Dr. Joseph Teguh Santoso, S.Kom, M.Kom

Pengembangan Web PHP

(Hypertext Prepocessor)

YAYASAN PRIMA AGUS TEKNIK

Pengembangan Web PHP (Hypertext Prepocessor)

Penulis :
Dr. Joseph Teguh Santoso, S.Kom., M.Kom

ISBN : 9 786235 734897

Editor :
Muhammad Sholikan, M.Kom

Penyunting :
Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

Desain Sampul dan Tata Letak :
Irdha Yunianto, S.Ds., M.Kom

Penebit :
Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

Redaksi :
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Distributor Tunggal :
Universitas STEKOM
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : info@stekom.ac.id

KATA PENGANTAR

Puji syukur pada Tuhan Yang Maha Esa bahwa buku yang berjudul "***Pengembangan Web PHP (Hypertext Preprocessor)***" ini dapat diselesaikan dengan baik. Hypertext Preprocessor atau yang biasa disebut PHP adalah sebuah bahasa pemrograman server-side berbasiskan kode-kode (script) yang digunakan untuk mengolah suatu data dan kemudian script dari PHP akan di proses di server. Jenis server yang sering digunakan antara lain Apache dan Nginx. PHP adalah bahasa pemrograman yang bersifat open source, artinya pengguna diberi kebebasan untuk mengembangkan dan memodifikasi sesuai dengan keinginannya. PHP biasanya digunakan untuk pengembangan website (dinamis atau statis) dan membuat aplikasi komputer. PHP mendukung banyak jumlah protokol besar seperti POP3,IMAP, dan LDAP yang dapat diintegrasikan dengan berbagai database populer seperti MySQL, Oracle, Sybase, dan Microsoft SQL Server. Mengapa menggunakan PHP? Berdasarkan hasil survey yang telah dilakukan oleh W3tech.com, bahasa pemrograman PHP digunakan 79% website diseluruh dunia, kemudian diikuti ASP.NET diposisi kedua dengan angka 10,7% dan urutan ketiga Java dengan angka 3,8%. Beberapa aspek yang membuat bahasa PHP begitu populer antara lain proses belajar pemrograman PHP relatif lebih mudah dibandingkan dengan bahasa lain, sebagian besar server web hosting secara default mendukung PHP, PHP dapat dijalankan diberbagai macam sistem operasi yang berbeda seperti Windows, Mac OS, Linux, dan sebagainya. Contoh website besar terkenal yang pengembangan webnya menggunakan bahasa pemrograman PHP ialah Facebook, Wordpress, Yahoo, dan Wikipedia.

Buku ini ditujukan bagi pembaca yang baru memulai untuk belajar tentang bahasa pemrograman PHP, karena didalam buku ini dibagi menjadi 6 bab dengan materi Pengenalan PHP dimulai dari pengenalan PHP, ekspresi dan kontrol PHP, langkah dasar membangun skrip PHP, sampai kunci praktis mempelajari bahasa pemrograman PHP. Akhir kata penulis ucapkan terima kasih dan semoga buku ini bisa memberikan manfaat bagi pembaca.

Semarang, Agustus 2022
Penulis

Dr. Joseph Teguh Santoso, M.Kom

DAFTAR ISI

BAB 1
PENGENALAN PHP

1.1 MEMAHAMI DASAR-DASAR PHP

PHP adalah bahasa scripting yang dirancang khusus untuk digunakan di web. PHP memiliki fitur untuk memprogram tugas-tugas yang diperlukan untuk mengembangkan aplikasi web dinamis. Bab ini dimulai dengan dasar-dasar dan berlanjut ke materi yang lebih kompleks. Bahkan jika kita pernah memiliki pengalaman dengan PHP sebelumnya. Bab ini menjelaskan tentang berbagai dasar penulisan skrip PHP — aturan yang berlaku untuk semua pernyataan PHP. Anggap saja, aturan dalam penulisan php ini sama dengan aturan tata bahasa dan tanda baca pada umumnya.

Cara Kerja PHP

Software PHP (perangkat lunak yang mengirimkan halaman web ke dunia) bekerja dengan server web. Ketika kita mengetik URL ke bilah alamat browser web, kita akan mengirim pesan ke server web pada URL tersebut, memintanya untuk mengirimi kita file HTML. Server web merespons dengan mengirimkan file yang diminta. Browser kita membaca file HTML dan menampilkan halaman web. Kita juga meminta file dari server web saat kita mengklik tautan di halaman web. Selain itu, server web memproses file saat kita mengklik tombol halaman web yang mengirimkan formulir. Ketika PHP diinstal, kita meminta file, server web menjalankan PHP, dan mengirimkan HTML kembali ke browser, berkat pemrograman di PHP.

1.2 CARA SERVER WEB DALAM MEMPROSES FILE PHP

Saat browser diarahkan ke file HTML (biasanya dengan ekstensi .html atau .htm), server web mengirimkan file apa adanya kepada browser, selanjutnya browser akan memproses file dan menampilkan halaman web yang dijelaskan oleh tag HTML dalam file. Ketika browser diarahkan ke file PHP (dengan ekstensi .php), server web akan mencari bagian PHP dalam file dan memprosesnya. Server web/prosesor PHP memproses file PHP dengan cara:

1. Server web mulai scanning file dalam mode HTML. Ini mengasumsikan pernyataan HTML dan mengirimkannya ke browser tanpa pemrosesan apa pun.
2. Server web berlanjut dalam mode HTML hingga menemukan tag pembuka PHP (<?php).
3. Ketika menemukan tag pembuka PHP, server web menyerahkan pemrosesan ke modul PHP. Server web kemudian mengasumsikan bahwa semua pernyataan adalah pernyataan PHP dan menggunakan modul PHP untuk mengeksekusi pernyataan PHP. Jika ada output dari PHP, server mengirimkan output ke browser.
4. Server web terus berada pada mode PHP hingga menemukan tag penutup PHP (?>).
5. Ketika server web menemukan tag penutup PHP, ia kembali ke mode HTML. Ini melanjutkan pemindaian, dan siklus berlanjut dari Langkah 1

Lebih khusus lagi, ketika PHP diinstal, server web dikonfigurasi untuk mengharapkan ekstensi file tertentu berisi pernyataan bahasa PHP. Seringkali ekstensinya adalah .php atau .phtml, tetapi ekstensi apa pun dapat digunakan. (Dalam buku ini, kita berasumsi bahwa .php adalah ekstensi untuk skrip PHP.) Ketika server web mendapat permintaan untuk file dengan ekstensi yang ditentukan, ia mengirimkan pernyataan HTML apa adanya, tetapi pernyataan PHP diproses oleh perangkat lunak PHP sebelum dikirim ke pemohon.

Memasukkan PHP kedalam HTML

Secara default, dokumen PHP diakhiri dengan ekstensi .php. Ketika server web menemukan ekstensi ini dalam file yang diminta, maka server beb meneruskannya ke

prosesor PHP secara otomatis sehingga server web dapat dikonfigurasi, dan beberapa web developer memilih untuk memaksa file yang diakhiri dengan .htm atau .html untuk diurai oleh prosesor PHP, biasanya karena mereka ingin menyembunyikan fakta bahwa mereka menggunakan PHP.

Program PHP kita bertanggung jawab untuk mengirimkan kembali file bersih yang cocok untuk ditampilkan di browser web. Paling sederhana, dokumen PHP hanya akan menampilkan HTML. Untuk membuktikan ini, kita dapat mengambil dokumen HTML biasa seperti file index.html dan menyimpannya sebagai index.php, dan itu akan ditampilkan identik dengan aslinya. Untuk memicu perintah PHP, kita perlu mempelajari tag baru. Tag bagian pertama dari php adalah:

```
<?php
```

Hal pertama yang mungkin kita perhatikan adalah bahwa tag belum ditutup. Ini karena seluruh bagian PHP dapat ditempatkan di dalam tag ini, dan selesai hanya ketika bagian penutup ditemukan, yang terlihat seperti ini:

```
?>
```

Contoh untuk Memanggil PHP:

```
<?php
Echo “Hello world”;
?>
```

Cara kita menggunakan tag ini cukup fleksibel. Beberapa programmer membuka tag di awal dokumen dan menutupnya tepat di akhir, mengeluarkan HTML apa pun langsung dari perintah PHP. Namun, yang lain memilih untuk menyisipkan hanya fragmen PHP terkecil yang mungkin ke dalam tag ini di mana pun skrip dinamis diperlukan, meninggalkan sisa dokumen dalam HTML standar. Jenis pemrogram yang terakhir umumnya berpendapat bahwa gaya pengkodean mereka menghasilkan kode yang lebih cepat, sedangkan yang pertama mengatakan bahwa peningkatan kecepatan sangat minimal sehingga tidak membenarkan kompleksitas tambahan untuk keluar masuk PHP berkali-kali dalam satu dokumen.

Saat kita mempelajari lebih lanjut, kita pasti akan menemukan gaya pengembangan PHP yang kita sukai. Saya telah mengadopsi pendekatan untuk menjaga jumlah transfer antara PHP dan HTML seminimal mungkin— umumnya hanya sekali atau dua kali dalam sebuah dokumen. Omong-omong, ada sedikit variasi pada sintaks PHP. Jika kita menelusuri Internet untuk contoh PHP, kita mungkin juga menemukan kode di mana sintaks pembuka dan penutup terlihat seperti ini:

```
<?
Echo “Hello world”;
?>
```

Meskipun tidak begitu jelas bahwa parser PHP sedang dipanggil, ini merupakan sintaks alternatif yang valid yang biasanya juga berfungsi, tetapi harus dihindari karena tidak kompatibel dengan XML dan penggunaannya sekarang tidak digunakan lagi (artinya tidak lagi direkomendasikan dan dapat dihapus di versi mendatang).

Catatan: Jika kita hanya memiliki kode PHP dalam sebuah file, kita dapat menghilangkan penutup ?>. Ini bisa menjadi praktik yang baik, karena akan memastikan bahwa kita tidak memiliki spasi berlebih yang bocor dari file PHP kita (terutama penting saat kita menulis kode berorientasi objek).

Ketika pernyataan bahasa PHP diproses, hanya output, atau apa pun yang diprint ke layar, yang dikirim oleh server web ke browser web. Pernyataan bahasa PHP, yang tidak menghasilkan output apa pun ke layar, tidak disertakan dalam output yang dikirim ke browser, sehingga kode PHP biasanya tidak terlihat oleh pengguna. Misalnya:

```
<?php echo “<p>Hello World</p>”; ?>
```

Di sini, echo adalah instruksi PHP yang memberitahu PHP untuk menampilkan teks yang akan datang. Perangkat lunak PHP memproses pernyataan PHP dan menghasilkan yang berikut:

```
<p>Hello World</p>
```

PHP tidak terintegrasi dengan semua server web tetapi berfungsi dengan banyak server web populer. PHP bekerja dengan baik dengan server web Apache. PHP juga bekerja dengan Microsoft Internet Information Services (IIS) dan lainnya. Jika kita dapat memilih atau memengaruhi pemilihan server web yang digunakan di organisasi Anda, Apache dapat kita pilih karena gratis, open source, stabil, dan populer. Saat ini mendukung lebih dari 60 persen dari semua situs web, menurut survei server web di www.netcraft.com. Apache berjalan di Windows, Linux, Mac OS, dan sebagian besar Unix.

Menggunakan PHP

Dengan menggunakan PHP penyematan aktivitas dinamis di halaman web akan menjadi sangat mudah. Saat kita memberi halaman ekstensi .php, mereka memiliki akses instan ke bahasa skrip. Dari sudut pandang *developer*, yang harus kita lakukan hanyalah menulis kode seperti berikut:

```
<?php
 Echo “ Hari ini adalah “ . date(“l”) . “. “;
?>
Hari ini adalah hari terakhir
```

Pembukaan <?php memberitahu server web untuk mengizinkan program PHP menginterpretasikan semua kode hingga tag ?>. Di luar konstruksi ini, semuanya dikirim ke client sebagai HTML langsung. Jadi teks yang ditampilkan hanyalah output ke browser; dalam tag PHP, fungsi tanggal bawaan menampilkan hari saat ini dalam seminggu sesuai dengan waktu sistem server. Hasil akhir-nya akan terlihat seperti ini:

```
Hari ini adalah hari Rabu. Ini adalah informasi terakhir
```

PHP adalah bahasa yang fleksibel, dan beberapa orang lebih suka menempatkan konstruksi PHP langsung di sebelah kode PHP, seperti ini:

```
Hari ini <?php echo date(“l”); ?>. Ini adalah informasi terakhir
```

Intinya adalah bahwa dengan PHP, web developer memiliki bahasa skrip yang, meskipun tidak secepat kompilasi kode kita dalam C atau bahasa serupa, sangat cepat dan juga terintegrasi dengan markup HTML.

Catatan: Jika kita bermaksud memasukkan contoh PHP dalam buku ini untuk bekerja bersama saya, kita harus ingat untuk menambahkan <?php di depan dan ?> setelahnya untuk memastikan bahwa juru bahasa PHP memprosesnya.

1.3 STRUKTUR PHP

Bagian ini tidak terlalu sulit, tetapi saya tetap menyarankan untuk mengerjakannya dengan hati-hati, karena ini menjadi dasar dari berbagai struktur php dalam buku ini.

Meneliti Struktur Script PHP

PHP adalah bahasa scripting tertanam ketika digunakan di halaman web. Ini berarti bahwa kode PHP tertanam dalam kode HTML. Penggunaan tag HTML untuk menyertakan bahasa PHP yang disematkan dalam file HTML— dengan cara yang sama saat menggunakan tag HTML lainnya. Kita membuat dan mengedit halaman web yang berisi PHP dengan cara yang sama seperti kita membuat dan mengedit halaman HTML biasa. Pernyataan bahasa PHP terlampir dalam tag PHP dengan bentuk berikut:

```
<?php          ?>
```

Terkadang, kita juga dapat menggunakan versi tag PHP yang lebih pendek. Kita dapat mencoba menggunakan <? dan ?> tanpa php. Jika tag pendek diaktifkan, kita dapat menyimpan sedikit pengetikan. Namun, jika kita menggunakan tag pendek, skrip kita tidak akan berjalan jika dipindahkan ke host web lain di mana tag pendek PHP tidak diaktifkan.

