `laser.php`, `shielding.php`, `impuls.php`, dan `warp.php`. Berikut ini adalah kode program untuk file-file tersebut.

[`torpedo.php`]

```
<?php

$torpedo = "Four Bays Photon Torpedo";

?>
```

```
[laser.php]
<?php

$laser = "Six Laser Canons";

?>
```

```
[shielding.php]
<?php

$shielding = "EM Polarization Shielding";

?>
```

```
[impuls.php]
<?php

$impuls = "Federation Impulse Power System";

?>
```

```
[warp.php]
<?php

$warp = "Matter/Antimatter Reactor (Warp Core)";

?>
```

Setelah Anda menuliskan semua file yang dibutuhkan seperti contoh di atas, maka jika Anda memanggil program utamanya (`coba6.php`), PHP secara otomatis akan mengikutsertakan program-program lain yang ditentukan melalui `require()` dan `include()`, membaca variabel `$torpedo`, `$laser`, `$shielding`, `$impuls`, dan `$warp`, serta menampilkan isi atau nilai dari variabel tersebut pada halaman yang kita panggil.

Apakah `require()` dan `include()` itu benar-benar sama cara kerjanya? Tentu saja tidak, sebab jika sama fungsinya tentu tidak selayaknya dibedakan fungsinya. Perbedaan mendasar antara kedua fungsi ini adalah:

- Fungsi `require()` akan selalu digantikan oleh isi dari file yang ditunjuk dalam fungsi ini dan tidak dapat digunakan dalam percabangan/perkondisian (seperti perkondisian

"jika ini maka require file anu") , karena file yang ditunjuk akan selalu direferensi tanpa peduli kondisi struktur/aliran *script*.

- Fungsi `include()` akan mengatur pembacaan file yang ditunjuk dapat sesuai dengan kondisi struktur/aliran *script*, sehingga fungsi ini dapat digunakan pada percabangan/perkondisian.

Melihat ciri-ciri di atas, `require()` akan sesuai digunakan untuk mereferensi file yang berisikan variabel dan fungsi-fungsi global yang digunakan pada seluruh bagian dari *script* utama. Sementara `include()` umumnya digunakan untuk menyisipkan kode program/*script* atau tag HTML pada program/*script* utama, misalkan untuk *header* atau *footer* setiap halaman dalam sebuah situs.

Catatan yang penting untuk kedua fungsi ini, parser PHP akan meninggalkan mode PHP dan kembali ke mode HTML standar pada saat membaca file yang ditunjukkan oleh kedua fungsi ini. Itu sebabnya pada contoh di atas, semua file yang ditunjuk oleh fungsi-fungsi ini selalu dimulai dengan tag `<?php` dan diakhiri dengan tag `?>` untuk mengembalikan mode file ke mode *script* PHP.

Contoh penggunaan fungsi `include()` yang umum untuk *header* dan *footer* pada halaman HTML.

```
<html>
<head>
<title>Title Halaman</title>
</head>

<body>

<?php

include("header.html");

?>

. . . . . isi halaman HTML . . . . .

<br>

<?php

include("footer.html");

?>

</body>
```

```
</html>
```

Dengan misalnya header .html berisi:

```
<table width="100%" bgcolor="#A0A0A0">
<tr>
<td bgcolor="#0000F0" align="center">LCAR : USS
Enterprise</td>
</tr>
</table>
```

dan footer .html berisi misalkan:

```
<table width="100%" bgcolor="#A0A0A0">
<tr>
<td bgcolor="#0000F0" align="center">
<font size="-1">(c) United Federation of Planets.</font>
</td>
</tr>
</table>
```

Dengan struktur halaman web seperti dicontohkan ini, maka kita dapat dengan mudah membuat keseragaman pada halaman-halaman situs/aplikasi yang kita bangun. Perubahan pada *header* dan *footer* dapat dilakukan dengan mengedit kedua file ini saja, tanpa perlu mengganti semua halaman situs/aplikasi yang telah dibangun. Bayangkan jika ada 100 halaman, tentu akan sangat memberikan kita waktu luang untuk bersantai daripada jika kita harus melakukan *update* halaman satu persatu untuk perubahan ini.

Sampai di sini, kita telah belajar konsep membangun blok PHP, sedikit perintah dasar PHP untuk tampilan layar di *browser*, dasar-dasar variabel, operasi matematis sederhana, cara penyisipan file pada *script* PHP. Pengetahuan ini akan dipakai sebagai dasar untuk melanjutkan pelajaran bagaimana membuat dan mengoperasikan masukan lewat form HTML. Anda perlu bersabar menantikan bagian kedua dari tulisan ini.

Bagian 2: Holodeck di USS Enterprise NCC-1701D

Pada bagian 1 dari artikel ini, kita telah belajar mengenai variabel dan operasi matematika sederhana terhadap variabel di PHP. Konsep dasar `require()` dan `include()` juga sudah kita kenal. Kali ini kita akan mencoba untuk melanjutkan sedikit mengenai cakupan variabel, sebelum berlanjut pada pengolahan form HTML.

Bar Ten-Fourty

Dalam pemrograman, seringkali kita ingin menggunakan variabel dengan cakupan yang berbeda-beda. Ada variabel yang kita inginkan untuk digunakan di seluruh program atau sering disebut variable global, ada variabel yang hanya ingin kita gunakan di dalam cakupan sebuah fungsi atau prosedur.

Variabel dalam PHP memiliki cakupan dalam konteks variabel itu didefinisikan. Umumnya variabel PHP hanya memiliki cakupan tunggal saja. Anda tidak perlu khawatir dengan keterbatasan ini, karena ternyata kita dapat memperluas cakupan variabel dengan memanfaatkan fungsi `require()` dan `include()` yang telah kita pelajari. Untuk lebih jelas, mari kita lihat contohnya pada program `coba7.php` di bawah ini.

```
<?php
$bartender = "Guinan";
include ("ten_fourty_bar.inc");
?>
```

Dalam contoh di atas, variabel `$bartender` akan memiliki cakupan pada file skrip `ten_fourty_bar.inc` (atau dapat pula dilihat secara sebaliknya). Jika kita membuat file skrip `ten_fourty_bar.inc` berisi perintah php seperti di bawah ini.

```
<?php
echo "Bartender di Bar Ten-Fourty saat ini adalah :
$bartender";
```

?>

Maka, hasil eksekusi program `coba7.php` adalah sebagai berikut

```
Bartender di Bar Ten-Fourty saat ini adalah : Guinan
```

Namun jika pada program `ten_fourty_bar.inc` dibuat sebuah fungsi, maka variabel `$bartender` tidak dapat mencakup ke dalam fungsi tersebut, kecuali jika `$bartender` dimasukkan sebagai argumen dalam fungsi tersebut atau `$bartender` dalam fungsi tersebut dideklarasikan sebagai variabel global dengan perintah deklarasi `global` atau lewat variabel `$GLOBALS[]`. Lebih detail mengenai penggunaan variabel dalam fungsi akan kita bahas lagi pada saat kita membicarakan mengenai perancangan fungsi.

Kode program PHP disimpan sebagai sebuah file skrip. Jika aplikasi web yang kita bangun memperlakukan file-file skrip itu sebagai modul dari aplikasi, maka mungkin kita akan mendapatkan masalah jika kita ingin menggunakan variabel lintas file skrip tersebut.

Kita dapat saja membuat sebuah file yang berisi nilai variabel yang akan dipanggil lintas file. File ini kemudian selalu di-`include()` pada modul-modul yang membutuhkan variabel tersebut. Namun solusi semacam ini tidak memungkinkan kita untuk melakukan perubahan dinamis pada variabel tersebut. Cara ini lebih tepat digunakan untuk konstanta global.

Cara lain adalah dengan melewati (*passing thru*) nilai variable dari satu file skrip/dokumen ke file skrip/dokumen lain yang dipanggil setelahnya. Terdapat dua cara untuk melakukan ini yaitu dengan:

- Menambahkan langsung variabel dan nilainya pada URL file skrip atau dokumen yang dipanggil. Contohnya kita memiliki host `localhost` dan file yang akan dipanggil adalah `deep_space_9_bar.php` terletak pada direktori `root`. Variable `$bartender` dapat dikenali nilainya oleh file `deep_space_9_bar.php` jika kita memanggil file tersebut dengan URL : http://localhost/deep_space_9_bar.php?bartender=Guinan. Jika kita ingin melewati variabel lain selain `$bartender`, misalkan `$pengisiacara="Data"`, maka kita dapat menambahkan URL tersebut sehingga menjadi http://localhost/deep_space_9_bar.php?bartender=Guinan&pengisiacara=Data. Antar variabel dipisahkan dengan karakter "&".

- Menggunakan cara form HTML baik dengan metode POST/GET yang akan melewati nilai dari *tag* <INPUT> untuk ditangkap sebagai variabel oleh file yang dituju dalam ACTION. Lebih jelas lagi mengenai hal ini akan kita bicarakan pada saat membahas tentang form HTML.

PHP memiliki keunikan lain karena dapat membuat nama variabel dari nilai variabel yang lain. Lihatlah contoh berikut ini.

```
<?php

$bartender = "Guinan";

$$bartender = "Bartender Misterius";

echo "$bartender, ${$bartender}\n";

echo "$bartender, $Guinan\n";

?>
```

Baris pertama adalah deklarasi variabel `$bartender` sekaligus pengisian nilainya dengan "Guinan", sedangkan baris yang kedua sebenarnya adalah deklarasi variabel dengan nama `$Guinan` yang diisi nilai "Bartender Misterius". Sekalipun perintahnya berbeda, baris ketiga dan keempat memberikan hasil keluaran yang sama yaitu.

```
Guinan, Bartender Misterius
```

Penulisan `$$bartender` dan `${$bartender}` adalah variasi cara untuk menyebut variabel yang sama (dalam konteks ini adalah variabel `$Guinan`). Tanda kurung kurawal "{...}" akan banyak gunanya jika kita telah mulai menggunakan nama variabel dari nilai variabel lain dalam larik (*array*) variabel. Misalnya variabel `${$bartender[1]}` artinya variabel dengan nama dari isi variabel indeks 1 dari larik variabel `$bartender`, sedangkan variabel `${$bartender}[1]` adalah variabel indeks 1 dari larik variabel dengan nama dari isi variabel `$bartender`.

Ketika saya menuliskan paragraf di atas, saya menjadi sedikit mengkhawatirkan kondisi Anda setelah membaca kalimat terakhir saya. Pegang kening Anda, dan jika panas, Anda dapat mencoba lagi membaca dan mengerti paragraf tersebut di kesempatan yang lain atau bisa juga Anda membaca keterangan berikut ini.

Notasi `$$bartender[1]` artinya kita memiliki larik variabel `$bartender`, misalkan isi `$bartender[1]="Guinan"`, `$bartender[2]="Q"`, `$bartender[3]="Riker"`, dan seterusnya. Maka notasi `$$bartender[1]` yang kita maksudkan adalah variabel `$Guinan` bukan `$Q` atau pun `$Riker`. Pada notasi kedua yakni `$$bartender`[1] artinya kita memiliki variabel non larik `$bartender` yang misalkan berisi nilai "Guinan", sehingga notasi `$$bartender`[1] secara implisit berarti kita memiliki variabel larik `$Guinan[1]`, `$Guinan[2]`, dan seterusnya.

Apakah Anda telah cukup puas dengan pembahasan mengenai variabel? Jika belum puas maka dengan sangat menyesal saya katakan bahwa saat ini tiba-tiba *mood* saya untuk membahas variabel telah hilang. *Mood* saya sekarang adalah membicarakan masalah form.

Pintu Masuk Holodeck di USS Enterprise

Form merupakan cara termudah, terumum, dan tercepat untuk membuat situs web Anda lebih hidup dan mampu berinteraksi dengan pengunjung yang mengaksesnya. Banyak keuntungan yang bisa Anda dapatkan dari penggunaan form dalam dokumen HTML, misalkan Anda dapat menanyakan apakah pengunjung situs Anda menyukai produk Anda atau bahkan Anda dapat meminta kepada pengunjung wanita yang cantik untuk menuliskan nomor teleponnya bagi Anda. Tentu saja jika mereka bersedia. Mungkin tidak banyak hasilnya, namun tidak ada salahnya berharap akan ada yang jatuh dalam perangkap Anda.

PHP membuat hidup kita menjadi lebih mudah karena PHP dapat membuat pemrosesan form untuk mengambil data masukan dari pengguna menjadi lebih sederhana dan cepat. Jangan pernah membayangkan Anda saat ini dapat menemukan kemudahan ini pada bahasa Perl atau C/C++, misalnya. Jika kebetulan suatu saat Anda menemukan kemudahan ini pada Perl, sebenarnya ini bukan salah PHP, namun hanya karena kebetulan Perl yang berkembang terlalu pesat.

Sebagai permulaan, Anda dapat mencoba membuat file HTML berikut ini yang dapat Anda simpan dengan nama `login.html`.

```
<html>

<head>
<basefont face="Arial">
</head>

<body>

<center>
<form method="GET" action="proseslogin.php">
<table cellspacing="5" cellpadding="5" border="1">

<tr>
<td colspan="2" align="center">
NCC-1701D USS Enterprise<br>
Fasilitas Holodeck
</td>
</tr>

<tr>
<td>
<font size="-1">Silakan Masukkan Nama Anda
</td>
<td>
<input type="text" name="namaofficer" size="20">
</td>
</tr>

<tr>
<td colspan="2" align="center">
<input type="submit" name="loginofficer" value="Login">
</td>
</tr>

</table>
</form>

</center>

</body>

</html>
```

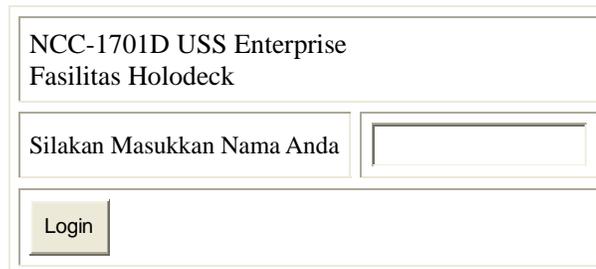
Hal kritical pada file di atas yang akan kita bahas adalah adalah *tag* <FORM>.

```
<form method="GET" action="proseslogin.php">  
  
.....  
  
</form>
```

Atribut `action` pada tag `<FORM>` menunjukkan nama dari file skrip di sisi server, yang dalam kasus kita ini akan bertugas untuk memproses informasi yang dimasukkan ke form. Sementara itu atribut `method` akan menentukan tata cara informasi akan dilewatkan ke file skrip yang ditunjuk oleh atribut `action`.

Dalam standar HTML, dikenal dua macam `method` untuk memproses informasi yang dimasukkan ke form agar dapat diproses oleh file skrip yang dituju, yaitu `GET` dan `PUT`. Penggunaan `GET` akan menyebabkan seluruh isian form dilewatkan ke file skrip yang dituju dengan cara ditambahkan pada URL file skrip yang dituju, sementara `PUT` tidak akan menambahkan URL file skrip yang dituju dengan hasil isian form. Untuk lebih jelasnya, Anda bisa kembali membuka referensi standar HTML yang Anda miliki.

File ini jika dipanggil lewat browser akan memberikan tampilan sebagai berikut.



NCC-1701D USS Enterprise Fasilitas Holodeck	
Silakan Masukkan Nama Anda	<input type="text"/>
<input type="button" value="Login"/>	

Untuk kasus kita saat ini, kita baru setengah jalan. Kita harus membuat file `proseslogin.php` yang akan menerima data isian dari file `login.html`. File skrip yang akan kita buat ini akan bertugas melakukan pengecekan nama officer yang akan memasuki ruang *holodeck*. Berhubung sampai saat ini kita belum mempelajari mengenai pernyataan kondisional (conditional statement) dan operator logika, maka sementara ini file `proseslogin.php` hanya akan kita buat untuk menunjukkan kepada Anda bagaimana data dikirimkan oleh form pada file `login.html` dan diproses atau digunakan pada file `proseslogin.php`.

Inilah file `proseslogin.php` yang kita akan kita buat.

```
<html>
<head>
<basefont face="Arial">
</head>

<body>
<center>
<font face="Arial" size="-1">
Hmm, pernahkah Anda berimajinasi, <? echo $namaofficer; ?> ?
<P>
Holodeck mampu membuat Anda berimajinasi dan menjalaninya
hampir tanpa batas.
<P>
Anda siap memasuki holodeck?
</font>
</center>
</body>

</html>
```

Misalkan Anda mengisikan data pada form, misalkan saja "Parto", kemudian anda tekan tombol "Login", maka tampak pada browser Anda sebagai berikut.

Hmm, pernahkah Anda berimajinasi, Parto ?

Holodeck mampu membuat Anda berimajinasi dan menjalaninya hampir tanpa batas.

Anda siap memasuki holodeck?

Seperti yang Anda lihat, pada saat sebuah form dikirimkan ke sebuah skrip PHP, semua pasangan variabel dan nilainya yang ada pada form tersebut secara otomatis tersedia untuk digunakan oleh skrip PHP tersebut. Dalam contoh di atas, saat form yang ada pada login.html dikirim, variabel \$namaofficer otomatis terbentuk pada skrip PHP proseslogin.php dan variabel ini langsung terisi dengan data yang diisikan pada form oleh pengguna.