Sebagai contoh, mulailah dengan skrip HTML yang menampilkan Hello World! di jendela browser, ditunjukkan pada skrip dibawah ini.

Skrip PHP Hello World

```
<!doctype html>
<html>
<head><title>Hello World Script</title></head>
<body>
<?php
  Echo “<p>Hello World!</p>\n”;
?>
</body>
</html>
```

Jangan melihat file secara langsung dengan browser Anda. Artinya, jangan pilih File⇨Open File dari menu browser kita untuk menavigasi ke file dan mengkliknya. Kita harus membuka file dengan mengetikkan URL-nya di bilah alamat browser. Jika kita melihat kode PHP ditampilkan di jendela browser alih-alih output yang kita harapkan, kita mungkin belum memulai file dengan URL-nya.

Dalam skrip PHP ini, bagian PHP adalah

```
<?php
  Echo “<p>Hello World!</p>;
?>
```

Tag PHP hanya menyertakan satu pernyataan — pernyataan echo. Pernyataan echo adalah pernyataan PHP yang akan sering kita gunakan. Outputnya teks sederhana yang disertakan di antara tanda kutip ganda. Ketika bagian PHP diproses, itu diganti dengan output. Dalam hal ini, outputnya adalah:

```
<p>Hello World!</p>
```

Jika kita mengganti bagian PHP dari skri diatas dengan output sebelumnya, skrip sekarang terlihat persis seperti skrip HTML. Jika kita membuka salah satu skrip di browser, kita akan melihat halaman web yang sama. Jika kita melihat source code yang dilihat browser (di browser, pilih View⇨Source), kita melihat daftar source code yang sama untuk kedua skrip. Kita dapat memiliki sebanyak mungkin bagian PHP dalam skrip yang kita butuhkan, dengan Bagian HTML sesuai kebutuhan, termasuk nol bagian PHP atau HTML. Misalnya, skrip berikut memiliki dua bagian PHP dan dua bagian HTML:

```
<html>
<head><title>Hello World Script</title></head>
<body>
<?php
  echo “<p>Hello World! </p>”;
?>
<p>This is HTML only. </p>”;
<?php
  echo “<p>Hello World! </p>”;
?>
<p> This is a second HTML section. </p>
</html>
```

1.4 MENGGUNAKAN COMMENT

Ada dua cara di mana kita dapat menambahkan komentar ke kode PHP Anda. Yang pertama mengubah satu baris menjadi komentar dengan mendahuluinya dengan sepasang garis miring, seperti ini:

```
// This is a comment
```

Versi fitur komentar ini adalah cara yang bagus untuk menghapus sementara satu baris kode dari program yang memberi kita kesalahan. Misalnya, kita dapat menggunakan komentar semacam itu untuk menyembunyikan baris kode debug hingga kita membutuhkannya, seperti ini:

```
// echo "X equals $x";
```

Kita juga dapat menggunakan jenis komentar ini langsung setelah baris kode untuk menjelaskan tindakannya, seperti ini:

```
$x += 10; // Increment $x by 10
```

Saat kita membutuhkan komentar beberapa baris, ada jenis komentar kedua, yang terlihat seperti contoh ini:

Contoh Multiline Comment

```
<?php
?*This is a section
   of multiline comments
   which will not be
   interpreted */
?>
```

Kita dapat menggunakan pasangan karakter /* dan */ untuk membuka dan menutup komentar hampir di mana saja kita suka di dalam kode Anda. Sebagian besar, jika tidak semua, programmer menggunakan konstruksi ini untuk sementara mengomentari seluruh bagian kode yang tidak berfungsi atau, karena satu dan lain alasan, mereka tidak ingin ditafsirkan.
Catatan: Kesalahan umum adalah menggunakan /* dan */ untuk mengomentari sebagian besar kode yang sudah berisi bagian yang dikomentari yang menggunakan karakter tersebut. Kita tidak dapat membuat sarang komentar dengan cara ini; penerjemah PHP tidak akan tahu di mana komentar berakhir dan akan menampilkan pesan kesalahan. Namun, jika kita menggunakan editor program atau IDE dengan penyorotan sintaks, jenis kesalahan ini akan lebih mudah dikenali.

1.5 MELIHAT SINTAKS PHP

Bagian PHP yang di tambahkan ke file HTML terdiri dari serangkaian pernyataan PHP. Setiap pernyataan PHP merupakan instruksi untuk PHP dalam melakukan atau menjalankan sesuatu. Sintaks bahasa PHP mirip dengan sintaks C. PHP sebenarnya lebih sederhana daripada C karena tidak menyertakan beberapa konsep C yang lebih sulit — konsep yang tidak diperlukan untuk memprogram situs web.

Menggunakan pernyataan sederhana dalam PHP

Pernyataan sederhana adalah instruksi untuk PHP dalam melakukan satu tindakan sederhana. Pernyataan echo yang ditunjukkan pada skrip dibawah ini merupakan pernyataan PHP sederhana yang menginstruksikan PHP untuk menampilkan teks di antara tanda kutip ganda. Pernyataan sederhana PHP mengikuti aturan berikut:

- Pernyataan PHP diakhiri dengan titik koma atau tag akhir PHP. PHP tidak memperhatikan spasi atau akhir baris, yang berarti PHP akan terus membaca pernyataan sampai menemukan titik koma atau tag penutup PHP, tidak peduli berapa banyak baris pernyataan itu.
- Pernyataan PHP dapat ditulis dalam huruf besar atau kecil. Dalam pernyataan echo, Echo, echo, ECHO, dan eCHo semuanya sama untuk PHP..

Berikut ini adalah pernyataan PHP yang valid dan menghasilkan output yang sama seperti yang kita lihat sebelumnya:

```
<?php
  Echo “<p>Hello</p>”
?>
```

Pernyataan echo berada pada baris yang sama dengan tag PHP. PHP membaca pernyataan hingga mencapai tag penutup, yang dilihat PHP sebagai akhir pernyataan. Contoh berikutnya juga menghasilkan output yang sama:

```
<?php
```

```
    echo “<p>Hello</p>”; echo “<p>World</p>
?>
```

Contoh ini berisi dua pernyataan echo PHP pada satu baris, keduanya diakhiri dengan titik koma. Jika mau, kita dapat menulis seluruh bagian PHP dalam satu baris panjang, selama kita memisahkan pernyataan dengan titik koma. Namun, naskah yang ditulis dengan cara ini akan sulit dibaca orang.

Sintaks Dasar

PHP adalah bahasa yang cukup sederhana dengan akar di C dan Perl, namun lebih mirip Java. PHP juga sangat fleksibel, tetapi ada beberapa aturan yang perlu kita pelajari tentang sintaks dan strukturnya.

Titik koma

Kita mungkin telah memperhatikan poin ini pada contoh sebelumnya bahwa perintah PHP diakhiri dengan titik koma, seperti ini:

```
$x += 10;
```

Mungkin penyebab kesalahan paling umum yang akan kita temui dengan PHP adalah melupakan titik koma ini. Hal ini menyebabkan PHP memperlakukan beberapa pernyataan seperti yang tidak dapat dipahami, membuatnya menghasilkan pesan kesalahan Parse.

Simbol $

Simbol $ telah digunakan dalam berbagai cara oleh berbagai bahasa pemrograman. Misalnya, jika kita pernah menulis dalam bahasa BASIC, kita akan menggunakan $ untuk mengakhiri nama variabel untuk menyatakannya sebagai string. Sedangkan, dalam PHP, kita harus menempatkan $ di depan semua variabel. Ini diperlukan untuk membuat parser PHP lebih cepat, karena parser PHP langsung tahu kapan pun menemukan variabel. Baik variabel kita berupa angka, string, atau array, semuanya harus terlihat seperti pada Contoh di bawah ini.

Contoh tiga jenis penugasan variabel yang berbeda

```
<?php
    $mycounter    = 1;
    $mystring     = “Hello”;
    $myarray      = array(“One”, “Two”, “Three”);
?>
```

Cukup banyak sintaks yang harus kita ingat. Tidak seperti bahasa yang sangat ketat tentang bagaimana kita membuat indentasi dan layout kode kita (misalnya, Python), PHP membuat kita benar-benar bebas untuk menggunakan (atau tidak menggunakan) semua indentasi dan spasi yang kita suka. Faktanya, penggunaan spasi putih yang masuk akal umumnya dianjurkan (bersama dengan komentar komprehensif) untuk membantu memahami kode. Ini juga membantu programmer lain ketika mereka harus memelihara kode Anda.

Menggunakan pernyataan kompleks

Terkadang kelompok pernyataan sederhana digabungkan menjadi satu blok. Sebuah blok diapit oleh kurung kurawal, { dan }. Sebuah blok pernyataan dieksekusi bersama-sama. Penggunaan umum dari blok adalah blok kondisional, di mana pernyataan dieksekusi hanya jika kondisi tertentu benar. Misalnya, kita mungkin ingin skrip kita melakukan hal berikut:

```
If (langit berwarna biru)
```

```
{
        pasang string pada anjing;
        ajak anjing jalan-jalan di taman;
}
```

Pernyataan-pernyataan ini diapit kurung kurawal untuk memastikan bahwa mereka dieksekusi sebagai blok. Jika langit berwarna biru, pasang string pada anjing dan ajak anjing berjalan-jalan di taman. Jika langit tidak berwarna biru, tidak ada pernyataan yang dapat dieksekusi (tanpa string maka tidak berjalan). Pernyataan PHP yang menggunakan blok, seperti pernyataan if adalah pernyataan kompleks. PHP membaca seluruh pernyataan kompleks, tidak berhenti pada titik koma pertama yang ditemuinya. Titik koma ini diperlukan, tetapi titik koma tidak diperlukan setelah kurung kurawal akhir.

Sintaks alternatif

Jika diinginkan, kita dapat mengganti kurung kurawal pertama dalam pernyataan sakelar dengan titik dua tunggal, dan kurung kurawal terakhir dengan perintah sakelar ujung. Namun, pendekatan ini tidak umum digunakan dan disebutkan di sini hanya jika kita menemukannya dalam kode pihak ketiga.

Contoh Alternatif switch pernyataan Sintaks

```
<?php
    switch ($page):
    case "Home":
        echo "You selected Home";
        break;
    // etc...
        case "Links":
            echo "You selected Links";
            break;
    endswitch;
?>
```

1.6 MENULIS KODE PHP

Kode PHP harus dibaca oleh manusia, serta oleh perangkat lunak PHP. Skrip PHP ditulis oleh manusia dan harus dimodifikasi, diperbarui, dan dipelihara oleh manusia. Script mungkin perlu dimodifikasi satu atau dua tahun di masa depan ketika programmer asli telah pensiun. Orang yang harus memodifikasi skrip harus bisa membaca dan memahami skrip, yang belum pernah dilihatnya sebelumnya. Akibatnya, kode PHP harus ditulis dengan gaya yang mudah dipahami manusia dengan cepat.

Secara umum, setiap pernyataan sederhana PHP ditulis dalam satu baris yang diakhiri dengan titik koma. Saat menulis blok pernyataan (kompleks), gaya pengkodean menentukan bahwa kita harus membuat indentasi pernyataan blok untuk menunjukkan dengan jelas di mana blok dimulai dan berakhir. Misalnya, dalam contoh pernyataan kondisional berikut, pernyataan sederhana di blok diberi indentasi:

```
If (langit berwarna biru)
{
        pasang string pada anjing;
        ajak anjing jalan-jalan di taman;
}
```

PHP tidak memerlukan indentasi, tetapi membantu manusia membaca kode. Dua gaya yang umum digunakan untuk penempatan kurung kurawal pembuka, sebagai berikut:

```
If (langit berwarna biru)
{
        pasang string pada anjing;
        ajak anjing jalan-jalan di taman;
 }
 If (langit berwarna biru) {
     pasang string pada anjing;
     ajak anjing jalan-jalan di taman;
 }
```

1.7 MENAMPILKAN KONTEN DI HALAMAN WEB

Kita menampilkan konten di halaman web kita dengan pernyataan echo atau print; mereka berdua melakukan hal yang sama. Pernyataan echo atau print menghasilkan output, yang dikirim ke browser pengguna. Bahkan, di mana pun kita melihat echo dalam bab ini, kita juga dapat menulis print. Browser menangani output sebagai HTML. Format umum dari pernyataan echo adalah:

echo outputitem,outputitem,outputitem,…

di mana aturan berikut berlaku:

- Item outputitem dapat berupa angka, string, atau variabel
- Buat daftar outputitem sebanyak yang dibutuhkan, pisahkan dengan koma.

Tabel 1.1 beberapa pernyataan echo dan output.

Pernyataan echo	
Pernyataan echo	**Output**
echo “Hello”;	Hello
echo 123;	123
echo “Hello”,”World!”	HelloWorld
echo Hello World!;	**Tidak valid; menghasilkan pesan kesalahan**
echo “Hello World!”;	Hello World!
echo ‘Hello World!’;	Hello World!

Pernyataan echo dan print menghasilkan baris teks yang dikirim ke browser. Browser menganggap teks sebagai HTML.Sehingga kita perlu memastikan bahwa output kita adalah kode HTML yang valid yang menggambarkan halaman web yang ingin dilihat pengguna. Ketika kita ingin menampilkan halaman web (atau bagian dari halaman web) dengan menggunakan PHP, kita perlu pertimbangkan tiga bagian yang terlibat dalam memproduksi halaman web:

- Skrip PHP: Pernyataan PHP yang kita tulis.
- Source code/Source code HTML: Source code untuk halaman web yang kita lihat saat kita memilih View⇨Source di browser Anda. Source code adalah output dari pernyataan echo atau print.
- Halaman web: Halaman web yang dilihat pengguna Anda. Hasil halaman web dari source code HTML.

Pernyataan echo atau printmengirim persis apa yang kita kirim ke browser — tidak lebih, tidak kurang. Jika kita tidak menggemakan tag HTML apa pun, maka tidak ada yang dikirim.

PHP mengizinkan beberapa karakter khusus yang memformat output, tetapi itu bukan tag HTML. Karakter khusus PHP hanya mempengaruhi output dari pernyataan echo atau print — bukan tampilan di halaman web. Misalnya, jika kita ingin memulai baris baru dalam output PHP atau source code HTML, kita harus menyertakan karakter khusus (\n) yang memberi tahu PHP untuk memulai baris baru. Namun, karakter khusus ini baru saja memulai baris baru di output; itu tidak mengirim tag HTML yang sebenarnya untuk memulai baris baru pada halaman web yang dihasilkan. Tabel dibawah ini menunjukkan contoh dari tiga bagian.

Tabel 1.2 Tiga perbedaan tahapan membuat web dengan PHP

Pernyataan echo	Source Code HTML	Tampilan hal.web
echo “Hello World!”; echo “ ”; echo “Here I am!”;	Hello World! Here I am!”	Hello World! Here I am!
echo “Hello”; echo “ World! \n”; echo “Here I am!”;	Hello World! Here I am!”	Hello World! Here I am!

Tabel di atas merangkum perbedaan tahapan dalam membuat halaman web dengan PHP. Untuk melihat perbedaan ini lebih dekat, perhatikan dua pernyataan print berikut:

```
print “Line 1”;
print “Line 2”;
```

Jika kita meletakkan baris ini dalam skrip, kita mungkin menginginkan halaman web menampilkan ini:

```
Line 1 Line 2
```

Namun, ini bukan output yang akan kita dapatkan. Halaman web akan menampilkan ini:

```
Line 1Line 2
```

Jika kita melihat source code untuk halaman web, kita melihat dengan tepat apa yang dikirim ke browser, yaitu:

```
Line 1Line 2
```

Perhatikan bahwa baris yang dikirim ke browser berisi persis karakter yang kita print — tidak lebih, tidak kurang. String karakter yang kita print tidak mengandung spasi, jadi tidak ada spasi yang muncul di antara baris. Perhatikan juga bahwa dua baris diprint pada baris yang sama. Jika kita ingin memulai baris baru, kita harus mengirim sinyal yang menunjukkan awal baris baru. Untuk memberi tanda bahwa baris baru dimulai di PHP, print karakter khusus \n. Ubah pernyataan print menjadi berikut ini:

```
print “line 1\n”;
print “line 2”;
```

Sekarang kita melihat yang berikut di halaman web:

line 1 line 2

Jika kita melihat source code, kita melihat ini:

```
line 1
line 2
```

Jadi, \n melakukan tugasnya: Ini memulai baris baru di output. Namun, HTML menampilkan output pada halaman web sebagai satu baris. Jika kita ingin HTML menampilkan dua baris, kita harus menggunakan tag, seperti tag
. Jadi, ubah karakter khusus akhir baris PHP menjadi tag HTML, sebagai berikut:

```
print “line 1<br />”;
print “line 2”;
```

Sekarang kita melihat apa yang kita inginkan di halaman web:

```
line 1
line 2
```

Jika kita melihat source code untuk output ini, kita akan melihat ini:

```
line 1<br />line 2
```

Gunakan \n secara bebas. Jika tidak, source code HTML kita akan memiliki beberapa baris yang sangat panjang. Misalnya, jika kita mencetak formulir panjang, semuanya mungkin menjadi satu baris panjang dalam source code, meskipun terlihat baik-baik saja di halaman web. Gunakan \n untuk memecah source code HTML menjadi baris yang masuk akal. Jauh lebih mudah untuk memeriksa dan memecahkan masalah source code jika itu bukan baris yang panjang.

1.8 VARIABEL

Ada metafora sederhana yang akan membantu kita memahami apa itu variabel PHP. Anggap saja mereka sebagai kotak korek api kecil.

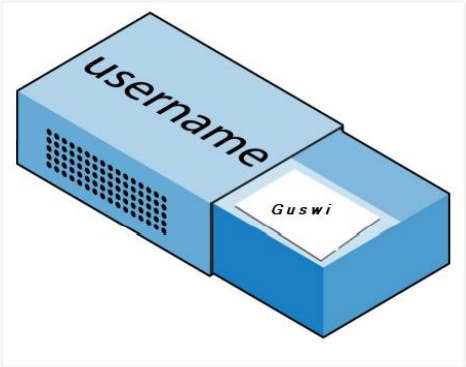
Menggunakan Variabel PHP

Variabel adalah wadah yang digunakan untuk menyimpan informasi. Variabel PHP didefinisikan sedikit berbeda tetapi konsepnya persis sama. Sebuah variabel memiliki nama, dan informasi disimpan dalam variabel. Misalnya, kita dapat memberi nama variabel $age dan menyimpan angka 12 di dalamnya. Informasi yang disimpan dalam variabel dapat digunakan nanti dalam skrip. Salah satu kegunaan paling umum untuk variabel adalah untuk menyimpan informasi yang diketik pengguna ke dalam formulir. Di bagian ini, kita memberi kita detail tentang cara memberi nama dan membuat variabel PHP dengan benar dan menetapkan nilai ke variabel tersebut. Kita juga memberi tahu kita cara menggunakan variabel dinamis dan menampilkan nilai dalam berbagai jenis pernyataan PHP.

Variabel string

Bayangkan kita memiliki kotak korek api di mana kita telah menulis kata nama pengguna. Kita kemudian menulis Guswi pada selembar kertas dan memasukkannya ke dalam kotak. Nah, itu proses yang sama dengan memberikan nilai string ke variabel, seperti ini:

$username = "Guswi"



Gambar 1.1 Kita dapat menganggap variabel sebagai kotak korek api yang berisi item

Tanda kutip menunjukkan bahwa "Guswi" adalah serangkaian karakter. Kita harus menyertakan setiap string dalam tanda kutip atau apostrof (tanda kutip tunggal), meskipun ada perbedaan tipis antara kedua jenis kutipan. Ketika kita ingin melihat apa yang ada di dalam kotak, kita harus membukanya, mengeluarkan selembar kertas, dan membacanya. Di PHP, untuk melakukukan hal tersebut akan terlihat seperti ini:

```
echo $username;
```

Atau kita dapat menetapkannya ke variabel lain (fotokopi kertas dan letakkan salinannya di kotak korek api lain), seperti ini:

```
$current_user = $username;
```

Jika kita ingin mulai mencoba PHP sendiri, kita dapat mencoba memasukkan contoh ke dalam IDE untuk melihat hasil instan, atau kita bisa memasukkan kode ke editor program dan simpan ke direktori root dokumen server And sebagai test1.php

Contoh Program PHP pertama Anda

```
<?php // test1.php
    $username       “Guswi”;
    Echo $username;
    Echo “<br>”;
    Echo $current_user;
?>
```

Sekarang kita dapat memanggilnya dengan memasukkan yang berikut ini ke bilah alamat browser Anda:

```
http://localhost/test1.php
```

Catatan: Jika selama penginstalan server web Anda. Kita mengubah port yang ditetapkan ke server menjadi selain 80, maka kita harus menempatkan nomor port tersebut di dalam URL dalam contoh ini dan semua contoh lain dalam buku ini. Jadi, misalnya, jika kita mengubah port ke 8080, URL sebelumnya menjadi:

```
http://localhost:8080/test1.php
```

Saya tidak akan menyebutkan ini lagi, jadi ingatlah untuk menggunakan nomor port jika diperlukan saat mencoba contoh apa pun atau menulis kode kita sendiri.
Hasil dari menjalankan kode ini harus berupa dua kemunculan nama "Guswi" yang pertama adalah hasil dari perintah echo $username, dan yang kedua dari perintah echo $current_user.

Variabel numerik

Variabel ini tidak hanya berisi string—mereka juga dapat berisi angka. Jika kita kembali ke analogi kotak korek api, untuk menyimpan angka 17 dalam variabel $count, yang setara akan menempatkan, katakanlah, 17 manik-manik dalam kotak korek api di mana kita telah menulis jumlah kata:

```
$count = 17;
```

Kita juga bisa menggunakan angka floating-point (berisi titik desimal); sintaksnya sama:

```
$count = 17.5;
```

Untuk membaca isi kotak korek api, kita cukup membukanya dan menghitung manik-maniknya. Di PHP, kita akan menetapkan nilai $count ke variabel lain atau mungkin hanya menggemakannya ke browser web.

Memberi nama variabel

Saat kita memberi nama variabel, ingatlah aturan berikut:

- **Identifier**: Semua nama variabel memiliki tanda dolar ($) di depannya. Ini memberitahu PHP bahwa itu adalah nama variabel.
- **Awal nam**a: Nama variabel harus dimulai dengan huruf atau garis bawah. Mereka tidak dapat memulai dengan angka.
- **Panjang yang dapat diterima**: Nama variabel dapat memiliki panjan berapa pun.
- **Karakter yang dapat diterima:** Nama variabel hanya dapat menyertakan huruf, angka, dan garis bawah.
- **Sensitivitas huruf besar dan kecil tidak sama.** Misalnya, $firstname dan $Firstname bukan variabel yang sama. Jika kita menyimpan informasi di $firstname, misalnya, kita tidak dapat mengakses informasi tersebut dengan menggunakan nama variabel $firstName.

Saat kita memberi nama variabel, gunakan nama yang memperjelas informasi apa yang ada dalam variabel. Menggunakan nama variabel seperti $var1, $var2, $A, atau $B tidak akan memberikan kejelasan skrip.

Aturan penamaan variabel

Saat membuat variabel PHP, kita harus mengikuti empat aturan berikut:

- Nama variabel harus dimulai dengan huruf alfabet atau karakter _ (garis bawah).
- Nama variabel hanya boleh berisi karakter a–z, A–Z, 0–9, dan _(garis bawah).
- Nama variabel tidak boleh mengandung spasi. Jika sebuah variabel harus terdiri lebih dari satu kata, itu harus dipisahkan dengan karakter _ (garis bawah) (mis., $user_name).
- Nama variabel peka terhadap huruf besar/kecil. Variabel $High_Score tidak sama dengan variabel $high_score.

Catatan: Untuk mengizinkan karakter ASCII yang diperluas yang menyertakan aksen, PHP juga mendukung byte dari 127 hingga 255 dalam nama variabel kecuali kode kita akan dipertahankan hanya oleh pemrogram yang akrab dengan karakter tersebut, mungkin lebih baik untuk menghindarinya, karena pemrogram yang menggunakan keyboard bahasa Inggris akan kesulitan mengaksesnya.

Membuat dan menetapkan nilai ke variabel

Variabel dapat menampung angka atau string karakter. Kita menyimpan informasi dalam variabel dengan satu tanda sama dengan (=). Misalnya, empat pernyataan PHP berikut ditetapkan ke variabel:

```
$age = 12;
$price = 2.55;
$number = -2;
$name = “Little Bo Peep”;
```

Perhatikan bahwa string karakter diapit oleh tanda kutip, tetapi angkanya tidak. Setiap kali kita memasukkan informasi ke dalam variabel yang sebelumnya tidak ada, maka kita telah membuat variabel tersebut. Misalnya, kita menggunakan pernyataan PHP berikut:

```
$firstname = “Agus”;
```

Jika pernyataan ini adalah pertama kalinya kita menyebutkan variabel $firstname, pernyataan ini membuat variabel dan menyetelnya ke "Agus". Jika kita memiliki pernyataan sebelumnya yang mengatur $firstname menjadi “Wibowo”, pernyataan ini mengubah nilai $firstname menjadi “Agus”. Kita juga dapat menghapus informasi dari variabel. Kita dapat melakukan ini untuk menghapus informasi atau untuk menginisialisasi variabel. Misalnya, pernyataan berikut mengambil informasi dari variabel $age:

```
$age = “”;
```

Variabel $age ada tetapi tidak berisi nilai, inibuka berarti bahwa $age disetel ke 0 (nol) karena 0 adalah sebuah nilai. Artinya $age tidak menyimpan informasi apapun. Ini berisi string dengan panjang 0. Kita dapat melangkah lebih jauh dan menghapus variabel dengan menggunakan pernyataan ini.

```
unset($age);
```

Setelah pernyataan ini dijalankan, variabel $age tidak ada lagi

Menggunakan variabele variables

PHP memungkinkan kita untuk menggunakan nama variabel dinamis, yang disebut *variable variables*. Kita dapat memberi nama variabel dengan nilai yang disimpan dalam variabel lain. Artinya, satu variabel berisi nama variabel lain. Misalnya, kita ingin membuat variabel bernama $city dengan nilai Semarang. Kita dapat menggunakan pernyataan berikut:

```
$name_of_the_variable = “kota”;
```

Pernyataan ini membuat variabel yang berisi nama yang ingin kita berikan ke variabel. Kemudian, kita menggunakan pernyataan berikut:

```
$$name_of_the_variable - “Semarang”;
```

Perhatikan karakter tanda dolar ($) tambahan di awal nama variabel. Hal ini menunjukkan variabel variables. Pernyataan ini membuat variabel baru dengan nama yang merupakan nilai dalam $name_of_the_variable, menghasilkan yang berikut:

```
$kota = “Semarang”;
```

Nilai $name_of_the_variable tidak berubah. Contoh selanjutnya menunjukkan cara kerja fitur ini. Dalam bentuknya, pernyataan skrip mungkin tampak tidak begitu berguna; kita mungkin melihat cara yang lebih baik untuk memprogram tugas ini. Nilai sebenarnya dari variabel variables menjadi jelas ketika mereka digunakan dengan array dan loop.

Misalkan kita ingin memberi nama serangkaian variabel dengan nama kota. Kita dapat menggunakan kode ini

```
$Ungaran = 360000;
$Semarang = 138000;
$cityname = “Ungaran”;
echo “The size of $cityname is ${$cityname}”;
$cityname = “Semarang”;
echo “The size of $cityname is ${$cityname}”;
```

Output dari kode ini adalah:

```
The size of Ungaran is 360000
The size of Semarang is 138000
```

Perhatikan bahwa kita perlu menggunakan kurung kurawal di sekitar nama variabel dalam pernyataan echo sehingga PHP tahu di mana nama variabelnya. Jika kita menggunakan pernyataan tanpa kurung kurawal, hasilnya adalah sebagai berikut:

```
The size of Ungaran is $Ungaran
```

Tanpa kurung kurawal di $$cityname, PHP mengonversi $cityname menjadi nilai selanjutnya letakkan $ tambahan di depannya, sebagai bagian dari string sebelumnya

Menampilkan nilai variabel

Kita dapat menampilkan nilai dalam variabel dengan menggunakan salah satu pernyataan berikut:

- echo
- print
- print_r
- var_dump

Menggunakan variabel dalam pernyataan echo dan print

Kita dapat menampilkan nilai dalam variabel pada halaman web dengan pernyataan echo atau print. Misalnya, jika kita menyetel variabel $usia ke 12 dan kemudian menggunakan pernyataan echo PHP berikut di bagian PHP, hasilnya adalah 12.

```
echo $age;
```

Jika kita menyertakan baris berikut dalam file HTML:

```
<p>Usia kita <?php echo $age ?>.</p>
```

output pada halaman web adalah:

Usia kita adalah 12 tahun.