Jika Anda mencoba melakukan hal yang sama dengan Perl, maka Anda perlu secara

eksplisit menulis kode untuk melakukan ekstrak dan mengambil nilai dari variabel-variabel dalam form. PHP telah secara otomatis melakukan semua ini untuk Anda, sehingga kode program Anda akan lebih sederhana dan proses *development* aplikasi menjadi lebih cepat.

Kehati-hatian Dalam Memilih

Saya tidak akan mengelak jika Anda memprotes contoh di atas terlalu sederhana. Memang saat ini kita belum mulai melakukan seleksi calon pengguna holodeck. Untuk dapat melakukannya, mari kita mempelajari pernyataan kondisional dan operator logika. Bentuk paling dasar dari pernyataan kondisional adalah perbandingan, misalnya "jika ini sama dengan itu maka lakukan hal ini, dan seterusnya".

PHP memiliki operator-operator logika yang sangat berguna untuk menyusun pernyataan kondisional. Berikut ini adalah daftarnya.

Misalkan $\$alpha=7$ dan $\$beta=4$.

Operator	Arti	Ekspresi	Hasil Evaluasi Nilai
==	sama dengan	$\$alpha == \$beta$	False
!=	tidak sama dengan	$\$alpha != \$beta$	True
>	lebih besar daripada	$\$alpha > \$beta$	True
<	lebih kecil daripada	$\$alpha < \$beta$	False
>=	lebih besar atau sama dengan	$\$alpha >= \$beta$	True
<=	lebih kecil atau sama dengan	$\$alpha <= \$beta$	False

PHP4 juga memperkenalkan sebuah operator logika baru, yang melakukan pengecekan baik kesamaan nilai maupun jenis nilai dari variabel. Operator ini adalah `===`. Pada bagian akhir bagian ini akan ditunjukkan ilustrasi penggunaan operator ini.

Officer Yang Berhak Masuk Holodeck

Bentuk paling sederhana dari pernyataan kondisional dalam PHP adalah pernyataan "if", yang kurang lebih adalah seperti di bawah ini:

```
if (kondisi)
{
    lakukan hal ini!!;
}
```

Bagian *kondisi* adalah merupakan ekspresi kondisional yang akan dievaluasi apakah hasilnya *true* (benar) atau *false* (salah). Jika hasil pengecekan *kondisi* bernilai *true*, maka seluruh kode PHP yang ada dalam blok *if* (di antara dua kurung kurawal) akan dieksekusi. Jika tidak (hasil pengecekan *false*), maka seluruh kode PHP dalam blok *if* akan dilewati dan eksekusi program akan dilanjutkan ke baris setelah blok *if*.

Sekarang kita coba memodifikasi program `proseslogin.php` dengan membuat sistem validasi/otentikasi sederhana untuk nama officer yang diperkenankan memasuki holodeck. Misalkan akses ke holodeck hanya diperbolehkan untuk officer "Riker".

```
<html>
<head>
<basefont face="Arial">
</head>

<body>
<center>

<?php
// validasi nama officer dan tampilkan pesan yang sesuai
if ($namaofficer == "Riker")

    {

?>
<font face="Arial" size="-1">
Hmm, pernahkah Anda berimajinasi, <? echo $namaofficer; ?> ?
<P>
Holodeck mampu membuat Anda berimajinasi dan menjalaninya
hampir tanpa batas.
<P>
Selamat datang di holodeck USS Enterprise. <BR>
Anda siap memasuki holodeck?
</font>

<?php

    }

?>

<?php
// jika nama officer tidak sesuai
```

```
if ($namaofficer != "Riker")
{
?>
<font face="Arial" size="-1">
Hmm, Anda ingin berimajinasi, <? echo $namaofficer; ?> ?
<P>
Sayang sekali, Anda dalam tugas.
<P>
Anda tidak diperkenankan memasuki holodeck.
</font>

<?php
    }
?>

</center>
</body>

</html>
```

Anda dapat menyusun beberapa blok if secara bertumpuk (*nested*) untuk melakukan penyeleksian lebih ketat terhadap beberapa kondisi. Misalkan Anda ingin mencari Lieutenant Worf, Anda dapat menyusun pengkondisian seperti di bawah ini.

```
<?
if ($pekerjaan == "Officer Starfleet")
{
if ($pesawat == "USS Enterprise")
{
if ($ras == "Klingon")
{
$nama = "Worf";
}
}
}
?>
```

Jika Tidak, Maka ...

Selain bentuk "if" seperti yang telah kita pelajari, PHP juga memiliki bentuk pernyataan kondisional "if-else", yang selain memiliki blok perintah PHP yang dieksekusi jika kondisi bernilai *true* juga memiliki blok perintah PHP yang akan dijalankan jika kondisi bernilai *false*.

Konstruksi "if-else" adalah seperti ini.

```
if (kondisi)
{
    lakukan hal ini!;
}
else
{
    lakukan hal itu!;
}
```

Dengan konstruksi ini, maka kita dapat membuat program `proseslogin.php` menjadi lebih efisien daripada menggunakan dua buah blok `if`.

```
<html>
<head>
<basefont face="Arial">
</head>

<body>
<center>

<?php
// validasi nama officer dan tampilkan pesan yang sesuai
if ($namaofficer == "Riker")

{

?>
<font face="Arial" size="-1">
```

```
    Hmm, pernahkah Anda berimajinasi, <? echo $namaofficer; ?> ?  
<P>  
    Holodeck mampu membuat Anda berimajinasi dan menjalaninya  
    hampir tanpa batas.  
<P>  
    Selamat datang di holodeck USS Enterprise. <BR>  
    Anda siap memasuki holodeck?  
</font>  
  
<?php  
  
    }  
else  
{  
    // jika nama officer tidak sesuai  
?>  
  
<font face="Arial" size="-1">  
    Hmm, Anda ingin berimajinasi, <? echo $namaofficer; ?> ?  
<P>  
    Sayang sekali, Anda dalam tugas.  
<P>  
    Anda tidak diperkenankan memasuki holodeck.  
</font>  
  
<?php  
  
    }  
?>  
  
</center>  
</body>  
  
</html>
```

Menu Harian Holodeck USS Enterprise

PHP juga menyediakan bentuk pernyataan kondisional "if-elseif-else" untuk menangani kemungkinan yang lebih banyak dari pemilihan kondisi. Bentuk pernyataan kondisional ini adalah seperti di bawah ini.

```
if (kondisi pertama benar)  
{
```

```
        lakukan tindakan 1;
    }
elseif (kondisi kedua benar)
    {
        lakukan tindakan 2;
    }
elseif (kondisi ketiga benar)
    {
        lakukan tindakan 3;
    }

..... dan seterusnya .....

else

    {
        lakukan tindakan yang lain;
    }
```

Mari kita lihat contoh penerapannya untuk membuat pilihan menu harian di holodeck USS Enterprise.

```
<html>
<head>
<style type="text/css">
td {font-family: Arial;}
</style>
</head>

<body>

<font face="Arial" size="+2">
Pilihan Menu Harian Holodeck USS Enterprise
</font>

<form method="GET" action="prosesmenu.php">
<table cellspacing="5" cellpadding="5" border="0">

<tr>
<td align="center">
Pilih Hari
</td>
<td align="right">
<select name="hari">
<option value="Senin">Senin
```

```
<option value="Selasa">Selasa
<option value="Rabu">Rabu
<option value="Kamis">Kamis
<option value="Jumat">Jumat
<option value="Sabtu">Sabtu
<option value="Minggu">Minggu
</select>
</td>
</tr>

<tr>
<td colspan="2" align="center">
<input type="submit" value="Klik Di Sini!">
</td>
</tr>

</table>
</form>
</body>

</html>
```

Dengan skrip di atas kita bermaksud membuat pilihan menu harian di holodeck. Simpanlah skrip di atas dengan nama `menu.php`. Untuk dapat bekerja, perlu kita buat skrip di bawah ini, yang akan kita simpan dengan nama `prosesmenu.php`.

```
<?php

if ($hari == "Senin")

    (
        $topik = "Romeo dan Juliet (Shakespeare)";
    )
elseif ($hari == "Selasa")
    (
        $topik = "Petualangan Robin Hood";
    )
elseif ($hari == "Rabu")
    (
        $topik = "Jurassic Park";
    )
elseif ($hari == "Kamis")
    (
        $topik = "Indiana Jones";
    )
```

```
elseif ($hari == "Jumat")
(
$topik = "Final Fantasy";
}
else
(
$topik = "Maaf, Holodeck USS Enterprise tutup saat
weekend.";
}

?>

<html>
<head>
<basefont face="Arial">
</head>

<body>
Menu Petualangan Holodeck USS Enterprise<br>
Hari <? echo $hari; ?> : <br>
<b><? echo $topik; ?><b>

</body>
</html>
```

Dengan dua skrip di atas, maka kita dapat memilih nama hari pada skrip pertama dan menekan tombol untuk mengaktifkan skrip kedua yang akan memilihkan topik holodeck untuk hari yang dipilih. Patut diingat, bahwa begitu sebuah kondisi dalam bentuk "if-elseif-else" ditemukan bernilai *true* maka seluruh kode dalam blok kondisi yang tersebut akan dieksekusi, dan berikutnya aliran program akan dilanjutkan pada baris kode setelah blok "if-elseif-else". Jadi dalam bentuk pernyataan kondisional seperti ini tidak ada dua buah blok kondisi yang akan dijalankan secara bersamaan. Hanya kondisi yang pertama kali ditemukan bernilai *true* yang akan dijalankan, selebihnya akan dilewatkan. Kode berikut ini akan memberi ilustrasi secara lebih baik.

```
<?php

$alpha = 12;
$beta = 15;
$delta = 19;
$gamma = 24;
```

```
$kondisi = "";

if ($alpha < $beta)
    {
    $kondisi .= "Alpha Lebih Kecil Daripada Beta <br>";
    }
elseif ($alpha < $delta)
    {
    $kondisi .= "Alpha Lebih Kecil Daripada Delta <br>";
    }
elseif ($alpha < $gamma)
    {
    $kondisi .= "Alpha Lebih Kecil Daripada Gamma <br>";
    }
else
    {
    $kondisi .= "Tidak Ada Kondisi Yang Sesuai <br>";
    }

?>

<html>
<head>
<basefont face="Arial">
</head>

<body>

<?php

echo $kondisi;

?>

</body>
</html>
```

Jika skrip di atas dijalankan, sebenarnya ada 3 kondisi yang bernilai *true*, namun pada kenyataannya hanya kode pada blok kondisi pertama ($\$alpha < \$beta$) saja yang dijalankan. Dua kondisi lainnya yang juga bernilai *true* tidak dijalankan, karena bentuk "if-elseif-else" hanya mengeksekusi blok kondisi pertama yang ditemukan bernilai *true*. kemudian akan dilanjutkan dengan mengeksekusi baris perintah/kode setelah bentuk pernyataan kondisional.

Notasi \$kondisi .= "bla-bla-bla" adalah notasi penyambungan sebuah untai (*string*). Notasi ini mirip dengan di bahasa pemrograman C/C++, dan berlaku juga untuk operator aritmatika. Tabel berikut ini menunjukkan notasi normal dan notasi singkatannya yang berlaku di PHP.

Notasi Normal	Notasi Singkat	Keterangan
<code>\$a = \$a + 1</code>	<code>\$a++</code>	Tambahkan 1 ke \$a dan simpan hasilnya di \$a
<code>\$a = \$a + \$x</code>	<code>\$a += \$x</code>	Tambahkan \$x ke \$a dan simpan hasilnya di \$a
<code>\$a = \$a - 1</code>	<code>\$a--</code>	Kurangkan 1 dari \$a dan simpan hasilnya di \$a
<code>\$a = \$a - \$x</code>	<code>\$a -= \$x</code>	Kurangkan \$x dari \$a dan simpan hasilnya di \$a
<code>\$a = \$a . \$x</code>	<code>\$a .= \$x</code>	Sambungkan string \$x ke string \$a dan simpan hasilnya di \$a

Dan, Atau, Tidak

Masih ingatkah dengan skrip mencari Worf? Mari kita lihat lagi skrip tersebut.

```
<?
if ($pekerjaan == "Officer Starfleet")
{
    if ($pesawat == "USS Enterprise")
    {
        if ($ras == "Klingon")
        {
            $nama = "Worf";
        }
    }
}
?>
```

Anda bisa tidak setuju dengan saya, namun sebenarnya skrip di atas terlalu kompleks dan sedikit mengerikan. PHP menawarkan juga operator logika yang dapat digunakan untuk

menyederhanakan skrip di atas. Tabel berikut ini akan menunjukkan operator logika dalam PHP.

Operator	Arti	Ekspresi	Hasil Evaluasi Nilai
&&	AND	<code>\$alpha == \$delta && \$alpha > \$beta</code>	True
		<code>\$alpha && \$beta < \$beta</code>	False
	OR	<code>\$alpha == \$delta alpha < \$beta</code>	True
		<code>\$alpha > \$delta alpha < \$beta</code>	False
!	NOT	<code>!\$alpha</code>	False

Dengan pengetahuan logika ini, maka kita bisa menulis kembali skrip pencarian Worf dengan lebih sederhana.

```
<?
if ($pekerjaan == "Officer Starfleet" && $pesawat == "USS
Enterprise" && $ras == "Klingon")
{
    $nama = "Worf";
}
?>
```

Bukankah skrip di atas lebih sederhana?

Sekali Lagi, Memilih Di Antara Banyak Pilihan

PHP juga menyediakan alternatif pernyataan kondisional selain dengan keluarga "if-else" yaitu bentuk "switch-case", dengan bentuk pernyataan seperti berikut ini.

```
switch (variabel_penentu)
```

```
{  
  
    case (kondisi_pertama_benar)  
        Lakukan Tindakan Untuk Kondisi Pertama;  
  
    case (kondisi_kedua_benar)  
        Lakukan Tindakan Untuk Kondisi Kedua;  
  
    case (kondisi_ketiga_benar)  
        Lakukan Tindakan Untuk Kondisi Ketiga;  
  
    ..... dan seterusnya .....  
  
}
```

Kini kita akan mencoba menulis kembali kode program prosesmenu.php yang digunakan untuk menampilkan menu harian holodeck. Dengan menggunakan bentuk "switch-case", skrip program akan menjadi seperti di bawah ini.

```
<?php  
  
// variabel penentu dalam hal ini adalah $hari yang dipilih  
pengguna  
switch ($hari)  
  
    {  
  
        // kondisi pertama  
        case "Senin":  
            $topik = "Romeo dan Juliet (Shakespeare)";  
            break;  
  
        // kondisi kedua  
        case "Selasa":  
            $topik = "Petualangan Robin Hood";  
            break;  
  
        // kondisi ketiga  
        case "Rabu":  
            $topik = "Jurassic Park";  
            break;  
  
    }
```

```
// kondisi keempat
case "Kamis":
$topik = "Indiana Jones";
break;

// kondisi kelima
case "Jumat":
$topik = "Final Fantasy";
break;

// jika selain kondisi yang di atas
default:
$topik = "Maaf, Holodeck USS Enterprise tutup saat
weekend.";
break;

}

?>

<html>
<head>
<basefont face="Arial">
</head>

<body>
Menu Petualangan Holodeck USS Enterprise<br>
Hari <? echo $hari; ?> : <br>
<b><? echo $topik; ?><b>

</body>
</html>
```

Ada beberapa kata kunci yang penting dalam penggunaan pernyataan "switch-case". Pertama adalah perintah `break` yang digunakan untuk keluar dari blok "switch" dan melanjutkan ke baris perintah sesudah blok tersebut setelah ditemukan sebuah kondisi `true` yang pertama. Tanpa penggunaan `break`, maka case berikutnya akan kembali dievaluasi walaupun telah case sebelumnya telah ditemukan bernilai `true`. Kata `default` digunakan untuk "menangkap" kondisi dimana nilai variable penentu tidak memenuhi semua kriteria/kondisi pada case-case yang ada.

Bersatu Kita Teguh

Sampai sejauh ini, program pengolahan form yang kita buat selalu menggunakan dua halaman web yaitu satu halaman HTML yang berisi form dan satu lagi adalah halaman

38

skrip PHP untuk memproses masukan form dan menghasilkan keluaran yang sesuai. PHP sesungguhnya menyediakan cara yang lebih baik untuk dapat menggabungkan dua halaman tersebut menjadi satu halaman saja, dengan cara menangkap nilai dari variabel yang dikirimkan oleh tombol pemroses di form.

Telah kita ketahui bahwa saat form dikirimkan ke skrip PHP, seluruh variabel form akan menjadi tersedia dalam lingkungan skrip PHP. Tombol pemroses, juga akan mengirimkan nilai dari variabel sesuai namanya, jika tombol pemroses ini ditekan dengan tujuan mengirimkan isian form. Dengan melakukan pengecekan terhadap ada tidaknya nilai variabel dari tombol pemroses, maka programmer dapat menggunakan file PHP tunggal untuk menghasilkan baik form isian maupun keluarannya jika isi form dikirimkan.