Tabel dibawah ini menunjukkan penggunaan variabel dalam beberapa pernyataan print dan outputnya. Untuk keperluan tabel, asumsikan bahwa $string1 diset ke Hello dan $string2 diset ke World!.

Tabel 1.3 Beberapa contoh penggunaan print dan outputnya

Pernyataan Print	Output
print $string1;	Hello
print $string1,$string2;	HelloWorld!
print “$string1 $string2”;	Hello World!
print “Hello “,$string2;	Hello World!
print “Hello”,” “,$string2;	Hello World!
print ‘$string1’,”$string2”;	$String1World!

Tanda kutip tunggal dan ganda memiliki efek yang berbeda pada variabel, sebagai berikut.

- **Tanda kutip tunggal (‘ ‘):** Saat kita menggunakan tanda kutip tunggal, nama variabel digaungkan apa adanya.
- **Tanda kutip ganda (“”):** Saat kita menggunakan tanda kutip ganda, nama variabel diganti dengan nilai variabel.

Terkadang kita perlu menyertakan nama variabel dalam kurung kurawal ({}) untuk menentukan nama variabel. Misalnya, pernyataan berikut tidak akan menampilkan burung sebagai variabel $pet.

```
$pet = “burung”;
echo “$petcage telah tiba.”;
```

Dengan kata lain, hasilnya tidak akan menjadi burung telah tiba. Sebaliknya, PHP akan mencari variabel $petcage dan tidak akan dapat menemukannya. Kita dapat menggemakan output yang benar dengan menggunakan kurung kurawal untuk memisahkan variabel $pet:

```
$pet = “burung”;
echo “petcage {$pet}telah tiba.”;
```

Pernyataan sebelumnya memberi Anda

Sangkar burung telah tiba.

Mengetahui berapa lama suatu variabel mempertahankan nilainya

Variabel menyimpan informasinya untuk seluruh skrip, bukan hanya untuk satu bagian PHP. Jika sebuah variabel disetel ke "ya" di awal file, variabel itu akan tetap menahan "ya" di akhir halaman. Misalnya, file kita memiliki pernyataan berikut:

```
<p>Hello World!</p>
<?php
      $age = 15;
      $name = “Vino”;
?>
<p>Hello World again!</p>
```

```
<?php
    echo $name;
?>
```

Pernyataan echo di bagian PHP kedua akan menampilkan Vino. Halaman web yang dihasilkan dari pernyataan ini adalah:

```
Hello World!
Hello World again!
Vino
```

Menampilkan variabel dengan pernyataan print_r

PHP menyediakan fungsi bernama print_r untuk melihat nilai dalam sebuah variabel. Kita dapat menulis pernyataan berikut untuk menampilkan nilai variabel:

```
$weekday - “Senin”;
print_r($weekday);
```

Output dari print_r adalah:

```
Senin
```

Menampilkan variabel dengan pernyataan var_dump

PHP menyediakan fungsi bernama var_dump yang dapat digunakan untuk menampilkan nilai variabel dan tipe datanya. Kita dapat menulis pernyataan berikut untuk menampilkan nilai variabel:

```
$weekday = “Senin”;
var_dump($weekday);
```

Output dari var_dump adalah:

```
string(6) “Senin”
```

Output menunjukkan bahwa nilai dalam $weekday adalah Senin. Outputnya juga menunjukkan bahwa nilainya adalah tipe data string yang panjangnya enam karakter. Kita dapat menggunakan var_dump untuk pemecahan masalah PHP. Penggunaannya sangat penting untuk tujuan itu.

1.9 OPERATOR

Operator adalah perintah matematika, string, perbandingan, dan logika seperti plus, minus, perkalian, dan pembagian. PHP sangat mirip dengan aritmatika biasa; misalnya, pernyataan berikut menghasilkan 8:

```
echo 6 + 2;
```

Operator aritmatika

Operator aritmatika digunakan untuk melakukan perhitungan matematika. Kita dapat menggunakannya untuk empat operasi utama (plus, minus, perkalian, dan pembagian) serta

untuk menemukan modulus (sisa setelah pembagian) dan untuk menambah atau mengurangi nilai.

Tabel 1.4 operator aritmatika

Operator	*Deskripsi*	*Contoh*
+	*Tambahan*	$j+1
-	*Pengurangan*	$j-6
***	*Pembagian*	$j*11
/	*Perkalian*	$j/4
%	*Modulus (sisa pembagian)*	$j%9
++	*Kenaikan*	++$j
--	*Pengurangan*	--$j

Operator penugasan

Operator ini digunakan untuk memberikan nilai ke variabel. Mereka mulai dengan yang sangat sederhana = dan melanjutkan ke +=, =, dan seterusnya. Operator += menambahkan nilai di sisi kanan ke variabel di sebelah kiri. Jadi, jika $count dimulai dengan nilai 5, pernyataan:

```
$count += 1;
```

set $count ke 6,

seperti pernyataan penugasan yang lebih familiar:

```
$count = $count + 1;
```

String memiliki operatornya sendiri, titik (.), yang dirinci di bagian Penggabungan string.

Tabel 1.5 Operator Penugasan 5

Operator	*Deskripsi*	*Contoh*
=	$j = 15	$j = 15
+=	$j += 5	$j = $j + 5
-=	$j -= 3	$j = $j – 3
***=*	$j *= 8	$j = $j * 8
/=	$j /= 16	$j = $j / 16
.=	$j .= $k	$j = $j . $k
%=	$j %= 4	$j = $j % 4

Operator perbandingan

Operator perbandingan umumnya digunakan di dalam konstruk seperti pernyataan if di mana kita perlu membandingkan dua item. Misalnya, kita mungkin ingin mengetahui apakah variabel yang telah kita tambahkan telah mencapai nilai tertentu, atau apakah variabel lain kurang dari nilai yang ditetapkan, dan seterusnya.

Perhatikan perbedaan antara = dan ==. Yang pertama adalah operator penugasan, dan yang kedua adalah operator perbandingan. Bahkan programmer yang lebih mahir terkadang dapat mengubah keduanya saat coding dengan tergesa-gesa, jadi berhati-hatilah.

Tabel 1.6 Operator perbandingan

Operator	*Deskripsi*	*Contoh*
==	*Adalah sama dengan*	$j == 4
!=	*Tidak sama dengan*	$j != 21
>	*Lebih besar dari*	$j > 3
<	*Kurang dari*	$j < 100
>=	*Lebih besar atau sama dengan*	$j >= 15
<=	*Kurang dari atau sama dengan*	$j <= 8

Operator logika

Sebagai aturan, jika sesuatu memiliki nilai TRUE atau FALSE, itu dapat dimasukkan ke operator logika. Operator logika mengambil dua input benar atau salah dan menghasilkan hasil yang benar atau salah.

Tabel 1.7 Operator Logikal

Operator	*Deskripsi*	*Contoh*
&&	**Dan**	$j == 3 && $k ==2
and	Prioritas rendah **dan**	$j == 3 and $k == 2
||	**Atau**	$j < 5 || $j > 10
or	Prioritas rendah **atau**	$j < 5 or $j > 10
!	**Tidak**	! ($j == $k)
xor	**Ekslusif atau**	$j xor $k

Perhatikan bahwa && biasanya dapat dipertukarkan dengan and; hal yang sama berlaku untuk || dan or. Tetapi and dan or memiliki prioritas yang lebih rendah, jadi dalam beberapa kasus, kita mungkin memerlukan tanda kurung tambahan untuk memaksakan prioritas yang diperlukan. Di sisi lain, ada kalanya hanya and atau or dapat diterima, seperti pada pernyataan berikut, yang menggunakan operator or:

```
mysql_select_db($database) or die("Unable to select database");
```

Operator paling tidak biasa dari operator ini adalah xor, yang berarti eksklusif atau dan mengembalikan nilai TRUE jika salah satu nilainya TRUE, tetapi nilai FALSE jika kedua inputnya TRUE atau kedua inputnya FALSE. Untuk memahami hal ini, bayangkan kita ingin membuat pembersih sendiri untuk barang-barang rumah tangga. Amonia merupakan pembersih yang baik, begitu juga bleaching, jadi kita ingin pembersih kita memiliki salah satunya. Tetapi pembersih tidak boleh memiliki keduanya, karena kombinasinya berbahaya. Di PHP, kita bisa merepresentasikan ini sebagai:

```
$ingredient = $ammonia xor $bleach;
```

Dalam cuplikan contoh, jika $amonia atau $bleach adalah TRUE, $ingredient juga akan disetel ke TRUE. Tetapi jika keduanya TRUE atau keduanya FALSE, $ingredient akan disetel ke FALSE.

Penugasan Variabel

Sintaks untuk menetapkan nilai ke variabel selalu *variabel = nilai*. Atau, untuk menetapkan kembali nilai ke variabel lain, itu adalah *variabel lain = variabel*. Ada juga beberapa operator penugasan lain yang berguna. Misalnya, kita sudah melihat:

```
$x += 10
```

yang memberitahu parser PHP untuk menambahkan nilai di sebelah kanan (dalam hal ini, nilai 10) ke variabel $x. Demikian juga, kita dapat mengurangi sebagai berikut:

```
$y -= 10;
```

Penambahan dan pengurangan variabel

Menambah atau mengurangi 1 adalah operasi yang umum, PHP menyediakan operator khusus untuk itu. Kita dapat menggunakan salah satu dari yang berikut ini sebagai pengganti operator += dan =:

```
++$x;
--$y;
```

Dalam hubungannya dengan tes (pernyataan if), kita dapat menggunakan kode berikut:

```
if (++$x == 10) echo $x;
```

Ini memberitahu PHP untuk terlebih dahulu menaikkan nilai $x dan kemudian menguji apakah nilainya 10; jika ya, keluarkan nilainya. Tetapi kita juga dapat meminta PHP untuk menaikkan (atau, dalam contoh berikut, mengurangi) variabel setelah nilai diuji, seperti ini:

```
if ($y-- == 0) echo $y;
```

yang memberikan hasil yang agak berbeda. Misalkan $y dimulai sebagai 0 sebelum pernyataan dieksekusi. Perbandingan akan mengembalikan hasil yang TRUE, tetapi $y akan disetel ke 1 setelah perbandingan dibuat. Jadi apa yang akan ditampilkan pernyataan echo: 0 atau 1? Coba tebak, lalu cobalah pernyataan prosesor PHP untuk mengonfirmasi. Karena kombinasi pernyataan ini membingungkan, ini harus dianggap hanya sebagai contoh pendidikan dan bukan sebagai panduan untuk gaya pemrograman yang baik. Singkatnya, apakah suatu variabel bertambah atau berkurang sebelum atau sesudah pengujian tergantung pada apakah operator kenaikan atau penurunan ditempatkan sebelum atau sesudah variabel.

Omong-omong, jawaban yang benar untuk pertanyaan sebelumnya adalah pernyataan echo akan menampilkan hasil 1, karena $y dikurangi tepat setelah diakses dalam pernyataan if, dan sebelum pernyataan echo.

Penggabungan string

Penggabungan string menggunakan titik (.) untuk menambahkan satu string karakter ke string lainnya. Cara paling sederhana untuk melakukannya adalah sebagai berikut:

```
echo "Anda memiliki " . $psn . " pesan.";
```

Dengan asumsi bahwa variabel $msgs diatur ke nilai 5, output dari baris kode ini adalah:

```
Anda memiliki 5 pesan.
```

Sama seperti kita dapat menambahkan nilai ke variabel numerik dengan operator +=, kita dapat menambahkan satu string ke string lainnya menggunakan .= seperti ini:

```
$bulletin .= $newsflash;
```

Dalam hal ini, jika $bulletin berisi buletin berita dan $newsflash memiliki flash berita, perintah menambahkan flash berita ke buletin berita sehingga $bulletin sekarang terdiri dari kedua string teks.

Tipe string

PHP mendukung dua jenis string yang dilambangkan dengan jenis tanda kutip yang kita gunakan. Jika kita ingin menetapkan string literal, mempertahankan konten yang tepat, kita harus menggunakan tanda kutip tunggal (apostrof) seperti ini:

```
$info = 'Preface variables with a $ like this: $variable';
```

Dalam hal ini, setiap karakter dalam string yang dikutip tunggal ditugaskan ke $info. Jika kita menggunakan tanda kutip ganda, PHP akan mencoba mengevaluasi $variable sebagai variabel. Di sisi lain, ketika kita ingin memasukkan nilai variabel di dalam string, kita melakukannya dengan menggunakan string bertanda kutip ganda:

```
echo "This week $count people have viewed your profile";
```

Seperti yang akan kita sadari, sintaks ini juga menawarkan bentuk penggabungan yang lebih sederhana di mana kita tidak perlu menggunakan titik, atau menutup dan membuka kembali tanda kutip untuk menambahkan satu string ke string lainnya. Ini disebut substitusi variabel, dan kita akan melihat beberapa aplikasi menggunakannya secara ekstensif.

Karakter arti khusus

Terkadang string perlu mengandung karakter dengan arti khusus yang mungkin diinterpretasikan secara tidak benar. Misalnya, baris kode berikut tidak akan berfungsi, karena tanda kutip kedua yang ditemukan dalam ejaan kata akan memberi tahu parser PHP bahwa akhir string telah tercapai. Akibatnya, sisa baris akan ditolak sebagai kesalahan:

```
$text = 'My spelling's atroshus'; // Erroneous syntax
```

Untuk memperbaikinya, kita dapat menambahkan garis miring terbalik langsung sebelum tanda kutip yang menyinggung untuk memberi tahu PHP agar memperlakukan karakter secara harfiah dan tidak menafsirkannya:

```
$text = 'My spelling\'s still atroshus';
```

Dan kita dapat melakukan trik ini di hampir semua situasi di mana PHP sebaliknya akan mengembalikan kesalahan dengan mencoba menafsirkan karakter. Misalnya, string yang dikutip ganda berikut akan ditetapkan dengan benar:

```
$text = "She wrote upon it, \"Return to sender\".";
```

Selain itu, kita dapat menggunakan karakter escape untuk menyisipkan berbagai karakter khusus ke dalam string seperti tab, baris baru, dan carriage return. Ini diwakili, seperti yang kita duga, oleh \t, \n, dan \r. Berikut adalah contoh penggunaan tab untuk meletakkan heading; itu disertakan di sini hanya untuk mengilustrasikan pelarian, karena di halaman web selalu ada cara yang lebih baik untuk melakukan tata letak:

```
$heading = "Date\tName\tPayment";
```

Karakter khusus yang didahului dengan garis miring terbalik ini hanya berfungsi dalam string bertanda kutip ganda. Dalam string yang dikutip tunggal, string sebelumnya akan ditampilkan dengan urutan \t jelek alih-alih tab. Dalam string yang dikutip tunggal, hanya tanda kutip yang lolos (\') dan garis miring terbalik yang lolos itu sendiri (\\) yang dikenali sebagai karakter yang lolos.