Mari kita coba menggabungkan dua halaman menjadi satu halaman skrip PHP dalam kasus menu harian holodeck. Berikut ini adalah skrip gabungannya, misalkan kita simpan dalam nama `menu.php`.

```
<?php

if (!$proses)

    {
    // jika $proses tidak memiliki nilai, artinya adalah
    // form tidak dalam proses pengiriman, maka skrip akan
    // menampilkan form isian.

    ?>

<html>
<head>
<style type="text/css">
td {font-family: Arial;}
</style>
</head>

<body>

<font face="Arial" size="+2">
Pilihan Menu Harian Holodeck USS Enterprise
</font>

<form method="GET" action="<? echo $PHP_SELF; ?>">
<table cellspacing="5" cellpadding="5" border="0">
```

```
<tr>
<td align="center">
Pilih Hari
</td>
<td align="right">
<select name="hari">
<option value="Senin">Senin
<option value="Selasa">Selasa
<option value="Rabu">Rabu
<option value="Kamis">Kamis
<option value="Jumat">Jumat
<option value="Sabtu">Sabtu
<option value="Minggu">Minggu
</select>
</td>
</tr>

<tr>
<td colspan="2" align="center">
<input type="submit" name="proses" value="Klik Di Sini!">
</td>
</tr>

</table>
</form>
</body>

</html>

<?php
    }

else

    {
    // jika $proses memiliki nilai, berarti data isian
    // form sedang dikirim, maka skrip akan memproses
    // isian form.

    // variabel penentu dalam hal ini adalah $hari yang
    dipilih pengguna
    switch ($hari)

        {

            // kondisi pertama
            case "Senin":
                $topik = "Romeo dan Juliet (Shakespeare)";
```

```
break;

// kondisi kedua
case "Selasa":
$topik = "Petualangan Robin Hood";
break;

// kondisi ketiga
case "Rabu":
$topik = "Jurassic Park";
break;

// kondisi keempat
case "Kamis":
$topik = "Indiana Jones";
break;

// kondisi kelima
case "Jumat":
$topik = "Final Fantasy";
break;

// jika selain kondisi yang di atas
default:
$topik = "Maaf, Holodeck USS Enterprise tutup saat
weekend.";
break;

}

?>

<html>
<head>
<basefont face="Arial">
</head>

<body>
Menu Petualangan Holodeck USS Enterprise<br>
Hari <? echo $hari; ?> : <br>
<b><? echo $topik; ?><b>

</body>
</html>

<?php
}
```

```
?>
```

Dengan skrip di atas, maka baik form maupun proses untuk dapat menghasilkan keluaran dapat disatukan dalam satu halaman skrip PHP. Variabel \$proses adalah variabel yang dihasilkan jika tombol pemroses ditekan. Ada tidaknya nilai variabel ini yang akan menentukan apakah skrip ini memberikan keluaran berupa form ataukah akan memproses hasil isian form.

Untuk agar form yang dikirim dapat menghasilkan variabel \$proses pada lingkungan skrip PHP, maka perlu dilakukan perubahan pada *tag* HTML untuk tombol submit, yaitu dari:

```
<input type="submit" value="Klik Di Sini!">
```

menjadi perlu ditambah atribut `name` seperti berikut ini.

```
<input type="submit" name="proses" value="Klik Di Sini!">
```

Hal lain yang patut dicermati adalah pada *tag* `<form>`. Alih-alih menggunakan atribut `action="menu.php"`, kita dapat menggunakan variabel *pre-defined* PHP yaitu `$PHP_SELF` yang akan secara tepat menunjukkan bahwa skrip yang dituju adalah skrip itu sendiri. Dengan demikian, perubahan nama file `menu.php` menjadi nama yang lain, tidak menyebabkan kita perlu memodifikasi atribut `action` pada *tag* `<form>`. Bentuk dari *tag* `<form>` menjadi seperti berikut ini.

```
<form method="GET" action="<? echo $PHP_SELF; ?>">
```

Catatan Tambahan

Operator ===

Berikut ini adalah contoh penggunaan operator === yang berfungsi untuk melakukan pengecekan variabel apakah memiliki nilai dan jenis yang sama.

```
<?php

if (!$proses)

    {
    // jika nilai variabel $proses tidak ada, maka
    // tampilkan halaman pertama (form isian)
    ?>

<html>
<head>
<style type="text/css">
td {font-family: Arial;}
</style>
</head>

<body>

<form method="GET" action="<? echo $PHP_SELF; ?>">
<table cellspacing="5" cellpadding="5" border="0">

<tr>
<td align="center">
Masukkan Sesuatu!
</td>
<td align="right">
<input type="text" name="var1">
</td>
</tr>

<tr>
<td align="center">
Masukkan Yang Lainnya!
</td>
<td align="right">
<input type="text" name="var2">
</td>
</tr>
```

```
<tr>
<td colspan="2" align="center">
<input type="submit" name="proses" value="Test Variabel">
</td>
</tr>

</table>
</form>
</body>

</html>

<?php
    }
else
    {
        // jika nilai variabel $proses ada, maka lakukan
        pemrosesan
        // terhadap isian form

        if ($var1 === $var2)

            {
                $hasil = "Kedua variabel identik dan berjenis
                sama."
            }

        else
            {
                $hasil = "Kedua variabel tidak identik dan/atau
                tidak berjenis sama."
            }

    }

?>

<html>
<head>
<basefont face="Arial">
</head>

<body>
<b><? echo $hasil; ?></b>

</body>
</html>
```

```
<?php
    }
?>
```

Alternatif Penulisan

PHP juga mendukung alternatif cara penulisan (*syntax*) untuk struktur kontrol yang telah kita bicarakan. Anda dapat menuliskan kode dengan cara seperti ini.

```
<?php

if ($warp == 0)
    {
    echo "Mesin Warp Tidak Diaktifkan.";
    }
else
    {
    echo "Mesin Warp Sedang Diaktifkan.";
    }

?>
```

atau Anda dapat menuliskan seperti ini

```
<?php

if ($warp == 0):

    echo "Mesin Warp Tidak Diaktifkan.";
else:
    echo "Mesin Warp Sedang Diaktifkan.";
endif;

?>
```

Alternatif kedua sama saja dengan yang pertama, dan secara sederhana dibuat dengan mengganti tanda kurung kurawal pertama pada setiap pasangan dengan tanda colon/titik dua [:], menghapus tanda kurung kurawal kedua, dan mengakhiri seluruh blok dengan sebuah perintah "endif".

Baiklah, cukup dahulu bagian kedua dari pelajaran dasar PHP ini. Selanjutnya, kita akan belajar melakukan perulangan, sedikit tentang array, dan lebih jauh mengenai form. Jangan sampai ketinggalan!

Bagian 4: Hei, Ternyata Bisa Bekerja!

Pada artikel-artikel sebelumnya, kita telah belajar cara menggunakan skrip PHP untuk memproses informasi masukan dari pengguna atau pengunjung situs kita. Anda sangat mungkin kini sudah cukup mahir membuat `<form>` dan komponen-komponennya untuk dapat diisi oleh pengguna, termasuk juga skrip untuk memproses dan menanggapi masukan dari pengguna. Sudah cukupkah?

Sampai sekarang kita belum pernah mempelajari cara menyimpan dan mengambil kembali informasi dalam situs kita, padahal proses penyimpanan, modifikasi, penghapusan, dan pembacaan data adalah hal yang nyaris tidak terpisahkan dari pemrosesan data. Apa gunanya kita memproses data, jika kita tidak mampu mengingatnya?

Saat inilah kita membutuhkan basis data (*database*). Inilah pahlawan kita dalam mempermudah pemrosesan informasi. Mari kita mulai.

Pasangan Kita Tahun Ini

Salah satu faktor yang membuat PHP menjadi sangat populer sebagai bahasa skrip dalam pembuatan aplikasi berbasis web dan situs web dinamis adalah karena bahasa ini mendukung demikian banyak sistem basis data, mulai dari `mSQL`, `MySQL`, `MS-SQL`, `MS-Access`, `PostgreSQL`, bahkan sampai `Oracle`. Fungsi-fungsi untuk mempermudah pengaksesan berbagai jenis basis data tersebut tersedia lebih dari cukup pada PHP, sehingga meringankan, menyederhakan, serta mempercepat proses pengembangan aplikasi berbasis web.

Dari sekian banyak kombinasi PHP dan sistem basis data yang ada, terpilih pasangan `PHP/MySQL` sebagai yang terbaik dan terharmonis. Mengapa? Keduanya adalah produk-produk terbaik dari gerakan *open-source*. Mudah-mudahan Anda tidak membuka kedok bahwa Anda bukan 'anak gaul' dengan bertanya apa yang dimaksud dengan gerakan *open-source*. Ke mana saja Anda selama ini?

`MySQL` memberikan hasil yang optimal dari sisi kecepatan dan reliabilitas manajemen data. Sifatnya yang *open-source* menyebabkan `MySQL` berkembang secara pesat dan digunakan begitu banyak pengguna yang tidak ingin membuang dana begitu besar untuk sebuah sistem basis data seperti jika menggunakan sistem basis data komersial. Untuk penggunaan pada jumlah data skala medium ke bawah, `MySQL` memang pas, apalagi ditambah ketersediaan `MySQL` pada berbagai platform populer seperti `Linux`, `FreeBSD`, dan `MS Windows 9x/NT/2000`. Produk *open source* lain dalam beberapa hal lebih unggul, misalnya `PostgreSQL` yang mampu menjamin integritas data dan dapat digunakan untuk jumlah data skala besar, namun keterbatasan platform pendukungnya

sangat berpengaruh terhadap popularitasnya. Saat ini, PHP secara *built-in* telah mendukung MySQL tanpa perlu modul tambahan.

Sementara promosinya kita hentikan dan kita mulai bekerja kembali.

Siapkan Semua Peralatan

Sebelum Anda lanjutkan, perlu Anda pastikan dahulu sistem basis data MySQL telah terinstall dengan baik di sistem Anda. Jika Anda tidak menemukan tanda-tanda kehidupan dari server MySQL, Anda dapat menginstalasi MySQL dari distribusi sistem operasi Anda (jika Anda menggunakan basis Linux) atau dengan mendownload versi terbarunya dari situs resmi MySQL di <http://www.mysql.com/>. Ada perusahaan lain yang mengedarkan versi modifikasi dari MySQL dengan tambahan fitur transaksi dan jenis tabel yang berbeda yaitu NuSphere (<http://www.nusphere.com/>), namun karena masalah ini masih jadi pertentangan antara MySQL AB sebagai perilis awal kode MySQL dan dengan NuSphere, maka Anda tidak dianjurkan menggunakan produk MySQL dari NuSphere.

Anda dapat belajar melakukan sendiri instalasi MySQL dari manual yang tersedia. Mungkin dibutuhkan sedikit usaha jika Anda melakukan instalasi di sistem Linux, apalagi jika Anda melakukan instalasi bukan dari distribusi biner, tapi percayalah Anda pasti mampu melakukannya. Instalasi MySQL di sistem operasi Windows relatif lebih mudah.

Jika instalasi telah selesai dilakukan dan server MySQL telah siap, baiklah Anda siapkan bekal berikutnya. Ada baiknya Anda mengenal bahasa SQL (*Structured Query Language*) yang umum digunakan untuk berinteraksi dengan server basis data. Pengetahuan mengenai SQL akan sangat membantu karena bahasa ini akan digunakan pada hampir semua interaksi PHP dengan MySQL. Jika Anda belum mengenal SQL, tidak perlu khawatir. Tetaplah maju tak gentar karena Anda tetap akan dibantu tahap demi tahap dalam artikel ini.

Bagaimana jika Anda salah satu pembenci MySQL? Karena PHP mendukung hampir semua sistem basis data populer yang ada, Anda akan tetap dapat menggunakan dasar-dasar teknik berinteraksi dengan server basis data yang dibahas dalam artikel ini untuk diaplikasikan pada sistem basis data yang Anda sukai. Bukalah manual PHP, maka Anda akan dapati bahwa fungsi-fungsi untuk mengakses basis data manapun sesungguhnya tidak jauh berbeda. Maka, tetaplah bergabung bersama kami.

Daftar Awak USS Enterprise NCC-1701D

Bayangkan Anda adalah seorang lulusan terbaik akademi militer United Federation of Planet bidang sistem informasi. Penugasan pertama Anda adalah pada kapal perang⁴⁸

angkasa terbaik USS Enterprise NCC-1701D dipimpin oleh Captain Jean Luc Piccard yang masih selalu mencari cara untuk menumbuhkan rambut di kepalanya. Ketika briefing awal, Anda diharapkan membuat sistem informasi yang mencatat daftar awak USS Enterprise NCC-1701D. Tentu, karena First Officer William T. Riker fanatik dengan PHP dan MySQL, Anda harus membuatnya dalam PHP.

Officer Geordi LaForge menjelaskan bahwa sistem informasi ini haruslah mencakup data nama, pangkat, jabatan, lama bertugas, e-mail, dan hobi dari setiap awak. Sehingga tabel basis data yang dibuat haruslah minimal memuat data tersebut.

Sebagai seorang lulusan terbaik, Anda paham bahwa perintah dasar SQL untuk pengoperasian basis data antara lain:

- `SELECT` ; digunakan untuk menampilkan data dari sebuah tabel,
- `INSERT` ; digunakan untuk memasukkan data baru ke sebuah tabel,
- `UPDATE` ; digunakan untuk memodifikasi data yang telah ada,
- `DELETE` ; digunakan untuk menghapus data pada sebuah tabel.

Perintah dasar di atas akan kita gunakan dalam membuat sistem informasi daftar awak USS Enterprise NCC-1701D, namun sebelumnya kita harus membuat basis data dan tabelnya pada MySQL kita. Bukalah program CLI (*Command Line Interface*) MySQL dengan mengetik baris perintah berikut pada *shell* sistem operasi Anda. Pastikan perintah/program `mysql` dapat dipanggil dari CLI sistem operasi Anda.

```
mysql -u [namapengguna] -p
```

Nama pengguna adalah nama pengguna pada sistem basis data MySQL yang telah dibuat pada saat instalasi dan konfigurasi server basis data MySQL. Jika diperlukan password, maka program `mysql` akan memberikan *prompt* bagi Anda untuk memasukkan password. Jika urusan protokol ini telah selesai, maka pada layar komputer Anda akan muncul *prompt* sebagai berikut.

```
mysql>
```

Misalkan Anda ingin membuat basis data penyimpanan awak kapal dengan nama `uss_enterprise`, dan awak kapal akan disimpan dalam tabel `awak` dalam basis data tersebut, maka ketiklah sebagai berikut.

```
CREATE DATABASE uss_enterprise;  
USE uss_enterprise;
```

Baris pertama perintah di atas adalah untuk membuat basis data dengan nama `uss_enterprise` pada MySQL. Baris berikutnya adalah perintah kepada CLI MySQL untuk menggunakan basis data `uss_enterprise`. Setiap baris yang akan dieksekusi harus diakhiri dengan tanda titik-koma/semi colon (;) dan diikuti dengan menekan tombol [return] atau tombol [enter]. Dengan demikian, Anda dapat meneruskan perintah yang cukup panjang pada baris berikutnya dengan tombol [enter], dengan catatan di akhir baris tidak ada tanda semi colon. Tanda semi colon dapat pula diganti dengan frasa `\g`.

Kini Anda dapat membuat tabel untuk menyimpan daftar awak, dengan mengetik perintah berikut ini.

```
CREATE TABLE awak (  
    KODE INT(11) NOT NULL AUTO_INCREMENT,  
    NAMA VARCHAR(50) NOT NULL,  
    PANGKAT VARCHAR(50) NOT NULL,  
    JABATAN VARCHAR(50) NOT NULL,  
    BERTUGAS SMALLINT(6) NOT NULL DEFAULT 0,  
    EMAIL VARCHAR(50) NOT NULL,  
    HOBI VARCHAR(50) NOT NULL,  
    PRIMARY KEY (KODE)  
);
```

Untuk menguji apakah tabel yang Anda buat sudah terbentuk, Anda dapat mencoba mengetik perintah berikut ini, masih pada CLI `mysql`.

```
DESCRIBE awak;
```

Jika tidak ada kesalahan dalam proses pembuatan table, maka seharusnya CLI mysql akan memberikan hasil sebagai berikut.

```
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| KODE       | int(11)       |      | PRI | NULL    |      |
| auto_increment |              |      |     |         |      |
| NAMA       | varchar(50)   |      |     |         |      |
| PANGKAT    | varchar(50)   |      |     |         |      |
| JABATAN    | varchar(50)   |      |     |         |      |
| BERTUGAS   | smallint(6)  |      |     | 0       |      |
| EMAIL      | varchar(50)   |      |     |         |      |
| HOBI       | varchar(50)   |      |     |         |      |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.06 sec)
```

Sekarang kita perlu mengisi basis data ini dengan data awal. Data awal yang kita masukkan ada dua. Mengapa dua? Jawabannya sederhana saja, karena saya hanya ingin memasukkan dua data saja. Mengapa bukan tiga atau satu? Karena saya lebih suka dua.

```
INSERT INTO awak VALUES (
    Null, 'Jean', 'Captain', 'Captain', 5,
    'piccard@ncc1701d.mil.ufp', 'Archeology');

INSERT INTO awak VALUES (
```

51

```
Null,'William T. Riker','Commander','First  
Officer',5,  
'riker@ncc1701d.mil.ufp','Holodeck Game');
```

Kolom/field pertama table awak (KODE) memiliki sifat `AUTO_INCREMENT`, jadi secara otomatis akan bertambah nilainya setiap pengisian data. Hal ini akan menjamin `primary key` (kolom KODE) selalu unik. Agar MySQL otomatis mengisi nilai pada kolom KODE, maka pada saat melakukan pengisian data (`INSERT`), kolom ini diberikan nilai `Null`. Mari kita lihat apakah proses pengisian data telah sukses.

```
SELECT * FROM awak \G
```

Perintah di atas adalah perintah SQL, yang artinya "ambillah semua data dan kolom dari tabel awak". Anda dapat membatasi kolom yang diambil dengan mengganti tanda * dengan nama kolom yang akan diambil dipisahkan dengan tanda koma (,). Pembatasan jumlah data yang diambil dapat dilakukan dengan menggunakan persyaratan (`WHERE ...`) atau dengan perintah `LIMIT`. Bacalah manual MySQL, Anda akan dapati penjelasan mengenai hal ini secara lebih detail.

Jika Anda jeli, mungkin Anda bertanya mengapa digunakan frasa `\G` untuk mengakhiri perintah `SELECT` dan bukannya semi colon (;) atau `\g`? Frasa `\G` menyebabkan hasil query ditampilkan di layar secara vertikal, bukan dalam bentuk tabel baris kolom. Jika data ditampilkan dalam bentuk tabel baris kolom, hasilnya kemungkinan besar akan berantakan karena terbatasnya lebar layar CLI (80 karakter). Berikut tampilan yang dihasilkan jika Anda berhasil memasukkan data ke tabel awak.

```
***** 1. row  
*****  
      KODE: 1  
      NAMA: Jean Luc Piccard  
      PANGKAT: Captain  
      JABATAN: Captain  
      BERTUGAS: 5  
      EMAIL: piccard@ncc1701d.mil.ufp  
      HOBI: Archeology
```

```
***** 2. row
*****
      KODE: 2
      NAMA: William T. Riker
      PANGKAT: Commander
      JABATAN: First Officer
      BERTUGAS: 5
      EMAIL: riker@ncc1701d.mil.ufp
      HOBI: Holodeck Game
2 rows in set (0.05 sec)
```

Anda dapat menggunakan perintah SQL `SELECT` untuk menghitung jumlah *record*/data pada tabel awak.

```
SELECT COUNT(*) FROM awak;
```

Hasilnya adalah seperti berikut ini

```
+-----+
| count(*) |
+-----+
|         2 |
+-----+
1 row in set (0.55 sec)
```

Sampai di sini, berarti basis data, tabel, dan datanya sendiri telah siap untuk tugas Anda. Kini kita segera mulai dengan membuat skrip PHP untuk mengakses basis data MySQL. Sebelumnya ketik `quit` pada program CLI MySQL untuk keluar dari aplikasi.

Hai MySQL, Saya Datang!

Untuk pemanasan, Mari kita buat dahulu skrip PHP sederhana untuk menghitung jumlah *record*/data yang ada di tabel awak.

```
<html>
<head>
<title>Menghitung Jumlah Record Pada Tabel Awak</title>
</head>

<body>

<?php
    // set beberapa variabel untuk mengakses basis data
    MySQL.

    // nama server basis data MySQL
    $server = "localhost";

    // nama pengguna basis data
    $namauser = "test"; // misalkan user adalah 'test'

    // password pengguna basis data
    $passuser = "test"; // misalkan password adalah 'test'

    // nama basis data
    $db = "uss_enterprise";

    // membuka koneksi PHP ke basis data MySQL
    $koneksi = mysql_connect($server, $namauser,
    $passuser);

    // menentukan perintah SQL untuk query
    $query = "SELECT COUNT(*) FROM awak";

    // jalankan perintah SQL untuk query
    // pada basis data uss_enterprise pada koneksi
    // yang sudah dibuka ($koneksi)
    $hasil = mysql_db_query($db, $query, $koneksi);

    // mengambil data dari variabel $hasil
    $jml_rec = mysql_result($hasil, 0);

    // tampilkan hasilnya di halaman html

    echo "Jumlah record/data pada tabel adalah : $jml_rec";

    // bebaskan memori yang digunakan untuk proses
    //karena hasil proses telah ditampilkan
    mysql_free_result($hasil);

?>

</body>
```

```
</html>
```

Anda akan dapat hasilnya sebagai berikut.

Jumlah record/data pada tabel adalah : 2

Selamat! Skrip PHP pertama Anda untuk mengakses MySQL telah berhasil Anda buat. Mari sekarang kita bicarakan tahap demi tahap cara kerja skrip yang baru kita buat di atas.

1. Langkah pertama yang harus dikerjakan adalah memberikan informasi-informasi penting yang dibutuhkan untuk membuat koneksi ke basis data. Informasi ini meliputi: nama server tempat MySQL yang akan diakses, nama pengguna database dan passwordnya, dan nama basis data yang akan diakses. Informasi ini disimpan dalam variabel PHP.
2. Untuk dapat berkomunikasi dengan basis data, maka dibutuhkan suatu koneksi basis data ke server MySQL. Semua komunikasi akan dilewatkan pada koneksi ini. Pada PHP, koneksi ke MySQL diinisiasi dengan perintah `mysql_connect()`.

```
$koneksi = mysql_connect($server, $namauser,  
$passuser);
```

Fungsi ini memiliki 3 parameter: nama server, nama pengguna MySQL dan passwordnya. Jika server basis data MySQL dan server web secara fisik berada dan beroperasi dalam satu mesin, maka nama server umumnya cukup ditulis `localhost` atau dengan nomor IP *loopback* `127.0.0.1`.

Hasil dari fungsi ini adalah sebuah "pengenal hubungan" (*link identifier*) yang dalam skrip PHP di atas disimpan pada variabel `$koneksi`. Pengenal ini akan selalu digunakan oleh skrip untuk berkomunikasi dengan basis data.

3. Setelah kita memiliki koneksi ke basis data, maka sekaranglah saatnya mengirim perintah query dalam SQL ke basis data kita. Fungsi yang digunakan adalah `mysql_db_query()`. Fungsi ini memiliki 3 parameter pula: nama basis data, string

query dalam bahasa SQL, dan pengenalan hubungan untuk koneksi yang telah kita bicarakan sebelumnya.

```
$query = "SELECT COUNT(*) FROM awak";  
$hasil = mysql_db_query($db, $query, $koneksi);
```

Hasil dari fungsi `mysql_query()` disimpan dalam variabel `$hasil`. Isi dari variabel `$hasil` ini sangat bergantung dari perintah query SQL yang diberikan. Variabel ini dapat saja berisi satu atau lebih baris atau kolom data yang ada pada basis data.

Anda dapat mengambil dan mengekstraksi isi dari variabel hasil query dengan berbagai fungsi yang tersedia dalam PHP sesuai dengan kebutuhan Anda. Kali ini kita gunakan saja fungsi `mysql_result()` yang akan menggunakan variabel hasil query dan nomor baris serta nama kolom (*optional*/tidak perlu ada) untuk mengambil informasi dari hasil query yang kita butuhkan.

```
$jml_rec = mysql_result($hasil, 0);
```

Fungsi di atas mengandung arti ambil baris indeks 0 (baris pertama) dari hasil indeks `$hasil`, dan hasilnya disimpan dalam variabel `$jml_rec`. Fungsi-fungsi lain sebagai alternatif dari `mysql_result()` akan kita bicarakan kemudian.

4. Akhirnya, adalah hal yang bijak untuk memperhatikan bahwa hasil dari query sangat mungkin cukup menyita memory yang sangat Anda butuhkan untuk proses selanjutnya. Hal ini terutama terjadi jika perintah query ini menghasilkan baris dan kolom dalam jumlah yang relatif besar. Anda dapat membebaskan penggunaan memory oleh variabel hasil query ini dengan perintah `mysql_free_result()`, setelah semua proses yang membutuhkan variabel hasil query itu telah selesai dilaksanakan.

Perlihatkan Sesuatu Padaku!

Skrip PHP di atas merupakan contoh yang amat dasar. Sekarang kita ingin menggunakan skrip PHP untuk menampilkan data yang ada pada tabel `awak`, bukan hanya jumlahnya

saja.

Kebetulan Anda termasuk species manusia yang pemalas, sehingga sangat enggan untuk menulis berulang-ulang perintah dan variabel yang sama pada setiap skrip PHP yang Anda buat. Anda akan memisahkan beberapa variabel untuk kebutuhan koneksi basis data dan menyimpannya pada file terpisah, misalkan koneksi.inc.php. File ini akan berisi skrip sebagai berikut.

```
<?php

    // set beberapa variabel untuk mengakses basis data
    MySQL.

    // nama server basis data MySQL
    $server = "localhost";

    // nama pengguna basis data
    $namauser = "test"; // misalkan user adalah 'test'

    // password pengguna basis data
    $passuser = "test"; // misalkan password adalah 'test'

    // nama basis data
    $db = "uss_enterprise";

    // membuka koneksi PHP ke basis data MySQL
    $koneksi = mysql_connect($server, $namauser, $passuser)
        or die("Salah server, nama pengguna, atau
passwordnya!");

?>
```

File ini yang kita sisipkan pada setiap skrip PHP yang akan kita buat dengan menggunakan perintah/fungsi `include()` atau `require()`. Anda ingin mengetahui kegunaan fungsi `die()`? Fungsi ini berguna untuk menghentikan seluruh eksekusi program dan menampilkan *string* yang tertentu jika proses eksekusi perintah gagal atau ditemukan kesalahan.

Skrip PHP berikut ini akan menampilkan isi dari tabel `awak` ke dalam format halaman HTML. Untuk mempermudah tata letak, kita akan minta bantuan pada *tag* `<table>` dan rekan-rekannya.

```
<html>
<head>
<title>Menampilkan Isi Tabel Awak</title>
</head>

<body>

<?php

    // ambil data koneksi dari file koneksi.inc.php
    require("koneksi.inc.php");

    // menentukan perintah SQL untuk query
    $query = "SELECT * FROM awak";

    // jalankan perintah SQL untuk query
    $hasil = mysql_db_query($db, $query, $koneksi) or
        die("Kesalahan pada query!");

    // tampilkan hasilnya di halaman html dengan tabel
    echo "<table border=1 cellpadding=1 cellspacing=0>\n";
    echo "<tr>\n";
    echo "<td>Kode</td>\n";
    echo "<td>Nama</td>\n";
    echo "<td>Pangkat</td>\n";
    echo "<td>Jabatan</td>\n";
    echo "<td>Tugas</td>\n";
    echo "<td>Hobi</td>\n";
    echo "</tr>\n";

    // gunakan perulangan while
    // perulangan akan terjadi sepanjang masih ditemukan
record
    while ($barisdata = mysql_fetch_array($hasil))
    {
        // isikan elemen array baris ke masing-masing
variabel
        $kode = $barisdata["KODE"];
        $nama = $barisdata["NAMA"];
        $pangkat = $barisdata["PANGKAT"];
        $jabatan = $barisdata["JABATAN"];
        $bertugas = $barisdata["BERTUGAS"]." th";
        $email = "mailto:".$barisdata["EMAIL"];
        $hobi = $barisdata["HOBI"];

        // format dalam baris dan kolom tabel
        echo "<tr>\n";
        echo "<td>$kode</td>\n";
        echo "<td>";
```

```
        echo "<a href=$email>$nama</a>";
        echo "</td>\n";
        echo "<td>$pangkat</td>\n";
        echo "<td>$jabatan</td>\n";
        echo "<td>$bertugas</td>\n";
        echo "<td>$hobi</td>\n";
        echo "</tr>\n";
    }

    echo "</table>\n";

    // bebaskan memori yang digunakan untuk proses
    mysql_free_result($hasil);

?>

</body>

</html>
```

Pada contoh di atas, digunakan fungsi `mysql_fetch_array()` yang akan mengekstraksi variabel hasil query `$hasil` ke dalam variabel array `$barisdata`. Indeks komponen variabel array ini secara otomatis adalah nama kolom dari hasil query. Dengan demikian, kita dapat mengakses tiap komponen/elemen dari variabel array `$barisdata` sesuai dengan nama kolomnya, seperti pada baris-baris perintah berikut ini.

```
$kode = $barisdata["KODE"];
$nama = $barisdata["NAMA"];
$pangkat = $barisdata["PANGKAT"];
.... dan seterusnya ....
```

Perulangan yang digunakan adalah perulangan `while` yang akan terus melakukan perulangan sampai fungsi `mysql_fetch_array()` tidak memberikan hasil atau dengan kata lain sampai pernyataan `$barisdata = mysql_fetch_array($hasil)` bernilai *false*. Berikut ini adalah hasil skrip di atas.

Kode	Nama	Pangkat	Jabatan	Tugas	Hobi
1	Jean Luc Piccard	Captain	Captain	5 th	Archeology

2	William T. Riker	Commander	First Officer	5 th	Holodeck Game
---	----------------------------------	-----------	---------------	------	---------------

Cara Lain Ada Nggak, Sih?

Jangan khawatir, karena begitu banyaknya fungsi yang disediakan pada PHP, mungkin suatu saat Anda akan bingung dalam memilih cara yang akan digunakan. Kita dapat juga menggunakan fungsi `mysql_fetch_row()` untuk maksud yang sama dengan di atas, hanya saja variabel array yang dihasilkan akan berindeks angka sederhana, mulai dari 0 untuk kolom pertama sampai dengan $(n - 1)$ untuk kolom terakhir (n). Berikut ini adalah contoh penerapan untuk maksud yang sama dengan skrip di sebelumnya.

```
<html>
<head>
<title>Menampilkan Isi Tabel Awak</title>
</head>

<body>

<?php
// ambil data koneksi dari file koneksi.inc.php
require("koneksi.inc.php");

// menentukan perintah SQL untuk query
$query = "SELECT * FROM awak";

// jalankan perintah SQL untuk query
$hasil = mysql_db_query($db, $query, $koneksi) or
        die("Kesalahan pada query!");

// tampilkan hasilnya di halaman html dengan tabel
echo "<table border=1 cellpadding=1 cellspacing=0>\n";
echo "<tr>\n";
echo "<td>Kode</td>\n";
echo "<td>Nama</td>\n";
echo "<td>Pangkat</td>\n";
echo "<td>Jabatan</td>\n";
echo "<td>Tugas</td>\n";
echo "<td>Hobi</td>\n";
echo "</tr>\n";

// gunakan perulangan while
// perulangan akan terjadi sepanjang masih ditemukan
record
while ($barisdata = mysql_fetch_row($hasil))
{
```

```
variabel // isikan elemen array baris ke masing-masing
$kode = $barisdata[0];
$nama = $barisdata[1];
$pangkat = $barisdata[2];
$jabatan = $barisdata[3];
$bertugas = $barisdata[4]." th";
$email = "mailto: ".$barisdata[5];
$hobi = $barisdata[6];

// format dalam baris dan kolom tabel
echo "<tr>\n";
echo "<td>$kode</td>\n";
echo "<td>";
echo "<a href=$email>$nama</a>";
echo "</td>\n";
echo "<td>$pangkat</td>\n";
echo "<td>$jabatan</td>\n";
echo "<td>$bertugas</td>\n";
echo "<td>$hobi</td>\n";
echo "</tr>\n";
}

echo "</table>\n";

// bebaskan memori yang digunakan untuk proses
mysql_free_result($hasil);

?>

</body>
</html>
```

Huruf tebal (*bold*) pada skrip di atas menunjukkan perbedaan dengan skrip sebelumnya.

Dengan fungsi `list()`, Anda juga dapat langsung menugaskan variabel-variabel tertentu untuk menerima hasil dari fungsi `mysql_fetch_row()`. Berikut adalah contohnya, masih dalam permasalahan yang sama dengan skrip sebelumnya.

```
<html>
<head>
<title>Menampilkan Isi Tabel Awak</title>
</head>

<body>
```

```
<?php

// ambil data koneksi dari file koneksi.inc.php
require("koneksi.inc.php");

// menentukan perintah SQL untuk query
$query = "SELECT * FROM awak";

// jalankan perintah SQL untuk query
$hasil = mysql_db_query($db, $query, $koneksi) or
die("Kesalahan          pada
query!");

// tampilkan hasilnya di halaman html dengan tabel
echo "<table border=1 cellpadding=1 cellspacing=0>\n";
echo "<tr>\n";
echo "<td>Kode</td>\n";
echo "<td>Nama</td>\n";
echo "<td>Pangkat</td>\n";
echo "<td>Jabatan</td>\n";
echo "<td>Tugas</td>\n";
echo "<td>Hobi</td>\n";
echo "</tr>\n";

// gunakan perulangan while
// perulangan akan terjadi sepanjang masih ditemukan
record
while
(list($kode,$nama,$pangkat,$jabatan,$bertugas,$email,$hobi
) =
mysql_fetch_row($hasil)
{

// modifikasi beberapa variabel hasil
mysql_fetch_row()
$bertugas = $bertugas." th";
$email = "mailto: ".$email;

// format dalam baris dan kolom tabel
echo "<tr>\n";
echo "<td>$kode</td>\n";
echo "<td>";
echo "<a href=$email>$nama</a>";
echo "</td>\n";
echo "<td>$pangkat</td>\n";
echo "<td>$jabatan</td>\n";
echo "<td>$bertugas</td>\n";
echo "<td>$hobi</td>\n";
echo "</tr>\n";

}
}
```

```
echo "</table>\n";

// bebaskan memori yang digunakan untuk proses
mysql_free_result($hasil);

?>

</body>
</html>
```

Fungsi `list()` menyebabkan array hasil dari `mysql_fetch_row()` langsung diisikan pada variabel-variabel yang didaftar pada fungsi `list()` sesuai dengan urutannya.