Perintah Beberapa Baris

Ada kalanya kita perlu mengeluarkan cukup banyak teks pada PHP, dan menggunakan beberapa pernyataan echo (atau print) akan memakan waktu dan berantakan. Untuk mengatasinya, PHP menawarkan dua kemudahan. Yang pertama adalah hanya menempatkan beberapa baris di antara tanda kutip.

Contoh pernyataan echo string multiline

```
<?php
$author = "Agus";
echo "Developers, Developers, developers, developers, developers,
developers, developers, developers, developers!
- $author.";
?>
```

Contoh tugas string multiline

```
<?php
    $author = "Wibowo";
    $text = " Mengukur kemajuan pemrograman dengan baris kode seperti
    mengukur kemajuan pembangunan pesawat berdasarkan berat.
    - $author.";
?>
```

PHP juga menawarkan urutan multiline menggunakan operator <<<—biasa disebut sebagai dokumen atau heredoc—sebagai cara untuk menentukan literal string, mempertahankan jeda baris dan spasi putih lainnya (termasuk indentasi) dalam teks.

Contoh pernyataan echo multibaris alternatif

```
<?php
    $author = "Josh";
    echo <<<_END
    Debugging dua kali lebih sulit daripada menulis kode di tempat pertama.
    Oleh karena itu, jika kita menulis kode sepandai mungkin,
    menurut definisi, kita tidak cukup pintar untuk men-debug-nya..
    - $author.";
_END;
```

```
?>
```

Kode ini memberi tahu PHP untuk menampilkan semua yang ada di antara dua tag _END seolah-olah itu adalah string bertanda kutip ganda (kecuali bahwa tanda kutip dalam heredoc tidak perlu diloloskan). Misalnya, bagi pengembang untuk menulis seluruh bagian HTML secara langsung ke dalam kode PHP dan kemudian hanya mengganti bagian dinamis tertentu dengan variabel PHP.

Penting untuk diingat bahwa penutup _END; tag harus muncul tepat di awal baris baru dan itu harus menjadi satu-satunya di baris itu—bahkan komentar tidak boleh ditambahkan setelahnya (atau bahkan satu spasi pun). Setelah kita menutup blok multiline, kita bebas menggunakan nama tag yang sama lagi.

Catatan: menggunakan <<<_END ... _END; heredoc, kita tidak perlu menambahkan \n karakter linefeed untuk mengirim linefeed—cukup tekan Return dan mulai baris baru. Selain itu, tidak seperti string yang dipisahkan oleh tanda kutip ganda atau tanda kutip tunggal, kita bebas menggunakan semua tanda kutip tunggal dan ganda yang kita suka di dalam heredoc, tanpa menghindarinya dengan mendahuluinya dengan garis miring (\).

Contoh penugasan variabel string multiline

```
<?php
    $author = "Caroline"
    $out = <<<_END
    Orang normal percaya bahwa jika tidak rusak, jangan memperbaikinya.
    Insinyur percaya bahwa jika tidak rusak, fitur belum sempurna.
    - $author.
_END;
?>
```

Variabel $out kemudian akan diisi dengan konten di antara dua tag. Jika kita menambahkan, bukan menugaskan, kita juga bisa menggunakan .= sebagai pengganti = untuk menambahkan string ke $out. Berhati-hatilah untuk tidak menempatkan titik koma secara langsung setelah kemunculan pertama _END, karena hal itu akan menghentikan blok multibaris bahkan sebelum dimulai dan menyebabkan pesan kesalahan Parse. Satu-satunya tempat untuk titik koma adalah setelah tag _END yang mengakhiri, meskipun aman untuk menggunakan titik koma di dalam blok sebagai karakter teks biasa.

Tag _END hanyalah salah satu yang saya pilih sebagai contoh, ini karena tidak mungkin digunakan di tempat lain dalam kode PHP. Kita dapat menggunakan tag apa pun yang kita suka, seperti _SECTION1 atau _OUTPUT dan seterusnya. Selain itu, untuk membantu membedakan tag seperti ini dari variabel atau fungsi, praktik umumnya adalah mengawalinya dengan garis bawah, tetapi kita tidak harus menggunakannya jika memilih untuk tidak menggunakannya.

Catatan: Meletakkan teks di atas beberapa baris biasanya hanya untuk memudahkan agar kode PHP kita lebih mudah dibaca, karena setelah ditampilkan di halaman web, aturan pemformatan HTML akan mengambil alih dan spasi akan dihilangkan (namun $author tetap diganti dengan nilai variabel). Jadi, jika kita memuat contoh output multibaris ini ke dalam browser, contoh tersebut tidak akan ditampilkan pada beberapa baris, karena semua browser memperlakukan baris baru seperti spasi. Namun, jika kita menggunakan fitur sumber tampilan browser, kita akan menemukan bahwa baris baru ditempatkan dengan benar, dan output muncul di beberapa baris.

Pengetikan Variabel

PHP adalah bahasa yang sangat longgar untuk diketik. Ini memiliki arti bahwa variabel tidak harus dideklarasikan sebelum digunakan, dan PHP selalu mengonversi variabel ke tipe yang dibutuhkan oleh konteksnya saat diakses. Misalnya, kita dapat membuat angka multi-digit dan mengekstrak digit ke-n darinya hanya dengan menganggapnya sebagai string. Dalam potongan kode berikut, angka 12345 dan 67890 dikalikan bersama, menghasilkan hasil 838102050, yang kemudian ditempatkan dalam variabel $number.

Contoh konversi otomatis dari angka ke string

```
<?php
    $number = 12345 * 67890;
    echo substr($number, 3, 1);
?>
```

Pada titik penugasan, $number adalah variabel numerik. Tetapi pada baris kedua, panggilan ditempatkan ke substr fungsi PHP, yang meminta satu karakter untuk dikembalikan dari $number, mulai dari posisi keempat (mengingat bahwa offset PHP dimulai dari nol). Untuk melakukan ini, PHP mengubah $number menjadi string sembilan karakter, sehingga substr dapat mengaksesnya dan mengembalikan karakter, yang dalam hal ini adalah 1.

Hal yang sama berlaku untuk mengubah string menjadi angka, dan seterusnya. Dalam Contoh dibawah ini, variabel $pi diatur ke nilai string, yang kemudian secara otomatis diubah menjadi angka floating-point di baris ketiga dengan persamaan untuk menghitung luas lingkaran, yang menghasilkan nilai 78.5398175.

Contoh mengonversi string menjadi angka secara otomatis

```
<?php
    $pi = "3.1415927";
    $radius = 5;
    echo $pi * ($radius * $radius);
?>
```

Dalam praktiknya, kita tidak perlu terlalu khawatir tentang tipe variabel Anda. Tetapkan saja nilai yang masuk akal bagi kita dan PHP akan mengonversinya. Kemudian, ketika kita ingin mengambil nilai, gunakan pernyataan echo.

1.10 MENGGUNAKAN KONSTANTA PHP

Konstanta PHP mirip dengan variabel. Konstanta diberi nama, dan nilai disimpan di dalamnya. Namun, konstanta tidak dapat diubah oleh skrip. Setelah kita menetapkan nilai untuk sebuah konstanta, maka nilainya akan tetap sama. Jika kita menggunakan konstanta untuk usia dan menyetelnya ke 21, maka nilainya akan selalu dan selamanya 21.

Konstanta digunakan ketika nilai diperlukan di beberapa tempat dalam skrip dan tidak berubah. Nilai diatur dalam konstanta di awal skrip. Dengan menggunakan konstanta di seluruh skrip, nilainya tidak akan berubah secara tidak sengaja. Dengan memberinya nama, kita akan mengetahui informasinya secara instan. Dan dengan menyetel konstanta sekali di awal skrip saja kita dapat mengubah nilai konstanta di satu tempat jika diperlukan tanpa harus mencari nilai di banyak tempat di skrip.

Konstanta menyimpan informasi untuk diakses, dengan kata lain, setelah kita mendefinisikannya, nilainya akan ditetapkan hingga akhir program dan tidak dapat diubah. Salah satu contoh penggunaan konstanta mungkin untuk menahan lokasi root server Ada (folder dengan file utama situs web). Konstanta akan didefinisikan seperti ini:

```
define("ROOT_LOCATION", "usrlocal/www/");
```

Kemudian, untuk membaca isi variabel, kita cukup merujuknya seperti variabel biasa (tetapi tidak didahului dengan tanda dolar):

```
$directory = ROOT_LOCATION;
```

Misalnya, kita dapat menetapkan satu konstanta yang merupakan nama perusahaan dan konstanta lain yang merupakan alamat perusahaan dan menggunakannya di mana pun diperlukan. Kemudian, jika perusahaan pindah, kita bisa mengubah nilai konstanta alamat perusahaan di awal skrip alih-alih harus mencari dan mengubah setiap tempat di skrip kita yang menggemakan nama perusahaan.

Kita dapat mengatur konstanta dengan menggunakan pernyataan define. Formatnya adalah:

```
define("constantname", "constantvalue");
```

Misalnya, untuk menetapkan konstanta dengan nama perusahaan, gunakan pernyataan berikut:

```
define("PERUSAHAAN", "Perusahaan Saya yang bagus");
```

Gunakan konstanta dalam skrip kita di mana pun kita membutuhkan nama perusahaan Anda:

```
echo PERUSAHAAN;
```

Kita dapat menggunakan nama apa pun untuk konstanta yang dapat kita gunakan untuk variabel, selama kita mengikuti konvensi ini:

- Tidak ada pengenal: Nama konstan tidak didahului dengan tanda dolar ($).
- Kasus: Berdasarkan konvensi, konstanta diberi nama yang semuanya huruf besar, sehingga kita dapat dengan mudah melihat konstanta, tetapi PHP sendiri tidak peduli apa yang kita beri nama konstanta. Kita tidak harus menggunakan huruf besar; itu lebih jelas.
- Karakter: kita dapat menyimpan string atau angka di dalamnya. Pernyataan berikut ini benar-benar oke dengan PHP:
  define (“AGE”,29);

Sekarang, setiap kali perlu menjalankan kode PHP di server yang berbeda dengan konfigurasi folder yang berbeda, kita hanya memiliki satu baris kode untuk diubah. Umumnya dianggap sebagai praktik yang baik untuk menggunakan huruf besar pada nama variabel konstan, terutama jika orang lain akan membaca kode Anda. Catatan: Dua hal utama yang harus kita ingat tentang konstanta adalah bahwa konstanta tidak boleh diawali dengan $ (seperti variabel biasa), dan kita dapat mendefinisikannya hanya dengan menggunakan fungsi define.

Konstanta yang telah ditentukan sebelumnya

PHP hadir siap pakai dengan lusinan konstanta standar yang biasanya tidak akan kita gunakan sebagai pemula di PHP. Namun, ada beberapa—dikenal sebagai konstanta ajaib—yang menurut kita berguna. Nama-nama konstanta ajaib selalu memiliki dua garis bawah di awal dan dua di akhir, sehingga kita tidak akan secara tidak sengaja mencoba memberi nama salah satu konstanta kita sendiri dengan nama yang sudah diambil.

Tabel 1.8 Konstanta Magic PHP

Konstan Magic	Deskripsi
LINE	Nomor baris file saat ini.
FILE	Path lengkap dan nama file dari file tersebut. Jika digunakan di dalam include, nama file yang disertakan akan dikembalikan. Di PHP 4.0.2, __FILE__ selalu berisi jalur absolut dengan tautan simbolik diselesaikan, sedangkan di versi yang lebih lama mungkin berisi jalur relatif dalam beberapa keadaan.
DIR	Direktori file. Jika digunakan di dalam sebuah include, direktori dari file yang disertakan akan dikembalikan. Ini setara dengan dirname(__FILE__). Nama direktori ini tidak memiliki garis miring kecuali direktori root. (Ditambahkan dalam PHP 5.3.0.)
FUNCTION	Nama fungsi. (Ditambahkan dalam PHP 4.3.0.) Pada PHP 5, mengembalikan nama fungsi seperti yang dideklarasikan (peka huruf besar-kecil). Di PHP 4, nilainya selalu huruf kecil.
CLASS	Nama kelas. (Ditambahkan dalam PHP 4.3.0.) Pada PHP 5, mengembalikan nama kelas seperti yang dideklarasikan (peka huruf besar-kecil). Di PHP 4, nilainya selalu huruf kecil.
METHOD	Nama metode kelas. (Ditambahkan dalam PHP 5.0.0.) Nama metode dikembalikan seperti yang dideklarasikan (peka huruf besar-kecil).
NAMESPACE	Nama namespace saat ini (peka huruf besar/kecil). Konstanta ini didefinisikan pada waktu kompilasi. (Ditambahkan dalam PHP 5.3.0.)

Salah satu kegunaan praktis dari variabel-variabel ini adalah untuk tujuan debugging, ketika kita perlu memasukkan sebaris kode untuk melihat apakah aliran program mencapainya:

```
echo "This is line " . __LINE__ . " of file " . __FILE__
```

Ini menyebabkan baris program saat ini dalam file saat ini (termasuk jalur) yang sedang dieksekusi menjadi output ke browser web.