Dari ketiga cara di atas, Anda dapat menentukan sendiri mana yang lebih Anda sukai. Jika Anda sangat konvensional, tidak praktis, bebal dan hanya ingin menggunakan fungsi `mysql_result()`, Anda bisa mengekstraksi variabel `$hasil` dengan cara sebagai berikut.

```
$kode = mysql_result($hasil,$i,"KODE");
$nama = mysql_result($hasil,$i,"NAMA");
$ pangkat = mysql_result($hasil,$i,"PANGKAT");
..... dan seterusnya .....
```

Dengan `$i` adalah indeks dari baris mulai dari 0 untuk baris pertama sampai $(n - 1)$ untuk baris ke- n . Tentu Anda harus memodifikasi perulangannya sehingga jumlah perulangan harus tepat sebanyak n kali atau tidak sama sekali jika query tidak menghasilkan satu pun baris data. Anda juga harus menugaskan bilangan pencacah `$i` yang terus bertambah agar pengaksesan bisa berlanjut ke record berikutnya. Sudahlah, pokoknya lebih rumit daripada ketiga cara di atas.

Mencari Jarum Dalam Tumpukan Jerami

Anda setuju bahwa sub judul di atas bombastis? Sama. Tetapi, biarlah atau ganti saja sub judulnya sesuai dengan keinginan Anda. Yang jelas, kita kini akan belajar membuat sebuah form untuk mencari sebuah data dan menampilkannya berdasarkan nama yang dimasukkan oleh pengguna.

Teknik menyatukan halaman form dengan proses dan hasilnya telah kita pelajari pada bagian sebelumnya, pasti dengan kecerdasan yang Anda miliki, tidak akan terlupakan begitu saja. Intinya kita akan membedakan status pencarian dengan mendeteksi adanya/nilai variabel tertentu (`$cari`) yang dikirim oleh tombol "Cari" pada form isian. Jika variabel ini bernilai, maka berarti dokumen/skrip PHP sedang dalam proses pencarian, jika tidak, tampilkan form untuk pencarian.

```
<html>
<head>
<title>Pencarian Nama</title>
<basefont face="Arial">
</head>

<body>

<?php

    // cek apakah kondisi form terkirim atau tidak
    if (!$cari)
    {

        // jika form tidak dalam kondisi terkirim,
        // tampilkan form pencarian nama

        ?>

        <center>
        <form action="<?php echo $PHP_SELF ?>"
method="POST">
        <font size=5>Program Pencarian Data Awak USS
Enterprise</font>
        <p>
        Masukkan nama awak yang dicari :
        <p>
        <input type="text" name="form_nama" size="50"
maxlength="50">
        <input type="submit" name="cari" value=" Cari Awak
">

        </form>
        </center>

        <?php

    }
    else
    {

        // jika form dalam kondisi terkirim,
        // lakukan pencarian dan tampilkan hasilnya
```

```
// ambil variabel untuk koneksi basis data
require("koneksi.inc.php");

// tentukan query dan kriteria pencarian
$query = "SELECT * FROM awak WHERE NAMA LIKE
'%" . $form_nama . "%'";

// lakukan proses query
$hasil = mysql_db_query($db, $query, $koneksi);

// cek apakah pencarian ada hasilnya
$jml_rec = mysql_num_rows($hasil);

if (!$jml_rec)
{
    // jika pencarian tidak ada hasilnya,
    // tampilkan pesan gagal

    ?>

    <center>
    <font size=5>Nama Awak tidak
ditemukan!</font><p>
    <a href="<?php echo $PHP_SELF?>">Klik di sini
    untuk kembali</a>
    </center>

    <?php
}
else
{
    // jika pencarian memberikan hasil,
    // tampilkan dalam halaman html

    // membuat tabel untuk menampilkan hasil
    pencarian
    echo "<font size=5>Hasil Pencarian ".
    "Ditemukan $jml_rec Data</font><br>\n";
    echo " <table border=1 cellpadding=1
    cellspacing=0>\n";
    echo "<tr>\n";
    echo "<td>Kode</td>\n";
    echo "<td>Nama</td>\n";
    echo "<td>Pangkat</td>\n";
    echo "<td>Jabatan</td>\n";
    echo "<td>Tugas</td>\n";
    echo "<td>Hobi</td>\n";
    echo "</tr>\n";

    // gunakan perulangan while
    // perulangan akan terjadi sepanjang masih
```

```
        // ditemukan record
        while
        (list($kode,$nama,$pangkat,$jabatan,$bertugas,
              $email,$hobi)
mysql_fetch_row($hasil))
        {

            // modifikasi beberapa variabel hasil
            // mysql_fetch_row()
            $bertugas = $bertugas." th";
            $email = "mailto:". $email;

            // format dalam baris dan kolom tabel
            echo "<tr>\n";
            echo "<td>$kode</td>\n";
            echo "<td>";
            echo "<a href=$email>$nama</a>";
            echo "</td>\n";
            echo "<td>$pangkat</td>\n";
            echo "<td>$jabatan</td>\n";
            echo "<td>$bertugas</td>\n";
            echo "<td>$hobi</td>\n";
            echo "</tr>\n";

        }

        echo "</table>\n";
        echo "<p>\n";
        echo "<a href=$PHP_SELF>Klik di sini ".
            "untuk kembali</a>\n";

        // bebaskan memori yang digunakan untuk proses
        mysql_free_result($hasil);

    }
}

?>

</body>
</html>
```

Skrip Pencarian Nama Awak USS Enterprise di atas menggunakan kriteria pencarian dengan operator LIKE yaitu "WHERE NAMA LIKE '%\$form_nama%'. Operator LIKE ini adalah operator pada bahasa SQL di MySQL yang memiliki cakupan pencarian lebih luas dari pada operator '='. Bentuk '%[string]%' akan menyebabkan pencarian dilakukan terhadap setiap data yang mengandung '[string]'. Operator LIKE ini juga tidak

membedakan huruf kapital dan huruf kecil. Berikut ilustrasinya.

Isi Kotak Teks Pada Form	Nama Awak yang didapat dari Pencarian
piccard	Jean Luc Piccard
PiCcArđ	Jean Luc Piccard
pic	Jean Luc Piccard
cp	-
c p	Jean Luc Piccard
a	Jean Luc Piccard, William T. Riker
card	Jean Luc Piccard
ill	William T.Riker

Fungsi baru yang Anda jumpai pada skrip di atas adalah `mysql_num_rows()` yang akan memberikan hasil jumlah baris/data yang dihasilkan dari proses query. Hasil dari fungsi ini disimpan pada variabel `$jml_rec` yang akan digunakan untuk menentukan apakah data yang dicari ditemukan atau tidak. Jika variabel `$jml_rec = 0` berarti data tidak ditemukan, dan jika lebih dari nol, maka seluruh hasil akan ditampilkan dalam tabel pada halaman HTML.

Cobalah jalankan skrip PHP di atas, lama-kelamaan Anda akan mengerti dasar-dasar alur kerja skrip untuk pencarian data. Skrip ini dapat dikembangkan lebih jauh sesuai dengan kebutuhan Anda.

Awak Kapal Baru

Sampai saat ini, Anda telah berulang kali menggunakan perintah SQL `SELECT` yang berguna untuk mengambil informasi dari basis data yang ada. Kini saatnya kita membuat skrip untuk mengisi basis data kita dengan awak-awak kapal USS Enterprise yang lainnya. Yang jelas, kita tidak mungkin memasukkan satu per satu data awak tersebut melalui Aplikasi CLI MySQL yang sangat membosankan dan tidak menarik itu. Kita ingin data dimasukkan melalui halaman HTML yang dibuat dengan skrip PHP kita. Caranya?

Untuk mengisi data baru ke dalam tabel basis data, kita menggunakan perintah SQL

lainnya, yaitu INSERT. Cobalah skrip PHP berikut ini. Agar pada langkah-langkah berikutnya Anda tidak perlu mengganti nama skrip ini, simpanlah dengan nama `awakinput.php`.

```
<html>
<head>
<title>Memasukkan Awak Baru</title>
<basefont face="Arial">
</head>

<body>

<?php

    // cek apakah kondisi form terkirim atau tidak
    if (!$tambah)
    {
        // jika form tidak dalam kondisi terkirim,
        // tampilkan form pencarian nama

        ?>

        <center>
        <form   action="<?php   echo   $PHP_SELF   ?>"
method="POST">
        <font size=5>Masukkan Data Awak USS Enterprise
Baru</font>
        <p>
        <table border=0 cellspacing=2 cellpadding=2>

            <tr>
            <td>Nama Awak</td>
            <td>
            <input type="text" name="form_nama" size="50"
maxlength="50">
            </td>
            </tr>

            <tr>
            <td>Pangkat</td>
            <td>
            <input type="text" name="form_pangkat" size="50"
maxlength="50">
            </td>
            </tr>
```

```
<tr>
<td>Jabatan</td>
<td>
<input type="text" name="form_jabatan" size="50"
maxlength="50">
</td>
</tr>

<tr>
<td>Lama Bertugas</td>
<td>
<input type="text" name="form_bertugas" size="2"
maxlength="2"> (dalam tahun)
</td>
</tr>

<tr>
<td>e-mail</td>
<td>
<input type="text" name="form_email" size="50"
maxlength="50">
</td>
</tr>

<tr>
<td>Hobi</td>
<td>
<input type="text" name="form_hobi" size="50"
maxlength="50">
</td>
</tr>

<tr>
<td colspan=2 align=center>
<input type="submit" name="tambah" value=" Tambah
">
</td>
</tr>

</table>
</form>
</center>

<?php
}
else
{

// jika form dalam kondisi terkirim,
// lakukan insert ke basis data
```

```
// ambil variabel untuk koneksi basis data
require("koneksi.inc.php");

// tentukan query dan kriteria pencarian
$query = "INSERT INTO awak VALUES (
    Null,
    '".addslashes($form_nama)."',
    '".addslashes($form_pangkat)."',
    '".addslashes($form_jabatan)."',
    $form_bertugas,
    '".addslashes($form_email)."',
    '".addslashes($form_hobi)."'
)";

// lakukan proses query
$hasil = mysql_db_query($db,$query,$koneksi)
        or die('Kesalahan pada proses
query!');

// Tampilkan pesan proses input telah selesai
?>

<center>
<font size=5>Proses Input Berhasil!</font><p>
Data Awak Nama
<b>
<?php echo addslashes($form_nama) ?>
</b>
telah disimpan.

<p>
<a href="<?php echo $PHP_SELF ?>">Klik di sini
untuk kembali</a>
</center>

<?php
}

?>

</body>
</html>
```

Anda mungkin belum mengenal kegunaan fungsi `addslashes()`. Fungsi ini berguna untuk memastikan bahwa data string yang dikirim ke server MySQL telah bebas dari karakter-karakter terlarang, seperti `'`, `"`, `\`, dan sebagainya, sehingga proses query

terjamin dari kegagalan.

Untuk mencoba skrip ini, masukkanlah data misalnya:

```
NAMA: Deanne Troi
PANGKAT: Commander
JABATAN: Counselor
BERTUGAS: 2
EMAIL: troi@ncc1701d.mil.ufp
HOBI: Fine Art
```

Untuk melihat apakah proses input yang kita lakukan berhasil, maka gunakan skrip PHP yang kedua, ketiga, atau keempat dari artikel ini. Hasilnya kurang lebih tampak sebagai berikut.

Kode	Nama	Pangkat	Jabatan	Tugas	Hobi
1	Jean luc Piccard	Captain	Captain	5 th	Archeology
2	William T. Riker	Commander	First Officer	5 th	Holodeck Game
3	Deanne Troi	Commander	Counselor	2 th	Fine Art

Selain menggunakan fungsi `mysql_db_query()`, Anda dapat juga menggunakan fungsi `mysql_query()` yang lebih praktis untuk digunakan berulang-ulang pada basis data yang sama. Sebelum fungsi ini, terlebih dahulu haruslah didefinisikan dahulu basis data yang digunakan dengan fungsi `mysql_select_db()`. Sehingga perintah

```
mysql_db_query($db, $query, $koneksi);
```

dapat diganti dengan

```
mysql_select_db($db, $koneksi);
mysql_query($query);
```

Bentuk yang kedua ini jauh lebih praktis jika kita secara berulang-ulang melakukan proses query pada basis data yang sama. Fungsi `mysql_select_db()` cukup dilakukan sekali saja diawal skrip ataupun diletakkan pada skrip koneksi.inc.php, dan selanjutnya cukup dengan perintah `mysql_query()` saja.

Data Salah, Tolong Diubah!

Tiba-tiba timbul masalah, Anda salah memasukkan data. Anda harus segera membuat skrip lain untuk mengubah data yang sudah dimasukkan, sebelum data ini diakses oleh para pengguna lainnya. Ayo cepat, kita diburu waktu!

Berbeda dengan menambahkan data baru, proses perbaikan (*edit*) data tidak berjalan dalam dua langkah: isi dan simpan. Sebelum kita memperbaiki data, kita harus memilih dahulu data yang akan diubah, data asli sebelum diubah ditampilkan, ubah data sesuai keinginan, kemudian simpan perubahannya. Untuk mudahnya, skrip mengubah data kita bagi menjadi 2 buah skrip, yang pertama adalah untuk mencari dan memilih data yang akan diubah dan skrip yang kedua untuk melakukan perubahan dan menyimpan perubahannya.

Skrip yang pertama ini akan kita modifikasi dari skrip keempat pada artikel ini, yang bertugas menampilkan semua data yang ada pada tabel awak. Berikut ini adalah skrip yang telah dimodifikasi. Tambahkan dan modifikasi skrip tampak pada bagian yang berhuruf tebal. Simpanlah hasil perubahan ini dengan nama `awakdsp.php`.

```
<html>
<head>
<title>Menampilkan Isi Tabel Awak</title>
<bfont face="Arial">
</head>

<body>

<?php

// ambil data koneksi dari file koneksi.inc.php
require("koneksi.inc.php");
```

```
// menentukan perintah SQL untuk query
$query = "SELECT * FROM awak";

// jalankan perintah SQL untuk query
$hasil = mysql_db_query($db, $query, $koneksi) or
        die("Kesalahan pada query!");

// tampilkan hasilnya di halaman html dengan tabel
echo "<font size=5>Data Awak USS Enterprise
NCC-1701-D</font>\n";
echo "<table border=1 cellpadding=1 cellspacing=0>\n";
echo "<tr>\n";
echo "<td>Kode</td>\n";
echo "<td>Nama</td>\n";
echo "<td>Pangkat</td>\n";
echo "<td>Jabatan</td>\n";
echo "<td>Tugas</td>\n";
echo "<td>Hobi</td>\n";
echo "<td>Pilihan</td>\n";
echo "</tr>\n";

// gunakan perulangan while
// perulangan akan terjadi sepanjang masih ditemukan
record
while
(list($kode,$nama,$pangkat,$jabatan,$bertugas,$email,$hobi
) =
        mysql_fetch_row($hasil))
{
    // modifikasi beberapa variabel hasil
mysql_fetch_row()
    $bertugas = $bertugas." th";
    $email = "mailto:". $email;

    // format dalam baris dan kolom tabel
echo "<tr>\n";
echo "<td>$kode</td>\n";
echo "<td>";
echo "<a href=$email>$nama</a>";
echo "</td>\n";
echo "<td>$pangkat</td>\n";
echo "<td>$jabatan</td>\n";
echo "<td>$bertugas</td>\n";
echo "<td>$hobi</td>\n";
echo "<td>";
echo "                                "<a
href=\ "awakedit.php?kodeawak=$kode\ ">Edit</a>";
echo "</td>\n";
echo "</tr>\n";
}
}
```

```
        echo "</table>\n";

        // bebaskan memori yang digunakan untuk proses
        mysql_free_result($hasil);

    ?>

</body>
</html>
```

Jalankan skrip PHP ini di browser Anda, dan saksikan perubahannya. Pada setiap baris data pada tabel, tersedia *hyperlink* untuk edit/ubah baris data yang bersangkutan di kolom paling kanan (kolom Pilihan). *Hyperlink* ini akan memanggil skrip `awakedit.php` yang akan kita buat berikut ini dengan sekaligus mengirim nilai variabel `$kodeawak` sesuai dengan kode dari data yang akan diedit.