1.11 PERBEDAAN ANTARA PERINTAH ECHO DAN PRINT

Sejauh ini, kita telah melihat perintah echo digunakan dalam beberapa cara berbeda untuk menampilkan teks dari server ke browser Anda. Dalam beberapa kasus, string literal telah dikeluarkan. Di tempat lain, string pertama kali digabungkan atau variabel telah dievaluasi. Saya juga telah menunjukkan output yang tersebar di beberapa baris. Tetapi ada juga alternatif untuk echo yang dapat kita gunakan: print. Kedua perintah tersebut sangat mirip, tetapi print adalah konstruksi seperti fungsi yang mengambil parameter tunggal dan memiliki nilai kembalian (yang selalu 1), sedangkan echo murni merupakan konstruksi bahasa PHP. Karena kedua perintah adalah konstruksi, keduanya tidak memerlukan penggunaan tanda kurung.

Pada umumnya, perintah echo akan sedikit lebih cepat daripada mencetak dalam output teks umum, karena tidak menetapkan nilai balik. Di sisi lain, karena tidak diimplementasikan seperti fungsi, echo tidak dapat digunakan sebagai bagian dari ekspresi yang lebih kompleks, sedangkan print bisa. Berikut adalah contoh untuk menampilkan apakah nilai variabel adalah TRUE atau FALSE menggunakan print, sesuatu yang tidak dapat kita lakukan dengan cara yang sama dengan echo, karena akan menampilkan pesan kesalahan Parse:

```
$b ? print "TRUE" : print "FALSE"
```

Tanda tanya hanyalah cara untuk menginterogasi apakah variabel $b adalah TRUE atau FALSE. Perintah mana pun yang ada di sebelah kiri titik dua berikut dijalankan jika $b adalah TRUE, sedangkan perintah di sebelah kanan dijalankan jika $b adalah FALSE. Namun, secara umum, contoh-contoh dalam buku ini menggunakan echo, dan saya menyarankan kita melakukannya juga sampai kita mencapai titik dalam pengembangan PHP kita sehingga kita menemukan kebutuhan untuk menggunakan print.

1.12 FUNGSI

Fungsi digunakan untuk memisahkan bagian kode yang melakukan tugas tertentu. Misalnya, mungkin kita sering perlu mencari tanggal dan mengembalikannya dalam format tertentu. Itu akan menjadi contoh yang baik untuk berubah menjadi suatu fungsi. Kode yang melakukannya mungkin hanya tiga baris panjangnya, tetapi jika kita harus menempelkannya ke program kita belasan kali, kita membuat program kita tidak perlu besar dan rumit, kecuali jika kita menggunakan suatu fungsi. Dan jika kita memutuskan untuk mengubah format data nanti, memasukkannya ke dalam suatu fungsi berarti harus mengubahnya hanya di satu tempat. Menempatkannya ke dalam suatu fungsi tidak hanya mempersingkat source code kita dan membuatnya lebih mudah dibaca, tetapi juga menambahkan fungsionalitas tambahan (permainan kata-kata), karena fungsi dapat melewati parameter untuk membuatnya bekerja secara berbeda. Mereka juga dapat mengembalikan nilai ke kode panggilan.

Contoh deklarasi fungsi sederhana

```
<?php
    function longdate($timestamp)
    {
        return date("l F jS Y", $timestamp);
    }
?>
```

Fungsi ini menggunakan timestamp Unix (angka integer yang mewakili tanggal dan waktu berdasarkan jumlah detik sejak pukul 00:00 pada 1 Januari 1970) sebagai inputnya dan kemudian memanggil fungsi tanggal PHP dengan format string yang benar untuk mengembalikan a tanggal dalam format Selasa 2 Mei 2017. Sejumlah parameter dapat dilewatkan di antara tanda kurung awal; kita telah memilih untuk menerima hanya satu. Kurung kurawal mengapit semua kode yang dijalankan saat kita nanti memanggil fungsi tersebut. Untuk menampilkan tanggal hari ini menggunakan fungsi ini, lakukan panggilan berikut dalam kode Anda:

```
echo longdate(time());
```

Panggilan ini menggunakan fungsi PHP time bawaan untuk mengambil timestamp Unix saat ini dan meneruskannya ke fungsi tanggal panjang yang baru, yang kemudian mengembalikan string yang sesuai ke perintah echo untuk ditampilkan. Jika kita perlu mencetak tanggal 17 hari yang lalu, kita hanya perlu mengeluarkan panggilan berikut:

```
echo longdate(time() - 17 * 24 60 60);
```

yang lolos ke longdate timestamp Unix saat ini dikurangi jumlah o detik sejak 17 hari yang lalu (17 hari × 24 jam × 60 menit × 60 detik). Fungsi juga dapat menerima beberapa parameter dan mengembalikan beberapa hasil.

1.13 LINGKUP VARIABEL

Jika kita memiliki program yang sangat panjang, sangat mungkin kita mulai kehabisan nama variabel yang bagus, tetapi dengan PHP kita dapat memutuskan cakupan variabel. Dengan kata lain, kita dapat, misalnya, mengatakan bahwa kita ingin variabel $temp hanya digunakan di dalam fungsi tertentu dan melupakan bahwa variabel itu pernah digunakan saat fungsi kembali. Sebenarnya, ini adalah cakupan default untuk variabel PHP. Atau, kita dapat memberi tahu PHP bahwa suatu variabel memiliki cakupan global dan dengan demikian dapat diakses oleh setiap bagian lain dari program Anda.

Variabel lokal

Variabel lokal adalah variabel yang dibuat di dalam, dan hanya dapat diakses oleh suatu fungsi. Pada umumnya ini adalah variabel sementara yang digunakan untuk menyimpan hasil yang sebagian diproses sebelum kembalinya fungsi. Satu set variabel lokal adalah daftar argumen ke suatu fungsi.

Contoh versi yang diperluas dari fungsi tanggal panjang

```
<?php
function longdate($timestamp)
{
    $temp = date("l F jS Y", $timestamp);
    return "The date is $temp";
    }
?>
```

Di sini kita telah menetapkan nilai yang dikembalikan oleh fungsi tanggal ke variabel sementara $temp, yang kemudian dimasukkan ke dalam string yang dikembalikan oleh fungsi tersebut. Segera setelah fungsi kembali, nilai $temp dihapus, seolah-olah tidak pernah digunakan sama sekali.

Contoh upaya ini untuk mengakses* $temp *dalam fungsi longdate akan gagal

```
<?php
    $temp = "The date is ";
    echo longdate(time());
    function longdate($timestamp)
    {
        return $temp . date("l F jS Y", $timestamp);
    }
?>
```

Namun, karena $temp tidak dibuat dalam fungsi longdate atau diteruskan sebagai parameter, longdate tidak dapat mengaksesnya. Sehingga cuplikan kode ini hanya menampilkan tanggal, bukan teks sebelumnya. Bahkan, pertama kali akan menampilkan pesan kesalahan *Notice: Undefined variable: temp.* Alasan untuk ini, secara default, variabel yang dibuat dalam suatu fungsi bersifat lokal untuk fungsi itu, dan variabel yang dibuat di luar fungsi apa pun hanya dapat diakses oleh kode non-fungsi.

Contoh menulis ulang untuk merujuk ke $temp dalam lingkup lokalnya memperbaiki masalah

```
<?php
    $temp = "The date is ";
    echo $temp . longdate(time());
    function longdate($timestamp)
    {
        return date("l F jS Y", $timestamp);
    }
?>
```

Contoh solusi alternatif: melewatkan $temp sebagai argumen

```
<?php
    $temp = "The date is ";
    echo longdate($temp, time());
    function longdate($text, $timestamp)
    {
        return $text . date("l F jS Y", $timestamp);
    }
?>
```

Catatan: Melupakan cakupan variabel adalah kesalahan pemrograman yang umum, jadi mengingat cara kerja cakupan variabel akan membantu kita men-debug beberapa masalah yang cukup kabur.

Variabel global

Ada kasus ketika kita memerlukan variabel untuk memiliki cakupan global, karena kita ingin semua kode kita dapat mengaksesnya. Selain itu, beberapa data mungkin berukuran besar dan kompleks, dan kita tidak ingin terus meneruskannya sebagai argumen ke fungsi. Untuk mendeklarasikan variabel memiliki cakupan global, gunakan kata kunci global. Mari kita asumsikan bahwa kita memiliki cara untuk memasukkan pengguna kita ke situs web kita dan ingin semua kode kita tahu apakah itu berinteraksi dengan pengguna yang masuk atau tamu. Salah satu cara untuk melakukannya adalah dengan membuat variabel global seperti $is_logged_in:

```
global $is_logged_in;
```

Sekarang fungsi login kita hanya perlu menyetel variabel itu ke 1 saat upaya login berhasil, atau 0 saat gagal. Karena cakupan variabel bersifat global, setiap baris kode dalam program kita dapat mengaksesnya. Kita harus menggunakan variabel global dengan hati-hati. Saya sarankan kita membuatnya hanya ketika kita benar-benar tidak dapat menemukan cara lain untuk mencapai hasil yang kita inginkan. Secara umum, program yang dipecah menjadi bagian-bagian kecil dan data terpisah tidak terlalu bermasalah dan lebih mudah dirawat. Jika kita memiliki program seribu baris (dan suatu hari kita akan melakukannya) di mana kita menemukan bahwa variabel global memiliki nilai yang salah di beberapa titik, berapa lama waktu yang kita perlukan untuk menemukan kode yang mengaturnya secara salah? Juga, jika kita memiliki terlalu banyak variabel global, kita berisiko menggunakan salah satu dari nama itu lagi secara lokal, atau setidaknya berpikir kita telah menggunakannya secara lokal, padahal

sebenarnya sudah dinyatakan sebagai global. Segala macam bug aneh dapat muncul dari situasi seperti itu.

Catatan: Terkadang saya mengadopsi konvensi untuk membuat semua nama variabel global menjadi huruf besar (seperti yang direkomendasikan bahwa konstanta harus huruf besar) sehingga saya dapat melihat secara sekilas ruang lingkup suatu variabel.

Variabel statis

Di bagian Variabel lokal, saya menyebutkan bahwa nilai variabel akan dihapus ketika fungsi berakhir. Jika suatu fungsi berjalan berkali-kali, itu dimulai dengan salinan baru dari variabel dan pengaturan sebelumnya tidak berpengaruh. Inilah kasus yang menarik. Bagaimana jika kita memiliki variabel lokal di dalam fungsi yang tidak ingin kita akses ke bagian lain dari kode Anda, tetapi kita juga ingin menyimpan nilainya untuk fungsi berikutnya dipanggil?

Contoh sebuah fungsi menggunakan variabel statis

```
<php
    function test()
    {
        static $count = 0;
        echo $count;
        $count++;
    }
?>
```

Di sini baris pertama pengujian fungsi membuat variabel statis yang disebut $count dan menginisialisasinya ke nilai 0. Baris berikutnya menampilkan nilai variabel; yang terakhir menambahnya. Lain kali fungsi dipanggil, karena $count telah dideklarasikan, baris pertama dari fungsi akan dilewati. Kemudian nilai $count yang bertambah sebelumnya ditampilkan sebelum variabel bertambah lagi. Jika kita berencana untuk menggunakan variabel statis, kita harus mencatat bahwa kita tidak dapat menetapkan hasil ekspresi dalam definisinya. Mereka dapat diinisialisasi hanya dengan nilai-nilai yang telah ditentukan sebelumnya.

Contoh deklarasi variabel statis yang diizinkan dan tidak diizinkan

```
<?php
    static $int = 0; // Allowed
    static $int = 1+2; // Disallowed (will produce a Parse error)
    static $int = sqrt(144); // Disallowed
?>
```

Variabel superglobal

Dimulai dengan PHP 4.1.0, tersedia beberapa variabel standar. Ini dikenal sebagai variabel superglobal, yang berarti bahwa mereka disediakan oleh lingkungan PHP tetapi bersifat global di dalam program, dapat diakses secara mutlak di mana saja.

Tabel 1.9 Variabel superglobal PHP

Nama Superglobal	Konten
$GLOBALS	Semua variabel yang saat ini didefinisikan dalam lingkup global skrip. Nama variabel adalah kunci dari array

$_SERVER	Informasi seperti header, jalur, dan lokasi skrip. Entri dalam larik ini dibuat oleh server web, dan tidak ada jaminan bahwa setiap server web akan menyediakan salah satu atau semua ini.
$_GET	Variabel yang diteruskan ke skrip saat ini melalui metode HTTP GET.
$_POST	Variabel yang diteruskan ke skrip saat ini melalui metode HTTP POST.
$_FILES	Item yang diunggah ke skrip saat ini melalui metode HTTP POST.
$_COOKIE	Variabel diteruskan ke skrip saat ini melalui cookie HTTP.
$_SESSION	Variabel sesi tersedia untuk skrip saat ini.
$_REQUEST	Isi informasi yang dikirimkan dari browser; secara default, $_GET, $_POST, dan $_COOKIE.
$_ENV	Variabel yang diteruskan ke skrip saat ini melalui metode lingkungan

Semua superglobal (kecuali $GLOBALS) diberi nama dengan satu garis bawah awal dan hanya huruf kapital; oleh karena itu, kita harus menghindari penamaan variabel kita sendiri dengan cara ini untuk menghindari kemungkinan kebingungan. Untuk mengilustrasikan bagaimana kita menggunakannya, mari kita lihat sedikit informasi yang digunakan banyak situs. Di antara banyak nugget informasi yang diberikan oleh variabel superglobal adalah URL halaman yang merujuk pengguna ke halaman web saat ini. Informasi halaman rujukan ini dapat diakses seperti ini:

```
$came_from = $_SERVER['HTTP_REFERER'];
```

Jika pengguna datang langsung ke halaman web Anda, seperti dengan mengetikkan URL-nya langsung ke browser, $came_from akan disetel ke string kosong.

Superglobal dan keamanan

Peringatan sebelum kita mulai menggunakan variabel superglobal, karena variabel tersebut sering digunakan oleh peretas yang mencoba menemukan eksploitasi untuk membobol situs web Anda. Apa yang mereka lakukan adalah memuat $_POST, $_GET, atau superglobal lainnya dengan kode berbahaya, seperti perintah Unix atau MySQL yang dapat merusak atau menampilkan data sensitif jika kita mengaksesnya secara naif.