Berikut ini adalah skrip untuk mengedit data yang akan disimpan dengan nama `awakedit.php`.

```
<html>
<head>
<title>Mengubah Data Awak</title>
<basefont face="Arial">
</head>

<body>

<?php
    // ambil variabel untuk koneksi basis data
    require("koneksi.inc.php");
    mysql_select_db($db, $koneksi);

    // cek apakah kondisi form terkirim atau tidak
    if (!$simpan)
    {

        // jika form tidak dalam kondisi terkirim,
        // tampilkan form pencarian nama
        // cek apakah variabel $kode dikirimkan
        if (!$kodeawak) {
            die('Tidak ada awak yang dipilih untuk
diedit!'); }

        // Tentukan query untuk ada yang akan diambil
```

```
$query = "SELECT * FROM awak WHERE
KODE= '$kodeawak'";

// jalankan query
$hasil = mysql_query($query) or
die('Kesalahan pada proses query!');

// cek dan ekstrak hasil query
$jml_rec = mysql_num_rows($hasil);
if (!( $jml_rec > 0 )) { die('Data tidak ditemukan!');
}

list($kode,$nama,$pangkat,$jabatan,$bertugas,$email,$hobi)
=
mysql_fetch_row($hasil);

?>

<center>
<form action="<?php echo $PHP_SELF ?>"
method="POST">
<font size=5>Edit Data Awak USS Enterprise</font>
<p>
<table border=0 cellspacing=2 cellpadding=2>

<tr>
<td>Nama Awak</td>
<td>
<input type="hidden" name="form_kode"
value="<?php echo $kode ?>">
<input type="text" name="form_nama" size="50"
value="<?php echo $nama ?>" maxlength="50">
</td>
</tr>

<tr>
<td>Pangkat</td>
<td>
<input type="text" name="form_pangkat" size="50"
value="<?php echo $pangkat ?>" maxlength="50">
</td>
</tr>

<tr>
<td>Jabatan</td>
<td>
<input type="text" name="form_jabatan" size="50"
value="<?php echo $jabatan ?>" maxlength="50">
</td>
</tr>
```

```
<tr>
<td>Lama Bertugas</td>
<td>
<input type="text" name="form_bertugas" size="2"
value="<?php echo $bertugas ?>" maxlength="2">
(dalam tahun)
</td>
</tr>

<tr>
<td>e-mail</td>
<td>
<input type="text" name="form_email" size="50"
value="<?php echo $email ?>" maxlength="50">
</td>
</tr>

<tr>
<td>Hobi</td>
<td>
<input type="text" name="form_hobi" size="50"
value="<?php echo $hobi ?>" maxlength="50">
</td>
</tr>

<tr>
<td colspan=2 align=center>
<input type="submit" name="simpan" value=" Simpan
">
</td>
</tr>

</table>

</form>
</center>

<?php

// bebaskan memori yang digunakan untuk proses
mysql_free_result($hasil);

}
else
{

// jika form dalam kondisi terkirim,
// lakukan perubahan basis data

// tentukan query
$query = "UPDATE awak SET
```

```
        NAMA='".addslashes($form_nama)."',
PANGKAT='".addslashes($form_pangkat)."',
JABATAN='".addslashes($form_jabatan)."',
        BERTUGAS=$form_bertugas,
EMAIL='".addslashes($form_email)."',
        HOBI='".addslashes($form_hobi)."'
        WHERE KODE='$form_kode';

// lakukan proses query
$hasil = mysql_query($query)
        or die('Kesalahan pada proses
query!');

// Tampilkan pesan proses edit telah selesai
?>

<center>
<font size=5>Proses Edit Berhasil!</font><p>
Data Awak Nama
<b><?php echo addslashes($form_nama) ?></b>
telah disimpan perubahannya.
<p>
<a href="awakdsp.php">Klik di sini untuk
kembali</a>
</center>

<?php
}
?>

</body>
</html>
```

Tidak ada yang baru pada skrip PHP di atas kecuali penggunaan perintah SQL UPDATE untuk mengubah data yang sudah ada. Kolom KODE tidak ikut diubah karena kolom ini adalah PRIMARY KEY sebagai acuan dari perubahan data yang dilakukan. Data KODE ini disimpan pada form sebagai jenis komponen form hidden, agar tetap dapat dikirimkan ke bagian skrip untuk menyimpan perubahan.

Cobalah ubah masa tugas Deanne Troi dari 2 tahun menjadi 3 tahun. Jika skrip yang Anda buat benar, maka hasil perubahan akan langsung terlihat.

Kalau Gajah Mati Meninggalkan Apa?

Masalah berikutnya, jika seorang awak sudah tidak lagi terdaftar sebagai awak USS ENTERPRISE baik karena perpindahan tugas maupun karena gugur, maka data awak tersebut harus dienyahkan dari sistem basis data kita. Anda perlu membuat skrip untuk menghapus data.

Indentik dengan proses edit, proses penghapusan juga harus melalui tahap pemilihan data yang akan dihapus, kemudian data yang akan dihapus ditampilkan, dan terakhir dilakukan proses penghapusan. Untuk itu kita gunakan dua skrip juga, satu skrip adalah modifikasi dari skrip awakdsp.php dan satu lagi skrip yang bertugas menampilkan data dan melakukan proses penghapusan data.

Langkah pertama, modifikasi skrip awakdsp.php dengan menambahkan *hyperlink* untuk penghapusan pada setiap baris data yang ada. Berikut adalah skrip awakdsp.php yang telah dimodifikasi. Penambahan yang terjadi adalah pada yang berhuruf tebal.

```
<html>
<head>
<title>Menampilkan Isi Tabel Awak</title>
<basefont face="Arial">
</head>

<body>

<?php

// ambil data koneksi dari file koneksi.inc.php
require("koneksi.inc.php");

// menentukan perintah SQL untuk query
$query = "SELECT * FROM awak";

// jalankan perintah SQL untuk query
$hasil = mysql_db_query($db, $query, $koneksi) or
die("Kesalahan          pada
query!");

// tampilkan hasilnya di halaman html dengan tabel
echo "<font size=5>Data Awak USS Enterprise
NCC-1701-D</font>\n";
echo "<table border=1 cellpadding=1 cellspacing=0>\n";
echo "<tr>\n";
echo "<td>Kode</td>\n";
echo "<td>Nama</td>\n";
```

```
        echo "<td>Pangkat</td>\n";
        echo "<td>Jabatan</td>\n";
        echo "<td>Tugas</td>\n";
        echo "<td>Hobi</td>\n";
        echo "<td>Pilihan</td>\n";
        echo "</tr>\n";

        // gunakan perulangan while
        // perulangan akan terjadi sepanjang masih ditemukan
record
    while
    (list($kode,$nama,$pangkat,$jabatan,$bertugas,$email,$hobi
) =
        mysql_fetch_row($hasil))
    {
        // modifikasi beberapa variabel hasil
mysql_fetch_row()
        $bertugas = $bertugas." th";
        $email = "mailto: ".$email;

        // format dalam baris dan kolom tabel
        echo "<tr>\n";
        echo "<td>$kode</td>\n";
        echo "<td>";
        echo "<a href=$email>$nama</a>";
        echo "</td>\n";
        echo "<td>$pangkat</td>\n";
        echo "<td>$jabatan</td>\n";
        echo "<td>$bertugas</td>\n";
        echo "<td>$hobi</td>\n";
        echo "<td>";
        echo "                                "<a
href=\ "awakedit.php?kodeawak=$kode\ ">Edit</a>";
        echo " ";
        echo "                                "<a
href=\ "awakdel.php?kodeawak=$kode\ ">Hapus</a>";
        echo "</td>\n";
        echo "</tr>\n";
    }

    echo "</table>\n";

    // bebaskan memori yang digunakan untuk proses
    mysql_free_result($hasil);

?>

</body>
</html>
```

Langkah berikutnya adalah membuat skrip awakdel.php sebagai berikut.

```
<html>
<head>
<title>Menghapus Data Awak</title>
<basefont face="Arial">
</head>

<body>

<?php

    // ambil variabel untuk koneksi basis data
    require("koneksi.inc.php");

    mysql_select_db($db, $koneksi);

    // cek apakah kondisi form terkirim atau tidak
    if (!$hapus)
    {
        // jika form tidak dalam kondisi terkirim,
        // tampilkan form pencarian nama

        // cek apakah variabel $kode dikirimkan
        if (!$kodeawak) {
            die('Tidak ada awak yang dipilih untuk
dihapus!'); }

        // Tentukan query untuk ada yang akan diambil
        $query = "SELECT * FROM awak WHERE
KODE=' $kodeawak' ";

        // jalankan query
        $hasil = mysql_query($query) or
            die('Kesalahan pada proses query!');

        // cek dan ekstrak hasil query
        $jml_rec = mysql_num_rows($hasil);
        if (!($jml_rec>0)) { die('Data tidak ditemukan!'); }
    }

    list($kode,$nama,$pangkat,$jabatan,$bertugas,$email,$hobi)
    =
        mysql_fetch_row($hasil);
```

```
?>

<center>
<form action="<?php echo $PHP_SELF ?>"
method="POST">
<font size=5>Edit Data Awak USS Enterprise</font>
<p>
<table border=0 cellspacing=2 cellpadding=2>

<tr>
<td>Nama Awak</td>
<td>
<input type="hidden" name="form_kode"
value="<?php echo $kode ?>">
<?php echo $nama ?>
</td>
</tr>

<tr>
<td>Pangkat</td>
<td>
<?php echo $pangkat ?>
</td>
</tr>

<tr>
<td>Jabatan</td>
<td>
<?php echo $jabatan ?>
</td>
</tr>

<tr>
<td>Lama Bertugas</td>
<td>
<?php echo $bertugas ?> tahun
</td>
</tr>

<tr>
<td>e-mail</td>
<td>
<?php echo $email ?>
</td>
</tr>

<tr>
<td>Hobi</td>
<td>
<?php echo $hobi ?>
</td>
</tr>
</tr>
```

```

        <tr>
        <td colspan=2 align=center>
        <input type="submit" name="hapus" value=" Hapus
Data ">
        </td>
        </tr>

        </table>
        </form>
        </center>

        <?php

        // bebaskan memori yang digunakan untuk proses
        mysql_free_result($hasil);

        }
        else
        {

        // jika form dalam kondisi terkirim,
        // lakukan penghapusan data

        // tentukan query
        $query = "DELETE FROM awak
                WHERE KODE='$form_kode'";

        // lakukan proses query
        $hasil = mysql_query($query)
                or die('Kesalahan pada proses
query!');

        // Tampilkan pesan proses hapus telah selesai

        ?>

        <center>
        <font size=5>Proses Hapus Berhasil!</font><p>
        Data Awak Nama
        <b><?php echo addslashes($form_nama) ?></b>
        telah dihapus.

        <p>

        <a href="awakdsp.php">Klik di sini untuk
        kembali</a>

        </center>

```

```
<?php  
  
    }  
  
?>  
  
</body>  
</html>
```

Cobalah sekarang hapus Deanne Troi mulai dari menjalankan file skrip `awakdsp.php` dan memilih *hyperlink* Hapus. Jika skrip Anda benar dalam pembuatannya, maka proses penghapusan akan berjalan dengan baik. Maka dengan ini tugas Anda membuat basis data awak USS Enterprise telah diselesaikan dengan gemilang. Simpan semua dokumen dan skrip Anda dan jangan lupa sesegera mungkin menghubungi Counselor Deanne Troi untuk minta ditraktir dalam rangka pindah tugasnya.

Butuh Bantuan Tambahan Dari PHP?

Puhh! Anda kini telah menguasai dengan baik cara mengakses basis data MySQL dari PHP. Dengan dasar ini, maka Anda sudah dapat mulai membuat web dinamis yang berbasis data. Namun, Anda merasa sering kali mengalami kesulitan untuk melacak kesalahan pemrograman yang tidak sengaja Anda lakukan? Umumnya kesalahan terjadi pada saat merancang perintah SQL, tetapi mencari di bagian mana dari perintah SQL yang salah itu juga tidak mudah dan butuh waktu. Adakah cara yang lebih cepat untuk membantu pekerjaan mencari kutu ini?

PHP, kembali lagi kita harus berterimakasih kepadanya, menyediakan banyak sekali alat bantu pelacakan kesalahan (*error tracking*). Untuk melacak kesalahan pada proses pengaksesan MySQL, PHP menyediakan fungsi-fungsi `mysql_errno()` yang menunjukkan nomor indeks dari kesalahan yang terjadi, dan `mysql_error()` yang memberikan keterangan kepada kita mengenai kesalahan apa yang terjadi. Berikut ini adalah contoh penggunaannya.

```
<html>
<head>
<title>Melacak Kesalahan SQL</title>
</head>

<body>

<?php
// ambil data koneksi dari file koneksi.inc.php
require("koneksi.inc.php");

// menentukan perintah SQL untuk query
$query = "SELECT FROM awak";

// jalankan perintah SQL untuk query
$hasil = mysql_db_query($db, $query, $koneksi);

if (!$hasil)
{

    $no_error = mysql_errno();
    $pesan_error = mysql_error();

    echo "Kesalahan MySQL No $no_error : $pesan_error";

}

?>

</body>
</html>
```

Jika skrip di atas dijalankan, maka akan muncul pesan kesalahan sebagai berikut.

```
Kesalahan MySQL No 1064 : You have an error in your SQL syntax near
'FROM awak'
at line 1
```

Akhirnya, selesai juga bagian artikel yang paling melelahkan ini. Pada bagian terakhir dari PHP? Siapa Takut! akan kita bicarakan mengenai kemampuan PHP untuk membaca dan menulis file serta mencoba merancang fungsi-fungsi buatan sendiri. Siapkan Anda untuk satu trayek lagi...

Bagian 5: Sebuah Perhentian Sementara

Membangkitkan Kenangan

Jika Anda masih juga membaca artikel bagian kelima ini, artinya Anda benar-benar tangguh dan berkemauan kuat. Perjalanan yang cukup panjang dan melelahkan pada bagian-bagian sebelumnya telah cukup banyak memberikan bekal bagi Anda untuk membangun sendiri situs-situs dinamis dengan skrip PHP.

Pada bagian keempat, Anda telah melihat betapa PHP dapat digunakan untuk mengakses basis data MySQL. PHP dapat menyimpan, mencari, memperbaiki, dan menghapus informasi yang tersimpan pada basis data MySQL dengan sangat mudah dan sederhana. Anda ternyata tidak perlu sering mengkerutkan kening Anda. Memang inilah salah satu kekuatan PHP, kemudahan koneksi dengan basis data, bukan hanya dengan basis data MySQL, tapi juga dengan banyak basis data populer lainnya.

Dalam kenyataannya, kita sering mengalami bahwa data tidak tersimpan secara rapi dalam baris dan kolom basis data, dan seringkali juga Anda harus menyimpan suatu informasi dalam model file teks ASCII. Problem utama menjadi muncul, karena ternyata Anda belum mengetahui caranya.

Ada tiga pilihan untuk menghadapi situasi ini. Pilihan pertama Anda dapat menyuruh seseorang untuk memasukkan data-data tersebut ke dalam basis data. Pilihan kedua Anda dapat mengangkat tangan sebahu dan kemudian menghela nafas panjang sembari memutuskan untuk keluar dari pekerjaan Anda sekarang. Pilihan yang lain, Anda dapat menggunakan PHP untuk menyelesaikan problem Anda.

Karena Anda telah menunjukkan ketangguhan Anda untuk belajar PHP, saya yakin pilihan ketiga yang akan Anda ambil, dan inilah isi dari bagian kelima artikel PHP? Siapa Takut!

Pada bagian akhir artikel terakhir ini, kita juga akan mempelajari mengenai pembuatan fungsi-fungsi pada skrip PHP, yang akan membantu kemalasan kita untuk menulis bagian skrip berulang-ulang. Kencangkan sabuk pengaman Anda, dan kita masuki putaran final perjalanan kita.

Ditemukan Manuskrip Tua

Seperti juga bahasa pemrograman yang lain, PHP juga dilengkapi dengan fungsi-fungsi yang lengkap untuk menulis dan membaca file dengan mudah. Mari kita buat dulu sebuah file teks yang akan kita beri nama `manuskrip.txt` sebagai berikut.

Beribu abad yang lalu, kami adalah ras cerdas yang hidup dan berkelana ke seluruh penjuru jagad raya ini. Ketika kami sampai pada masa di mana ras kami tidak mampu mempertahankan populasi dan menuju kepunahan, kami menciptakan sebuah program protein yang membawa kode genetik dasar ras kami. Protein ini mampu berevolusi selama ribuan abad sampai akhirnya akan membentuk ras cerdas serupa kami. Kami sebarkan program ini pada sistem-sistem planet yang mampu mendukung evolusi program kami.

Jika kalian menemukan manuskrip ini, maka itulah tanda bahwa program protein kami telah berhasil membentuk kalian menjadi ras cerdas yang menguasai jagad raya ini, seperti harapan kami.

Sebelum Anda dapat menggunakan PHP untuk membaca isi dari file ini, Anda harus memastikan bahwa Anda memiliki *permission* untuk membacanya. Pada sistem berbasis Unix/Linux, hal ini dapat diyakinkan dengan perintah:

```
$ chmod 744 manuskrip.txt
```

Kini kita buat skrip PHP sederhana untuk membaca file tersebut dan menunjukkan isi serta ukuran file:

```
<?php
    // baca file
    $ukuran_file = readfile("manuskrip.txt");

    // tampilkan ukuran file
    echo "<br>Ukuran File = $ukuran_file byte.";
?>
```

Maka hasil tampilannya kurang lebih seperti ini.

Beribu abad yang lalu, kami adalah ras cerdas yang hidup dan berkelana ke seluruh penjuru jagad raya ini. Ketika kami sampai pada masa di mana ras kami tidak mampu mempertahankan populasi dan menuju kepunahan, kami menciptakan sebuah program protein yang membawa kode genetik dasar ras kami. Protein ini mampu berevolusi selama ribuan abad sampai akhirnya akan membentuk ras cerdas serupa kami. Kami sebar program ini pada sistem-sistem planet yang mampu mendukung evolusi program kami. Jika kalian menemukan manuskrip ini, maka itulah tanda bahwa program protein kami telah berhasil membentuk kalian menjadi ras cerdas yang menguasai jagad raya ini, seperti harapan kami.

Ukuran File = 681 byte.

Fungsi `readfile()` ternyata hanya sederhana tugasnya; membaca file dan menampilkan isinya. Fungsi ini juga menghasilkan ukuran file yang kita tampilkan pada skrip PHP di atas melalui perintah `echo`.