Oleh karena itu, kita harus selalu membersihkan superglobal sebelum menggunakannya. Salah satu cara untuk melakukannya adalah melalui fungsi PHP htmlentities. Ini mengubah semua karakter menjadi entitas HTML. Misalnya, karakter kurang dari dan lebih besar dari (< dan >) diubah menjadi string < dan > sehingga mereka dianggap tidak berbahaya, seperti semua kutipan dan garis miring terbalik, dan seterusnya. Oleh karena itu, cara yang jauh lebih baik untuk mengakses $_SERVER (dan superglobal lainnya) adalah:

```
$came_from = htmlentities($_SERVER['HTTP_REFERER']);
```

Catatan: Menggunakan fungsi htmlentities untuk sanitasi adalah praktik penting dalam keadaan apapun di mana pengguna atau data pihak ketiga lainnya sedang diproses untuk output, tidak hanya dengan superglobal.

1.14 MEMAHAMI JENIS DATA

Nilai yang disimpan dalam variabel atau konstanta disimpan sebagai tipe data tertentu. PHP menyediakan delapan tipe data ini:

- Integer: Bilangan bulat
- Floating-point number (float): Nilai numerik dengan angka desimal

- String: Serangkaian karakter
- Boolean: Nilai yang bisa benar atau salah
- NULL: Nilai yang tidak mewakili nilai
- Array: Sekelompok nilai dalam satu variabel
- Object: Struktur yang dibuat dengan kelas
- Resource: Referensi yang mengidentifikasi koneksi

Berikut adalah beberapa hal yang perlu kita ketahui tentang bekerja dengan tipe data:

- PHP menentukan tipe data secara otomatis. Saat menulis skrip PHP, kita tidak perlu menentukan tipe data yang kita simpan. Dua pernyataan berikut menyimpan tipe data yang berbeda:

```
$var1 = 123;
$var2 = “123”;
```

- Nilai untuk $var1 disimpan sebagai bilangan bulat. Nilai untuk $var2 disimpan sebagai string karena diapit tanda kutip.
- PHP mengonversi tipe data secara otomatis saat dibutuhkan. Misalnya, jika kita menambahkan dua variabel, satu berisi integer dan satu berisi float, PHP mengubah integer menjadi float sehingga keduanya dapat ditambahkan.
- Kita dapat menentukan tipe data. Terkadang, kita mungkin ingin menyimpan nilai sebagai tipe data yang berbeda dari tipe data yang disimpan secara otomatis oleh PHP. Kita dapat mengatur tipe data untuk variabel dengan pemeran, sebagai berikut:

```
$var3 = “222”;
$var4 = (int) $var3;
```

- Pernyataan ini menetapkan $var4 sama dengan nilai dalam $var3, mengubah nilai dari string menjadi integer. Kita juga dapat melakukan cast menggunakan (float) atau (string).
- Kita dapat menanyakan tipe data. Kita dapat mengetahui tipe data mana yang disimpan dalam variabel dengan var_dump(). Misalnya, kita dapat menampilkan variabel sebagai berikut:

```
var_dump($var4);
```

- Output dari pernyataan ini adalah sebagai berikut:

```
int(222)
```

BAB 2
EKSPRESI DAN KONTROL PHP

2.1 EKSPRESI

Mari kita mulai dengan bagian paling mendasar dari bahasa pemrograman apa pun: *ekspresi*. Ekspresi adalah kombinasi nilai, variabel, operator, dan fungsi yang menghasilkan nilai. Ini akrab bagi siapa saja yang telah mengambil aljabar sekolah menengah:

```
y = 3(abs(2x) + 4)
```

yang dalam PHP akan menjadi:

```
$y = 3 (abs(2 $x) + 4);
```

Nilai yang dikembalikan (y, atau $y dalam kasus ini) dapat berupa angka, string, atau nilai Boolean (dinamai setelah George Boole, ahli matematika dan filsuf Inggris abad kesembilan belas). Sekarang, kita seharusnya sudah familiar dengan dua tipe nilai pertama, tapi saya akan menjelaskan yang ketiga.

2.2 TRUE ATAU FALSE?

Nilai Boolean dasar dapat berupa TRUE atau FALSE. Misalnya, ekspresi “20 > 9” (20 lebih besar dari 9) adalah TRUE, dan ekspresi “5 == 6” (5 sama dengan 6) adalah FALSE. Catatan: Perhatikan bahwa saya menggunakan huruf besar untuk nama TRUE dan FALSE. Ini karena mereka adalah konstanta yang telah ditentukan sebelumnya dalam PHP. Kita juga dapat menggunakan versi huruf kecil, jika kita mau, karena mereka juga sudah ditentukan sebelumnya. Faktanya, versi huruf kecil lebih stabil, karena PHP tidak mengizinkan kita untuk mendefinisikan ulang; yang huruf besar mungkin didefinisikan ulang—sesuatu yang harus kita ingat jika kita mengimpor kode pihak ketiga.

Contoh di bawah ini menunjukkan beberapa ekspresi sederhana: dua yang baru saja saya sebutkan, ditambah beberapa lagi. Untuk setiap baris, itu mencetak huruf antara a dan d, diikuti oleh titik dua dan hasil dari ekspresi. Tag
 ada untuk membuat jeda baris dan dengan demikian memisahkan output menjadi empat baris dalam HTML.

Contoh Empat ekspresi Boolean sederhana

```
<?php
    echo "a: [" . (20 > 9) . "] ";
    echo "b: [" . (5 == 6) . "] ";
    echo "c: [" . (1 == 0) . "] ";
    echo "d: [" . (1 == 1) . "] ";
?>
```

Output dari kode ini adalah sebagai berikut:

```
a: [1]
b: []
c: []
d: [1]
```

Perhatikan bahwa kedua ekspresi a: dan d: mengevaluasi ke TRUE, yang memiliki nilai 1. Tetapi b: dan c:, yang mengevaluasi ke FALSE, tidak menunjukkan nilai apapun, karena dalam PHP konstanta FALSE didefinisikan sebagai NULL, atau tidak ada. Untuk memverifikasi ini sendiri, kita dapat memasukkan kode seperti contoh di bawah ini.

Contoh Mengeluarkan nilai TRUE dan FALSE

```
<?php // test2.php
    echo "a: [" . TRUE . "] ";
    echo "b: [" . FALSE . "] ";
?>
```

yang akan memberikan output sebagai berikut:

```
a: [1]
b: []
```

Dalam beberapa bahasa FALSE dapat didefinisikan sebagai 0 atau bahkan 1, jadi ada baiknya memeriksa definisinya di setiap bahasa.

Catatan: Sekarang kita sepenuhnya memasuki era HTML5, dan XHTML tidak lagi direncanakan untuk menggantikan HTML, kita tidak perlu menggunakan bentuk
 yang menutup sendiri dari tag
, atau elemen kosong apa pun (yang tanpa tag penutup), karena / sekarang opsional. Oleh karena itu, saya telah memilih untuk menggunakan gaya yang lebih sederhana dalam buku ini. Jika kita pernah membuat tag HTML non-void menutup sendiri (seperti), tag tersebut tidak akan berfungsi di HTML5 karena / akan diabaikan, dan kita harus menggantinya dengan, misalnya, ... Namun, kita tetap harus menggunakan bentuk sintaks HTML
 saat menggunakan XHTML.

2.3 LITERAL DAN VARIABEL

Bentuk ekspresi yang paling sederhana adalah literal, yang berarti sesuatu yang mengevaluasi dirinya sendiri, seperti angka 73 atau string "Halo". Ekspresi juga bisa berupa variabel, yang mengevaluasi nilai yang telah ditetapkan padanya. Keduanya adalah jenis ekspresi, karena mereka mengembalikan nilai.

Contoh Literal dan variabel

```
<?php
    $myname = "Agus";
    $myage = 37;
    echo "a: " . 73 . ""; // Numeric literal
    echo "b: " . "Hello" . " "; // String literal
    echo "c: " . FALSE . ""; // Constant literal
    echo "d: " . $myname . ""; // String variable
    echo "e: " . $myage . " "; // Numeric variable
?>
```

Dan, seperti yang kita harapkan, kita melihat nilai pengembalian dari semua ini dengan pengecualian c:, yang mengevaluasi ke FALSE, tidak mengembalikan apa pun pada output berikut:

```
a: 73
```

b: Hello
c:
d: Agus
e: 37

Dalam hubungannya dengan operator, dimungkinkan untuk membuat ekspresi yang lebih kompleks yang mengevaluasi hasil yang bermanfaat.

Saat kita menggabungkan konstruksi tugas atau aliran kontrol dengan ekspresi, hasilnya adalah pernyataan. Contoh dibawah menunjukkan satu dari masing-masing. Yang pertama memberikan hasil dari ekspresi 366 - $day_number ke variabel $days_to_new_year, dan yang kedua mengeluarkan pesan ramah hanya jika ekspresi $days_to_new_year < 30 bernilai TRUE.

Contoh sebuah ekspresi dan pernyataan

```
<?php
    $days_to_new_year = 366 - $day_number; // Expression
    if ($days_to_new_year < 30)
    {
        echo "Not long now till new year"; // Statement
    }
?>
```

Operator

PHP menawarkan banyak operator hebat mulai dari aritmatika, string, dan operator logika hingga penugasan, perbandingan, dan banyak lagi.

Tabel 2.1 Jenis operator PHP

Operator	Deskripsi	Contoh
Aritmetic	Matematika dasar	$a + $b
Array	Kesatuan array	$a + $b
Assignment	Penetapan nilai	$a = $b +23
Bitwise	Memanipulasi bit dalam byte	12 ^ 9
Comparison	Bandingkan dua nilai	$a < $b
Eksekusi	Mengeksekusi konten centang kembali	‘ls -al’
Kenaikan/penurunan	Tambah atau kurangi 1	$a++
Logika	Boolean	$a dan $b
String	Rangkaian	$a . $b

Setiap operator mengambil jumlah operan yang berbeda:

- Operator unary, seperti incrementing ($a++) atau negation (-$a), yang mengambil satu operan.
- Operator biner, yang mewakili sebagian besar operator PHP, termasuk penambahan, pengurangan, perkalian, dan pembagian.
- Satu operator ternary, yang berbentuk ? x : y. Ini adalah pernyataan if satu baris singkat yang memilih antara dua ekspresi, tergantung pada hasil ekspresi ketiga.

Prioritas Operator

Jika semua operator memiliki prioritas yang sama, mereka akan diproses sesuai urutan kemunculannya. Padahal, banyak operator memang memiliki prioritas yang sama.

Contoh tiga ekspresi yang setara

1 + 2 + 3 − 4 + 5
2 − 4 + 5 + 3 + 1
5 + 2 − 4 + 1 + 3

Di sini kita akan melihat bahwa meskipun angka (dan operator sebelumnya) telah dipindahkan, hasil dari setiap ekspresi adalah nilai 7, karena operator plus dan minus memiliki prioritas yang sama. Kita dapat mencoba hal yang sama dengan perkalian dan pembagian

Contoh tiga ekspresi yang juga setara

1 2 3 / 4 5
2 / 4 5 3 1
5 2 / 4 1 * 3

Contoh tiga ekspresi menggunakan operator dengan prioritas campuran

1 + 2 * 3 − 4 5
2 − 4 5 3 + 1
5 + 2 − 4 + 1 3

Jika tidak ada operator yang didahulukan, ketiga ekspresi ini akan dievaluasi menjadi 25, 29, dan 12, masing-masing. Tetapi karena perkalian dan pembagian lebih diutamakan daripada penambahan dan pengurangan, ada tanda kurung tersirat di sekitar bagian ekspresi ini, yang akan terlihat seperti contoh dibawah ini.

Contoh tiga ekspresi yang menunjukkan tanda kurung tersirat

1 + (2 3) − (4 5)
2 − (4 5 3) + 1
5 + 2 − 4 + (1 * 3)

PHP harus mengevaluasi subekspresi dalam tanda kurung terlebih dahulu untuk mendapatkan ekspresi setengah jadi, lihat contoh dibawah ini.

Contoh setelah mengevaluasi subekspresi dalam tanda kurung

1 + (6) − (20)
2 − (60) + 1
5 + 2 − 4 + (3)

Hasil akhir dari ekspresi ini masing-masing adalah 13, 57, dan 6 (sangat berbeda dari hasil 25, 29, dan 12 yang akan kita lihat seandainya tidak ada operator yang didahulukan). Tentu saja, kita dapat mengganti prioritas operator default dengan memasukkan tanda kurung kita sendiri dan memaksakan hasil asli yang akan kita lihat seandainya tidak ada operator yang diutamakan.

Contoh memaksa evaluasi kiri-ke-kanan

((1 + 2) * 3 − 4) * 5
(2 − 4) 5 3 + 1
(5 + 2 − 4 + 1) * 3

Dengan memasukkan tanda kurung dengan benar, sekarang kita melihat nilai masing-masing 25, 29, dan 12.

Tabel 2.2 prioritas operator PHP (tinggi ke rendah)

Operator	**Jenis**
()	Tanda kurung

++ --	Kenaikan/penurunan
!	Logika
*** / %**	Aritmatika
+ - .	Aritmatika dan String
<< >>	Sedikit demi sedikit (Bitwise)
< <= > >= <>	Perbandingan
== != === !==	Perbandingan
&	Bitwise (dan referensi)
^	Bitwise
|	Bitwise
&&	Logika
| |	Logika
? :	Ternary
= += –= *= /= .= %= &= != ^= <<= >>=	Penentapan
and	Logika
xor	Logika
or	Logika

2.4 KETERKAITAN (*ASSOCIATIVITY*)

Kita telah melihat pemrosesan ekspresi dari kiri ke kanan, kecuali di mana prioritas operator berlaku. Tetapi beberapa operator memerlukan pemrosesan dari kanan ke kiri, dan arah pemrosesan ini disebut asosiatifitas operator. Untuk beberapa operator tidak ada asosiatif. Asosiatif menjadi penting dalam kasus di mana kita tidak secara eksplisit memaksakan prioritas, jadi kita perlu menyadari tindakan default operator.