Begitu sederhananya, sehingga mungkin kita tidak akan begitu sering menggunakan fungsi `readfile()`. Kita membutuhkan fungsi yang lebih berguna lagi, yang selain melakukan apa yang dapat dilakukan fungsi `readfile()`, dapat digunakan untuk memformat data yang dibaca. Cobalah gunakan fungsi `file()` yang akan menghasilkan pembacaan file dalam bentuk variabel *array* PHP. Setiap elemen pada variabel *array* tersebut mewakili satu baris dalam file yang dibaca, dan jumlah elemen variabel *array* yang ada akan menunjukkan jumlah baris dalam file. Skrip PHP berikut ini akan lebih memberi kejelasan.

```
<?php

// tentukan nama file yang akan dibaca
$nama_file = "manuskrip.txt";

// baca file, masukkan ke array
$isi = file($nama_file);

// cari jumlah baris
$jumlah_baris = sizeof($isi);

echo "File $nama_file berisi $jumlah_baris baris.<p>";

// tampilkan hasil pembacaan baris per baris dengan
perulangan for
for ($i=0; $i<$jumlah_baris; $i++)
{
```

87

```
        echo "Baris " . ($i+1) . " : <br>";  
        echo $isi[$i] . "<br>";  
    }  
  
?>
```

Pada skrip PHP di atas kita menggunakan fungsi `file()` untuk membaca isi file yang ditunjuk oleh variabel `$nama_file` dan baris demi baris isi file tersebut disimpan dalam variabel `array $isi`. Jumlah elemen dari variabel `$isi` didapatkan dengan menggunakan fungsi `sizeof()`, dan hasilnya disimpan dalam variabel `$jumlah_baris`. Baris demi baris isi dari file yang tersimpan pada elemen-elemen variabel `array $isi` ditampilkan dengan menggunakan perulangan `for`.

Hasil dari skrip PHP di atas adalah sebagai berikut.

File manuskrip.txt berisi 3 baris.

Baris 1 :

Beribu abad yang lalu, kami adalah ras cerdas yang hidup dan berkelana ke seluruh penjuru jagad raya ini. Ketika kami sampai pada masa di mana ras kami tidak mampu mempertahankan populasi dan menuju kepunahan, kami menciptakan sebuah program protein yang membawa kode genetik dasar ras kami. Protein ini mampu berevolusi selama ribuan abad sampai akhirnya akan membentuk ras cerdas serupa kami. Kami sebar program ini pada sistem-sistem planet yang mampu mendukung evolusi program kami.

Baris 2 :

Baris 3 :

Jika kalian menemukan manuskrip ini, maka itulah tanda bahwa program protein kami telah berhasil membentuk kalian menjadi ras cerdas yang menguasai jagad raya ini, seperti harapan kami.

Hilang Tak Berbekas

Skrip PHP di atas, semuanya dibuat dengan asumsi dasar bahwa file yang berusaha dibaca oleh skrip PHP telah tersedia. Dalam kenyataannya bisa saja file yang dimaksud tidak belum ada. Kondisi ini dapat menyebabkan waktu Anda terbuang untuk mencari kesalahan program. Anda dapat menghindari kejadian semacam ini dengan fungsi `file_exists()` yang dapat digunakan untuk menguji keberadaan file dan menampilkan pesan kesalahan yang sesuai jika file tersebut tidak ditemukan. Anda dapat_{gg}

dimodifikasi skrip PHP sebelumnya, sehingga menjadi seperti berikut ini (perubahan ditunjukkan dengan huruf tebal).

```
<?php

    // tentukan nama file yang akan dibaca
    $nama_file = "manuskrip.txt";

    // cek keberadaan file
    if (file_exists($nama_file))
    {
        // baca file, masukkan ke array
        $sisi = file($nama_file);

        // cari jumlah baris
        $jumlah_baris = sizeof($sisi);

        echo "File $nama_file berisi $jumlah_baris
baris.<p>";

        // tampilkan hasil pembacaan baris per baris dengan
        perulangan for
        for ($i=0; $i<$jumlah_baris; $i++)
        {
            echo "Baris " . ($i+1) . " : <br>";
            echo $sisi[$i]. "<br>";
        }
    }
    else
    {

        echo "<b>File tidak ditemukan!</b>";

    }

?>
```

Jalankan program di atas dengan sebelumnya mengganti nama file `manuskrip.txt` atau dengan menghapus file tersebut. Maka pesan kesalahan **File tidak ditemukan!** akan muncul pada browser Anda.

PHP juga menawarkan banyak fungsi yang dapat memberikan informasi lebih detail

mengenai keadaan sebuah file, berikut ini di antaranya:

- `is_dir()`
Untuk mengetahui apakah file yang ditunjuk merupakan direktori atau bukan.
- `is_link()`
Untuk mengetahui apakah file yang ditunjuk merupakan file *link*.
- `is_executable()`
Untuk mengetahui apakah file yang ditunjuk dapat dieksekusi dari *shell*.
- `is_readable()`
Untuk mengetahui apakah file yang ditunjuk merupakan file yang dapat dibaca.
- `is_writable()`
Untuk mengetahui apakah file yang ditunjuk merupakan file yang dapat ditulis.
- `fileowner()`
Untuk mendapatkan id pengguna pemilik file.
- `filegroup()`
Untuk mendapatkan id group pengguna pemilik file.
- `fileperms()`
Untuk mendapatkan jenis *permission* dari suatu file.
- `filesize()`
Untuk mendapatkan ukuran dari suatu file.
- `filetype()`
Untuk mendapatkan jenis dari suatu file.

Fungsi-fungsi di atas sebagian besar hanya berarti pada implementasi PHP di sistem file berbasis Unix/Linux. Anda tidak akan mendapatkan hasil yang akurat jika fungsi-fungsi tersebut dijalankan pada sistem operasi lain, misalnya MS Windows 9x atau NT/2000/XP.

Anda dapat mencoba skrip berikut ini pada sistem Unix/Linux Anda. Hasilnya mungkin akan sangat berbeda jika Anda jalankan skrip ini pada sistem operasi yang lain

```
<?php
// tentukan nama file yang akan dibaca
$nama_file = "manuskrip.txt";

// cek keberadaan file
if (file_exists($nama_file))
{
    echo "Nama File : <b>$nama_file</b><br>";

    // check apakah file merupakan direktori
    if (is_dir($nama_file))
    {
```

90

```
        echo "File adalah direktori.<br>";
    }

    // check apakah file merupakan link
    if (is_link($nama_file))
    {
        echo "File adalah link.<br>";
    }

    // check apakah file executable
    if (is_executable($nama_file))
    {
        echo "File sifatnya executable.<br>";
    }

    // check apakah file readable
    if (is_readable($nama_file))
    {
        echo "File sifatnya readable.<br>";
    }

    // check apakah file writable
    if (is_writable($nama_file))
    {
        echo "File sifatnya writable.<br>";
    }

    echo "Ukuran File : ".filesize($nama_file)."<br>";
    echo "Pemilik File : ".fileowner($nama_file)."<br>";
    echo "Group Pemilik File : ".filegroup($nama_file)."<br>";
    echo "Permission File : ".fileperms($nama_file)."<br>";
    echo "Jenis File : ".filetype($nama_file)."<br>";
}
else
{
    echo "<b>File tidak ditemukan!</b>";
}

?>
```

Pada sistem Unix/Linux, Anda akan mendapatkan hasil sebagai berikut:

```
Nama File : manuskrip.txt
File sifatnya readable.
File sifatnya writable.
Ukuran File : 681
```

91

Pemilik File : 538
Group Pemilik File : 100
Permission File : 33188
Jenis File : file

Pada sistem MS Windows 98, Anda kemungkinan akan mendapatkan hasil sebagai berikut:

Nama File : **manuskrip.txt**
File sifatnya readable.
File sifatnya writable.
Ukuran File : 681
Pemilik File : 0
Group Pemilik File : 0
Permission File : 33206
Jenis File : file

Inikah, Tanda-tandanya...

Sampai sejauh ini kita masih selalu membicarakan masalah pembacaan file, bagaimana dengan penulisan file? Penulisan file dalam PHP merupakan proses yang lebih kompleks dan melibatkan *pointer* file. *Pointer* file selain menandai file yang dibuka, juga memiliki informasi mengenai penanda letak/posisi dalam sebuah file yang dibuka. Proses penulisan (maupun pembacaan) dilakukan mulai pada posisi *pointer* file ini.

Fungsi yang akan kita gunakan untuk melakukan penulisan file adalah fungsi `fopen()`, yang sesungguhnya juga dapat digunakan untuk pembacaan file. Fungsi ini akan membuka sebuah *pointer* file, dan membutuhkan dua parameter yaitu nama file dan mode untuk file yang akan dibuka. Mode ini juga sekaligus menentukan posisi awal *pointer* file. Jenis-jenis mode pada fungsi `fopen()` adalah:

- Mode Hanya Baca (*read-only*), diaktivasi dengan parameter "r". Mode ini akan menempatkan posisi *pointer* file pada awal file.
- Mode Hanya Tulis (*write-only*), diaktivasi dengan parameter "w". Mode ini akan membuat file baru dan menempatkan posisi *pointer* file pada awal file. Jika file yang dibuka telah ada, maka isi dari file tersebut langsung dihapus.
- Mode Baca-Tulis (*read-write*), diaktivasi dengan parameter "r+". File dibuka untuk dapat dibaca maupun ditulis. Posisi *pointer* file adalah pada awal file.
- Mode Tambahkan/Append (hanya tulis), diaktivasi dengan parameter "a". Pada mode ini, posisi *pointer* file adalah pada akhir dari file yang akan memudahkan kita untuk menambah/menyambung data ke data yang telah ada. Jika file yang dimaksud tidak ditemukan, maka file yang baru akan langsung dibuat.
- Mode Tambahkan/Append (baca/tulis), diaktivasi dengan parameter "a+". Pada mode ini, posisi *pointer* file juga pada akhir dari file yang akan memudahkan kita untuk menambah/menyambung data ke data yang telah ada. Jika file yang dimaksud tidak

ditemukan, maka file yang baru akan langsung dibuat. Tambahannya, selain menambah data, kita juga dapat melakukan pembacaan file.

Untuk sistem operasi yang tidak dapat membedakan antara file biner dan file teks ASCII biasa, misalkan di keluarga MS Windows, dibutuhkan sebuah parameter lagi jika kita ingin mengakses file jenis biner, yaitu parameter "b". Parameter ini tidak berguna jika kita gunakan pada sistem operasi berbasis Unix atau Linux karena secara otomatis sistem filenya telah membedakan file biner dengan file teks ASCII.

Setelah membuka file dengan mode yang tepat, untuk melakukan penulisan ke file dapat kita gunakan fungsi `fputs()`. Jika kita selesai bekerja dengan file, maka kita harus menutup file tersebut dengan perintah `fclose()`. Contoh skrip PHP berikut ini akan menunjukkan caranya.

```
<?php
    // tentukan nama file
    $nama_file = "cobafile.txt";

    // buka file
    $kodefile = fopen($nama_file,"a");

    // tentukan data string
    $data = "Ini adalah transmisi sub space dari United
Federation
        of Planets. Prioritas transmisi ini adalah
pribadi dan
        rahasia. Jika benar demikian, apakah mungkin
Anda
        menangkap transmisi ini... ha... ha... ha...!";

    // tuliskan data ke file
    fputs($kodefile, $data);

    // tutup file
    fclose($kodefile);

?>
```

Pada skrip PHP di atas, kita menentukan terlebih dahulu nama file yang akan dibuka. Fungsi `fopen()` yang kita gunakan akan membuka file tersebut dengan mode *append write only* dan menghasilkan sebuah *pointer* file yang kita simpan pada variabel `$kodefile`. Jadi setiap yang kita gunakan variabel `$kodefile`, maksudnya adalah file₀₃

(dalam contoh di atas `cobafile.txt`), yang dibuka untuk proses *append write only*. Posisi *pointer* file akan diletakkan pada akhir dari file tersebut, untuk memastikan penambahan data tidak merusak data sebelumnya. Jika file yang dimaksud tidak ditemukan, maka PHP otomatis membuat file baru dengan nama `cobafile.txt`.

String data ditentukan setelah itu, kemudian dilakukan penulisan ke file dengan fungsi `fputs()`. Untuk memastikan penulisan dilakukan ke file yang benar, maka harus diyakinkan bahwa *pointer* file yang digunakan sebagai parameter dari fungsi `fputs()` adalah `$kodefile`. Parameter yang lain dari fungsi ini adalah string data yang akan ditulis, yang disimpan pada variabel `$data`. Terakhir, tentu saja, kita harus selalu membiasakan kerja rapi, jika kita membuka sesuatu, maka akhiri dengan menutup setelah semua aktivitas kita selesai. Membiarkan sesuatu tetap terbuka dapat mempermalukan kita, terutama jika berkaitan dengan hal-hal vital. Fungsi `fclose()` kita gunakan untuk menutup file yang telah kita buka sebelumnya. Parameter fungsi ini adalah *pointer* file, untuk memastikan kita telah menutup file yang benar.

Sekarang kita akan membuat sebuah contoh buku tamu yang bersahaja dengan ilmu yang telah kita dapatkan selama ini.

```
<html>
<head>
  <title>Buku Tamu Sederhana</title>
  <basefont face="Arial">
</head>

<body>

<?php

  // tentukan nama file
  $nama_file = "bukutamu.txt";

  // cek apakah ada form yang dikirim
  if ($simpan)
  {
    // buka file
    $kodefile = fopen($nama_file,"a");

    // simpan isi form

    fputs($kodefile, date("r",time())."<br>");

    if (trim($nama)== "") { $nama = "Mr. X"; }
    fputs($kodefile, "Nama : <b>$nama</b><br>");
```

```
        if (trim($e-mail)== "") { $email = "(tidak ada)"; }
        fputs($kodefile, "e-mail : <b>$email</b><br>");

        if (trim($komentar)== "") { $komentar = "(tidak
ada)"; }
        fputs($kodefile, "Komentar :
<b>$komentar</b><br>");

        fputs($kodefile, ".<br>");

        // tutup file
        fclose($kodefile);

    }

?>
<font size=5>Mohon Isi Buku Tamu Ini</font>
<form action="<?php echo $PHP_SELF; ?>" method="POST">
<table border=0 cellpadding=2 cellspacing=2>
<tr>
<td valign="top">Nama</td>
<td>
<input type="text" name="nama" size=50 maxlength=50>
</td>
</tr>

<tr>
<td>e-mail</td>
<td valign="top">
<input type="text" name="email" size=50 maxlength=50>
</td>
</tr>

<tr>
<td valign="top">Komentar</td>
<td>
<textarea name="komentar" rows="4"
cols="50"></textarea>
</td>
</tr>

<tr>
<td colspan=2 align="center">
<input type="submit" name="simpan" value="Simpan">
</td>
</tr>

</table>
</form>

<br>
<hr>
```

```
<br>

<?php

// tampilkan isi file jika ada
if (file_exists($nama_file))
{
    echo "<b>Daftar Buku Tamu Yang Ada</b><p>";
    readfile($nama_file);
}
else
{
    echo "File buku alamat belum ada!";
}

?>

</body>
</html>
```

Nah, sebuah buku tamu yang sederhana namun fungsional telah berhasil kita buat dengan PHP tanpa basis data. Anda mungkin menemukan fungsi baru yaitu `date()` dan `time()`, yang berhubungan dengan waktu saat proses dilakukan. Fungsi `time()` (tanpa parameter) akan memberikan waktu saat fungsi ini dijalankan dihitung dari jumlah detik semenjak Epoch Unix atau tanggal 1 Januari 1970 jam 00:00:00 GMT. Untuk dapat menampilkan informasi waktu ini dengan benar, maka dibutuhkan fungsi `date()` yang memiliki dua parameter/argumen. Argumen pertama adalah format waktu, dan argumen lainnya adalah informasi waktu, dalam hal ini adalah hasil dari fungsi `time()`. Format waktu "r" menghasilkan bentuk standar RFC 822, contohnya "Thu, 25 Dec 2001 17:05:07 +0700". Format-format lain mengenai tanggal dan waktu ini akan kita bicarakan pada artikel yang terpisah.

Saat skrip PHP di atas dijalankan untuk pertama kalinya, akan muncul form isian buku alamat, dan pesan bahwa file buku alamat belum ada. Jika form diisi dan dikirim dengan tombol simpan, maka skrip secara otomatis memanggil dirinya sendiri dan menambahkan data isian form ke file teks. Jika file tidak ditemukan, maka akan dibuat file baru. Setelah melakukan penambahan data teks, maka form kembali ditampilkan, dan isi file teks buku alamat ditampilkan setelah form. Demikian seterusnya. Ternyata semudah itu, ya!?

Jelangkung, Jelangkung, Di Sini Ada Pesta Kecil!

Bagian paling akhir dari artikel ini akan membahas teknik dasar membuat fungsi dalam skrip PHP. Mengapa kita perlu fungsi? Jawabannya adalah karena sifat pemalas kita membuat adanya keengganan kita menulis kode yang sama berulang-ulang. Hidup

akan lebih nyaman karena waktu penulisan kode jadi lebih lebih singkat, dan tentu saja, makin sedikit kode, jumlah kesalahan yang akan muncul juga relatif berkurang. Kening kita tidak akan terlalu banyak berkerut meneliti baris per baris kode yang salah.

Penulisan skrip dalam bentuk fungsi-fungsi terpisah akan membuat program kita lebih bersifat modular. Dengan model seperti ini, pengembangan lebih lanjut skrip kita menjadi lebih cepat dan mudah.

Sebagian besar perintah dalam skrip PHP adalah merupakan fungsi. Fungsi-fungsi yang kita gunakan selama ini, misalnya fungsi `echo()`, fungsi basis data, merupakan fungsi yang telah terdefinisi sejak awal dalam PHP. Kita tinggal menggunakannya sesuai dengan tujuan fungsi.

Sebagai makhluk yang tidak pernah puas, kadang-kadang kita ingin membuat fungsi-fungsi sendiri, yang menggabungkan fungsi-fungsi yang ada menjadi sesuai dengan keinginan di hati kita yang terdalam. Kedengarannya puitis sekali.

PHP menyediakan fasilitas pembuatan fungsi sendiri, sehingga hasrat Anda untuk hidup lebih nyaman akan terpenuhi. Sulitkah? Kalau sulit, berarti Anda sedang tidak mempelajari PHP...

Bentuk dasar pendeklarasian atau pendefinisian fungsi adalah sebagai berikut:

```
<?php
    function jelangkung()
    {
        [perintah 1];
        [perintah 2];
        [perintah 3];
        .... dan seterusnya ...
    }
??gt;
```

Setelah mendefinisikan sebuah fungsi, maka kita dapat dengan mudah memanggilnya melalui skrip. Lihatlah contoh berikut ini.

```
<?php  
  
function jelangkung()  
{  
    [perintah 1];  
    [perintah 2];  
    [perintah 3];  
    .... dan seterusnya ...  
}  
  
jelangkung();  
  
??gt;
```

Pada saat parser PHP menemukan perintah pemanggilan fungsi, maka kontrol eksekusi program akan beralih pada lokasi dimana fungsi tersebut didefinisikan. Setelah seluruh perintah dalam blok fungsi itu dijalankan, maka kontrol eksekusi program kembali ke lokasi tempat pemanggilan fungsi tersebut.

Menjadi Calon Kadet Starfleet

Hm... main jelangkung ternyata tidak menarik. Kembali kita ke Star Trek. Contoh berikut ini akan menjelaskan mengenai implementasi fungsi sederhana. Berikut ini adalah pertanyaan dan jawaban soal benar/salah untuk seleksi calon kadet Starfleet, kekuatan militer United Federation of Planets.

```
<html>  
<head>  
    <title>Latihan Soal Jawab Calon Kadet Starfleet</title>  
    <basefont face="Arial">  
</head>  
  
<body>  
  
<?php  
  
    // definisikan fungsi-fungsi  
    function jawaban_benar()  
    {  
        echo "Benar!";  
    }  
  
    function jawaban_salah()  
    {
```

```
        echo "Salah!";  
    }  
  
?>  
<p>  
Ras yang pertama kali berjumpa dengan ras bumi adalah ras  
Vulcan.<br>  
Jawaban : <?php jawaban_benar() ?>  
  
<p>  
Kode pesawat USS Enterprise pada Star Trek: The Original Series  
adalah  
NCC-1701-C.<br>  
Jawaban : <?php jawaban_salah() ?>  
  
<p>  
Kapal Angkasa ras Romulan (War Bird) dapat hilang dari  
pandangan.<br>  
Jawaban : <?php jawaban_benar() ?>  
  
<p>  
Teknologi Transporter dan Replikator memiliki prinsip kerja  
yang sama.<br>  
Jawaban : <?php jawaban_benar() ?>  
  
<p>  
Teknologi Warp yang digunakan ras Borg menggunakan Transwarp  
Conduit.<br>  
Jawaban : <?php jawaban_benar() ?>  
  
</body>  
</html>
```

Berikut ini adalah hasil skrip PHP bila dijalankan.

Ras yang pertama kali berjumpa dengan ras bumi adalah ras Vulcan.
Jawaban : Benar!

Kode pesawat USS Enterprise pada Star Trek: The Original Series adalah
NCC-1701-C.
Jawaban : Salah!

Kapal Angkasa ras Romulan (War Bird) dapat hilang dari pandangan.
Jawaban : Benar!

Teknologi Transporter dan Replikator memiliki prinsip kerja yang sama.

Jawaban : Benar!

Teknologi Warp yang digunakan ras Borg menggunakan Transwarp Conduit.

Jawaban : Benar!

Kita dapat memanggil sebuah fungsi sebelum didefinisikan, asal masih dalam satu skrip. Ubahlah skrip sebelumnya menjadi seperti di bawah ini. Hasil keluarannya ditanggung tidak berbeda.

```
<html>
<head>
    <title>Latihan Soal Jawab Calon Kadet Starfleet</title>
    <basefont face="Arial">
</head>

<body>

<p>
Ras yang pertama kali berjumpa dengan ras bumi adalah ras
Vulcan.<br>
Jawaban : <?php jawaban_benar() ?>

<p>
Kode pesawat USS Enterprise pada Star Trek: The Original Series
adalah
NCC-1701-C.<br>
Jawaban : <?php jawaban_salah() ?>

<p>
Kapal Angkasa ras Romulan (War Bird) dapat hilang dari
pandangan.<br>
Jawaban : <?php jawaban_benar() ?>

<p>
Teknologi Transporter dan Replikator memiliki prinsip kerja
yang sama.<br>
Jawaban : <?php jawaban_benar() ?>

<p>
Teknologi Warp yang digunakan ras Borg menggunakan Transwarp
Conduit.<br>
Jawaban : <?php jawaban_benar() ?>

<?php

// definisikan fungsi-fungsi
```

```
function jawaban_benar()
{
    echo "Benar!";
}

function jawaban_salah()
{
    echo "Salah!";
}

?>

</body>
</html>
```

Apa Yang Kuberikan, Apa Yang Kudapat

Fungsi-fungsi dapat pula kita berikan argumen untuk diolah di dalam fungsi, dan dapat pula kita buat agar memberikan sebuah hasil untuk diolah di luar fungsi. Contoh berikut adalah hasil modifikasi fungsi pada skrip terdahulu, sehingga fungsi-fungsi tersebut dapat memberikan sebuah hasil untuk diolah di luar fungsi.

```
<html>
<head>
    <title>Latihan Soal Jawab Calon Kadet Starfleet</title>
    <basefont face="Arial">
</head>

<body>

<?php

    // definisikan fungsi-fungsi
function jawaban_benar()
{
    $jawaban = "Benar!";
    return $jawaban;
}

function jawaban_salah()
{
    $jawaban = "Salah!";
    return $jawaban;
}
```

```
?>
<p>
Ras yang pertama kali berjumpa dengan ras bumi adalah ras
Vulcan.<br>
Jawaban : <?php $hasil = jawaban_benar(); echo $hasil; ?>

<p>
Kode pesawat USS Enterprise pada Star Trek: The Original Series
adalah
NCC-1701-C.<br>
Jawaban : <?php $hasil = jawaban_salah(); echo $hasil; ?>

<p>
Kapal Angkasa ras Romulan (War Bird) dapat hilang dari
pandangan.<br>
Jawaban : <?php $hasil = jawaban_benar(); echo $hasil; ?>

<p>
Teknologi Transporter dan Replikator memiliki prinsip kerja
yang sama.<br>
Jawaban : <?php $hasil = jawaban_benar(); echo $hasil; ?>

<p>
Teknologi Warp yang digunakan ras Borg menggunakan Transwarp
Conduit.<br>
Jawaban : <?php $hasil = jawaban_benar(); echo $hasil; ?>

</body>
</html>
```

Terlihat perbedaan cara penanganan hasil fungsi. Pada skrip sebelumnya fungsi tidak memberikan hasil/nilai yang dapat diolah, tetapi langsung menampilkan sesuatu pada media keluaran standar (melalui perintah `echo()`). Karena fungsi yang telah dimodifikasi tidak dapat langsung menampilkan hasil fungsi tetapi memberikan hasil/nilai, maka sebuah variabel (dalam skrip di atas adalah `$hasil`) ditugaskan untuk menampung nilai yang dihasilkan oleh fungsi, sebelum ditampilkan dengan perintah `echo()`.

Contoh berikut ini memberikan gambaran mengenai argumen/parameter fungsi.

```
<html>
<head>
  <title>Persamaan Luas Segitiga</title>
  <basefont face="Arial">
</head>
```

```
<body>

<?php

    // definisikan fungsi-fungsi
    function luas_segitiga($alas, $tinggi)
    {
        $jawaban = 0.5 * $alas * $tinggi;
        return $jawaban;
    }

    $alas_segitiga = 6;
    $tinggi_segitiga = 9;

?>
<p>
Luas Segitiga dengan<br>
- Alas : <?php echo $alas_segitiga; ?><br>
- Tinggi : <?php echo $tinggi_segitiga; ?><br>
Adalah =
<?php
    $hasil =          luas_segitiga($alas_segitiga,
$tinggi_segitiga);
    echo $hasil;
?>
<br>

</body>
</html>
```

Fungsi `luas_segitiga()` yang kita buat, membutuhkan dua argumen, yaitu argumen panjang alas (`$alas`) dan argumen tinggi segitiga (`$tinggi`). Nilai kedua argumen ini diberikan pada saat pemanggilan fungsi yang disimpan dalam bentuk variabel PHP selama fungsi tersebut dieksekusi agar dapat digunakan pada proses eksekusi perintah-perintah PHP di dalam blok fungsi. Fungsi ini akan menghasilkan nilai luas segitiga yang diperoleh dari operasi matematis terhadap kedua argumen tersebut.

Setiap saat di dalam skrip PHP, kita dapat memanggil/menggunakan secara berulang-ulang fungsi-fungsi yang kita definisikan, baik yang didefinisikan dalam skrip tersebut maupun pada skrip lain yang telah diikutsertakan dengan perintah `include()` atau `require()`. Betapa sebuah berita yang baik bagi kita, karena kita tidak perlu menghabiskan waktu lagi untuk menulis beberapa bagian kode secara berulang-ulang. Anda pun bisa memiliki lebih banyak waktu untuk menonton acara TV kegemaran Anda.

Masalah Pribadi dan Masalah Umum

Kini kita akan melihat bagaimana sebuah variabel dapat dipakai oleh fungsi. Aturan dasarnya adalah:

- Setiap variabel dalam sebuah fungsi adalah variabel lokal yang hanya berlaku di dalam fungsi itu sendiri. Variabel ini hanya bertahan dalam memori saat fungsi dieksekusi dan hilang dari memori saat fungsi selesai dieksekusi. Variabel lokal ini tidak dapat diakses dari luar fungsi.
- Fungsi pada dasarnya tidak dapat mengakses variabel yang berada di luar fungsi tersebut.
- Operasi terhadap argumen di dalam fungsi, tidak akan berpengaruh apa-apa terhadap variabel di luar fungsi yang digunakan sebagai argumen, bahasa dewanya, setiap argumen bersifat *passed by value*.

Mari kita lihat contoh skrip berikut ini.

```
<?php

    // berikan nilai ke sebuah variabel di luar fungsi
    $jenis_kecepatan = "impuls";

    // fungsi yang akan mengubah nilai variabel
    function ubah_kecepatan($nama)
    {
        $jenis_kecepatan = $nama;
        return $jenis_kecepatan;
    }

    // tampilkan nilai variabel sebelum pemanggilan fungsi
    echo "Sebelum pemanggilan fungsi, jenis kecepatan adalah
    $jenis_kecepatan.<p>" ;

    // panggil fungsi
    ubah_kecepatan("warp");

    echo "Setelah pemanggilan fungsi, jenis kecepatan adalah
    $jenis_kecepatan." ;

?>
```

Jalankan skrip PHP di atas, dan Anda akan dapatkan hasil sebagai berikut.

Sebelum pemanggilan fungsi, jenis kecepatan adalah impuls.

Setelah pemanggilan fungsi, jenis kecepatan adalah impuls.";

Perubahan nilai variabel `$jenis_kecepatan` di dalam fungsi tidak berpengaruh terhadap nilai variabel `$jenis_kecepatan` yang di luar fungsi, karena kedua variabel ini berbeda konteksnya. Variabel `$jenis_kecepatan` yang ada di dalam fungsi adalah variabel lokal fungsi dan tidak ada kaitannya dengan variabel `$jenis_kecepatan` yang berada di luar fungsi. Itu sebabnya nilai variabel `$jenis_kecepatan` di luar fungsi tetap tidak berubah setelah pemanggilan fungsi.

Bagaimana jika Anda ingin sekali mengubah nilai variabel di luar fungsi tapi dari dalam fungsi? Ada dua pilihan. Pilihan pertama, Anda putus asa dan tidak mau menggunakan PHP lagi, atau Anda dapat gunakan fungsi/perintah `global`.

Fungsi/perintah `global` akan membuat pengecualian dari aturan di atas, dan langsung membuat variabel yang ada pada fungsi mereferensikan variabel dengan nama sama yang ada di luar fungsi. Tidak usah bingung atas bahasa berbelit ini, langsung saja perhatikan modifikasi skrip sebelumnya.

```
<?php
// berikan nilai ke sebuah variabel di luar fungsi
$jenis_kecepatan = "impuls";

// fungsi yang akan mengubah nilai variabel
function ubah_kecepatan($nama)
{
    global $jenis_kecepatan;
    $jenis_kecepatan = $nama;
    return $jenis_kecepatan;
}

// tampilkan nilai variabel sebelum pemanggilan fungsi
echo "Sebelum pemanggilan fungsi, jenis kecepatan adalah
$jenis_kecepatan.<p>" ;

// panggil fungsi
ubah_kecepatan("warp");

echo "Setelah pemanggilan fungsi, jenis kecepatan adalah
$jenis_kecepatan." ;

?>
```

Kini coba Anda jalankan skrip ini lagi, niscaya Anda akan mendapatkan hasil seperti di bawah ini.

Sebelum pemanggilan fungsi, jenis kecepatan adalah impuls.

Setelah pemanggilan fungsi, jenis kecepatan adalah warp.";

Kebebasan Yang Berarti

Sejauh ini, fungsi-fungsi yang Anda buat selalu tertentu jumlah argumennya. Bisa tidak ada sama sekali, atau dalam jumlah yang pasti. Bagaimana jika kita tidak dapat memastikan jumlah argumen yang kita masukkan ke sebuah fungsi? Kadang satu, dua atau sekali waktu sepuluh argumen. Bisakah PHP menangani hal ini?

Karena topik ini masuk dalam artikel, jelas jawabannya pasti "bisa". Baiklah, sekarang kita lihat caranya.

PHP 4, menyediakan fungsi-fungsi yang dapat kita gunakan untuk maksud ini, yaitu fungsi `func_num_args()` yang akan menghasilkan jumlah argumen yang diberikan pada sebuah fungsi, dan fungsi `func_get_args()` yang akan menyimpan semua argumen fungsi yang diberikan ke dalam bentuk variabel *array*. Kita dapat melihat implementasinya pada skrip berikut ini.

```
<html>
<head>
<title>Fungsi Dengan Jumlah Argumen Bebas</title>
</head>

<body>

<?php
function awak_enterprise()
{
    $argumen = func_get_args();
    return $argumen;
}

$nama_awk = awak_enterprise("Piccard", "Riker",
"LaForge", "Worf");

?>
```

Berikut ini adalah nama-nama awak kapal USS Enterprise:


```
<?php
for ($i=0; $i < sizeof($nama_awak); $i++)
{
    echo "<li>" . $nama_awak[$i] . "\n";
}
?>
</ul>
</body>

</html>
```

Hasil dari skrip di atas adalah:

Berikut ini adalah nama-nama awak kapal USS Enterprise:

- Piccard
- Riker
- LaForge
- Worf

Beginilah Akhirnya...

Selesai sudah perjalanan panjang kita dalam mempelajari dasar-dasar yang paling mendasar bagi pemula PHP. Dengan bekal lima bagian artikel ini, sesungguhnya Anda telah cukup siap untuk memulai sendiri membangun aplikasi atau situs dinamis dengan PHP. Tentunya, Anda harus banyak berlatih sendiri dan membaca artikel-artikel lain baik tingkat pemula, menengah maupun lanjutan/terapan. Kini Anda sadar, bahwa ternyata menggemari Star Trek dapat membantu Anda, paling tidak akan melatih Anda untuk berani mengunjungi tempat yang belum pernah dikunjungi oleh siapa pun.

Tetaplah belajar PHP, dan jangan bosan menantikan topik lain dari belajar PHP. Sampai di sini dulu Merdeka!*)

Biografi dan Profil



Ivan Irawan. Lulus Cum Laude dari Teknik Nuklir Universitas Gadjah Mada tahun 1995. Tinggal di Semarang, memiliki pengalaman lebih dari 7 tahun pada operasional bank swasta nasional. Saat ini mendirikan usaha konsultan sistem informasi [AtlantisIndonesia](#) yang berbasis di Semarang. Konsultan Teknologi Informasi dan Manajemen Proses Operasi pada beberapa bank, institusi pembiayaan/finansial, dan perusahaan manufaktur. Mendalami berbagai bahasa pemrograman Visual Basic, Delphi/Kylix, C/C++, Java, dan PHP.

Ivan Irawan adalah programmer AI-Billinet, sistem billing dan administrasi warnet yang telah digunakan di lebih dari 100 warung internet di seluruh Indonesia. Aktif di Kelompok Pengguna Linux Semarang (KPLi Semarang) dan telah menulis beberapa buku komputer dan artikel-artikel pemrograman.

Informasi lebih lanjut tentang penulis ini bisa didapat melalui:

URL: <http://skriphpp.xoasis.com>

Email: rii@ai.co.id