Tabel 2.3 Asosiati operator

Operator	Deskripsi	Associativity
CLONE NEW	Membuat objek baru	Tidak ada
< <= >= == != === !== <>	Perbandingan	Tidak ada
!	Logika NOT	Kanan
~	Bitwise NOT	Kanan
++ --	Kenaikan dan penurunan	Kanan
(int)	Cast ke bilangan bulat	Kanan
(double) (float) (real)	Cast ke angka floating-point	Kanan
(string)	Cast ke string	Kanan
(array)	Cast ke array	Kanan
(object)	Cast ke object	Kanan
@	Menghambat pelaporan kesalahan	Kanan
= += –= *= /=	Penugasan	Kanan
.= %= &= |= ^= <<= >>=	Penugasan	Kanan
+	Penambahan dan unary plus	Kiri
-	Pengurangan dan negasi	Kiri
*****	Pengurangan dan negasi	Kiri
/	Divisi	Kiri
%	Modulus	Kiri
.	Penggabungan string	Kiri
<< >> & ^ |	Sedikit demi sedikit	Kiri

?:	Terner	Kiri
|| && dan or xor	Logika	Kiri
,	Pemisah	Kiri

Contoh pernyataan tugas ganda

```
<?php
   $level = $score = $time = 0;
?>
```

Penetapan ganda ini hanya mungkin jika bagian paling kanan dari ekspresi dievaluasi terlebih dahulu dan kemudian pemrosesan berlanjut ke arah kanan-ke-kiri. Catatan: Sebagai pemula PHP, kita harus menghindari potensi jebakan asosiasi operator dengan selalu menyarangkan subekspresi kita di dalam tanda kurung untuk memaksakan urutan evaluasi. Ini juga akan membantu pemrogram lain yang mungkin harus memelihara kode kita untuk memahami apa yang terjadi.

Bekerja dengan bilangan bulat dan bilangan floating-point

Bilangan bulat adalah bilangan 1, 10, dan 333. Bilangan titik-mengambang, juga disebut bilangan real, adalah bilangan yang berisi nilai desimal, seperti 3,1 atau 0,667. PHP menyimpan nilai sebagai integer atau float secara otomatis.

Melakukan operasi aritmatika pada tipe data numerik

PHP memungkinkan kita untuk melakukan operasi aritmatika pada angka. Kita menunjukkan operasi aritmatika dengan dua angka dan operator aritmatika. Misalnya, satu operator adalah tanda plus (+), sehingga kita dapat menunjukkan operasi aritmatika seperti ini:

```
1 + 2
```

Kita juga dapat melakukan operasi aritmatika dengan variabel yang berisi angka, sebagai berikut:

```
$n1 = 1;
$n2 = 2;
$sum = $n1 + $n2;
```

Kita dapat menambahkan nomor yang bukan tipe data yang sama, sebagai berikut:

```
$n1 = 1,5;
$n2 = 2;
$sum = $n1 + $n2;
```

PHP mengonversi $n2 menjadi float (2.0) dan menambahkan dua nilai. $sum kemudian menjadi pelampung.

2.5 MENGGUNAKAN OPERATOR ARITMATIKA

PHP menyediakan lima operator aritmatika. Kita dapat melakukan beberapa operasi aritmatika sekaligus. Misalnya, pernyataan berikut melakukan tiga operasi:

```
$result = 1 + 2 * 4 + 1;
```

Urutan di manakah aritmatika dilakukan adalah hal yang penting. Kita bisa mendapatkan hasil yang berbeda tergantung pada operasi mana yang dilakukan terlebih dahulu. PHP melakukan perkalian dan pembagian terlebih dahulu, diikuti dengan penambahan dan pengurangan. Jika pertimbangan lain sama, PHP bergerak dari kiri ke kanan. Akibatnya, pernyataan sebelumnya menetapkan $result ke 10, dalam urutan berikut:

$result = 1 + 2 * 4 + 1 (Pertama melakukan perkalian.)
$result = 1 + 8 + 1 (Selanjutnya melakukan penambahan paling kiri.)
$result = 9 + 1 (Selanjutnya melakukan penambahan yang tersisa.)
$result = 10

Kita dapat mengubah urutan pelaksanaan aritmatika dengan menggunakan tanda kurung. Aritmatika di dalam tanda kurung dilakukan terlebih dahulu. Misalnya, kita dapat menulis pernyataan sebelumnya dengan tanda kurung seperti ini:

$ result = (1 + 2) * 4 + 1;

Pernyataan ini menetapkan $result ke 13, dengan urutan sebagai berikut:

$result = (1 + 2) * 4 + 1 (Pertama menghitung dalam tanda kurung.)
$result = 3 * 4 + 1 (Selanjutnya melakukan perkalian.)
$result = 12 + 1 (Selanjutnya melakukan penambahan.)
$ result = 13

Pada prinsip lebih baik-aman-dari-maaf, yang terbaik adalah menggunakan tanda kurung kapan pun lebih dari satu jawaban dimungkinkan

Memformat angka sebagai jumlah matauang

Seringkali, angka yang kita gunakan adalah jumlah matauang, seperti harga produk. Kita ingin pelanggan kita melihat harga dalam format yang tepat di halaman web. Dengan kata lain, jumlah dolar harus selalu memiliki dua tempat desimal. Namun, PHP menyimpan dan menampilkan angka dalam format yang paling efisien. Jika angkanya adalah 10.000, maka akan ditampilkan sebagai 10. Untuk memasukkan angka ke dalam format rupiah yang benar, kita dapat menggunakan sprintf. Pernyataan berikut memformat angka ke dalam format dolar:

$*newvariablename* = sprintf(“%01.2f”, $oldvariablename);

Pernyataan ini memformat ulang angka dalam *$oldvariablename* dan menyimpannya dalam format baru di *$newvariablename*, yang merupakan tipe data string. Misalnya, pernyataan berikut menampilkan uang dalam format yang benar:

$price = 25;
$f_price = sprintf(“%01.2f”,$price);
echo “$f_price”;

Kita melihat yang berikut di halaman web:

25.000

Jika kita menampilkan variabel dengan var_dump($f_price), hasilnya adalah:

```
string(5) “25.000”
```

Jika kita ingin koma untuk memisahkan ribuan nomor Anda, kita dapat menggunakan number_format. Pernyataan berikut membuat format dolar dengan koma:

```
$price = 25.000;
$f_price = angka_format($price,2);
echo “$f_price”;
```

Kita melihat yang berikut di halaman web:

```
25.000.00
```

2 dalam pernyataan number_format menyetel format ke dua tempat desimal. Kita dapat menggunakan angka apa pun untuk mendapatkan sejumlah tempat desimal. Juga, kita dapat menambahkan $ di depan jumlah dolar dalam output seperti ini:

```
echo “$”. $f_price;
```

Operator Relasional

Operator relasional menguji dua operan dan mengembalikan hasil Boolean dari TRUE atau FALSE. Ada tiga jenis operator relasional: persamaan, perbandingan, dan logika.

Persamaan

Seperti yang telah kita temui beberapa kali dalam bab ini, operator persamaan adalah == (dua tanda sama dengan). Penting untuk tidak membingungkannya dengan operator penugasan = (satu tanda sama dengan).

Contoh menetapkan nilai dan menguji kesetaraan

```
<?php
    $month = "March";
    if ($month == "March") echo "It's springtime";
?>
```

Seperti yang kita lihat, dengan mengembalikan TRUE atau FALSE, operator kesetaraan memungkinkan kita menguji kondisi menggunakan, misalnya, pernyataan if. Tapi itu bukan keseluruhan cerita, karena PHP adalah bahasa yang diketik secara longgar. Jika dua operan dari ekspresi kesetaraan memiliki tipe yang berbeda, PHP akan mengonversinya ke tipe apa pun yang paling masuk akal.

Misalnya, string apa pun yang seluruhnya terdiri dari angka akan dikonversi menjadi angka setiap kali dibandingkan dengan angka. Dalam contoh dibawah ini, $a dan $b adalah dua string yang berbeda, dan oleh karena itu kita tidak mengharapkan pernyataan if untuk menghasilkan hasil.

Contoh operator kesetaraan dan identitas

```
<?php
    $a = "1000";
    $b = "+1000";
```

```
    if ($a == $b) echo "1";
    if ($a === $b) echo "2";
?>
```

Namun, jika kita menjalankan contoh, kita akan melihat bahwa itu menghasilkan angka 1, yang berarti bahwa pernyataan if pertama dievaluasi menjadi TRUE. Ini karena kedua string pertama kali dikonversi ke angka, dan 1000 adalah nilai numerik yang sama dengan +1000. Sebaliknya, pernyataan if kedua menggunakan operator identitas—tiga tanda sama dengan berturut-turut—yang mencegah PHP mengonversi tipe secara otomatis. $a dan $b karena itu dibandingkan sebagai string dan sekarang ditemukan berbeda, jadi tidak ada output.

Seperti halnya memaksa operator didahulukan, setiap kali kita ragu tentang bagaimana PHP akan mengonversi tipe operan, kita dapat menggunakan operator identitas untuk menonaktifkan perilaku ini. Dengan cara yang sama kita dapat menggunakan operator kesetaraan untuk menguji operan yang sama, kita dapat mengujinya tidak sama dengan menggunakan !=, operator ketidaksetaraan. Perhatikan contoh dibawah, yang merupakan penulisan ulang Contoh dibawah di mana operator persamaan dan identitas telah diganti dengan inversnya.

Contoh pertidaksamaan dan bukan operator identik

```
<?php
    $a = "1000";
    $b = "+1000";

    if ($a != $b) echo "1";
    if ($a !== $b) echo "2";
?>
```

Pernyataan if pertama tidak menampilkan angka 1, karena kode tersebut menanyakan apakah $a dan $b tidak sama satu sama lain secara numerik. Sebagai gantinya, ia mengeluarkan angka 2, karena pernyataan if kedua menanyakan apakah $a dan $b tidak identik satu sama lain dalam jenis operan mereka saat ini, dan jawabannya adalah TRUE; mereka tidak sama.

Operator perbandingan

Dengan menggunakan operator perbandingan, kita dapat menguji lebih dari sekadar kesetaraan dan ketidaksetaraan. PHP juga memberi kita > (lebih besar dari), < (lebih kecil dari), >= (lebih besar dari atau sama dengan), dan <= (kurang dari atau sama dengan) untuk dimainkan. Contoh dibawah ini menunjukkan operator ini digunakan.

Contoh empat operator pembanding

```
<?php
    $a = 2; $b = 3; if ($a > $b) echo "$a lebih besar dari $b";
    if ($a < $b) echo "$a lebih sedikit dari $b";
    if ($a >= $b) echo "$a lebih besar atau sama dengan $b";
    if ($a <= $b) echo "$a lebih seidikit atau sama dengan $b";
>?
```

Dalam contoh ini, di mana $a adalah 2 dan $b adalah 3, berikut adalah outputnya:

2 lebih sedikit dari 3
2 lebih sedikit atau sama dengan 3

Coba contoh ini sendiri, ubah nilai $a dan $b, untuk melihat hasilnya. Coba atur ke nilai yang sama dan lihat apa yang akan terjadi.

Operator "?"

Salah satu cara untuk menghindari verbositas pernyataan if dan else adalah dengan menggunakan operator ternary yang lebih ringkas, ?, yang tidak biasa karena membutuhkan tiga operan daripada dua tipikal. Operator ? melewati ekspresi yang harus dievaluasi, bersama dengan dua pernyataan untuk dieksekusi: satu untuk saat ekspresi dievaluasi ke TRUE, yang lain untuk saat FALSE. Contoh dibawah menunjukkan kode yang mungkin kita gunakan untuk menulis peringatan tentang tingkat bahan bakar mobil ke dasbor digitalnya.

Contoh Menggunakan operator?

```
<?php
    echo $fuel <= 1 ? "isi tangki sekarang" : "Ada cukup bahan bakar";
?>
```

Dalam pernyataan ini, jika ada satu galon fuel (bahan bakar) atau kurang dari itu (yaitu, jika $bahanbakar diatur ke 1 atau kurang), string Isi tangki sekarang dikembalikan ke pernyataan echo sebelumnya. Jika tidak, string Ada cukup bahan bakar dikembalikan. Kita juga dapat menetapkan nilai yang dikembalikan dalam pernyataan? ke variabel

Contoh menetapkan hasil bersyarat ? ke variabel

```
<?php
    $enough = $fuel <= 1 ? FALSE : TRUE;
?>
```

Di sini $enough akan diberi nilai TRUE hanya jika ada lebih dari satu galon bahan bakar; jika tidak, akan diberi nilai FALSE. Jika kita menemukan kode ? membingungkan operator, kita bebas untuk mengikuti pernyataan if, tetapi kita harus terbiasa dengannya, karena kita akan melihatnya di kode orang lain. Ini bisa sulit dibaca, karena sering kali menggabungkan beberapa kemunculan dari variabel yang sama. Misalnya, kode seperti berikut ini cukup populer:

```
$saved = $saved >= $new ? $saved : $new;
```

Jika kita membongkarnya dengan hati-hati, kita dapat mengetahui apa yang dilakukan kode ini:

```
$saved =          // Set the value of $saved to...
    $saved >= $new // Check $saved against $new
?                 // Yes, comparison is true ...
    $saved        // ... so assign the current value of $saved
:                 // No, comparison is false ...
    $new;         // ... so assign the value of $new
```

Ini adalah cara ringkas untuk melacak nilai terbesar yang kita lihat saat program berjalan. Kita menyimpan nilai terbesar di $saved dan membandingkannya dengan $new setiap kali kita mendapatkan nilai baru. Programmer akrab dengan operator ? merasa lebih nyaman daripada jika pernyataan untuk perbandingan singkat tersebut. Saat tidak digunakan untuk menulis

kode ringkas, biasanya digunakan untuk membuat beberapa keputusan sebaris, seperti saat kita menguji apakah suatu variabel disetel sebelum meneruskannya ke suatu fungsi.

2.6 MENGGUNAKAN STRING KARAKTER

String karakter adalah serangkaian karakter. Karakter adalah huruf, angka, dan tanda baca. Bila angka digunakan sebagai karakter maka angka tersebut bersifat sama seperti huruf. angka tidak bisa digunakan dalam aritmatika. Misalnya, nomor telepon disimpan sebagai string karakter karena hanya perlu disimpan — tidak untuk ditambahkan atau dikalikan.

Menetapkan string ke variabel

Saat kita menyimpan string karakter dalam variabel, kita memberi tahu PHP di mana string dimulai dan diakhiri dengan menggunakan tanda kutip ganda atau tanda kutip tunggal. Misalnya, dua pernyataan berikut menghasilkan hasil yang sama:

```
$string = “Hello World!”;
$string = ‘Hello World!’;
```

Namun, misalkan kita ingin menyimpan string sebagai berikut:

```
$string = 'Ini adalah rumah Sally';
echo $string;
```

Pernyataan ini tidak akan berfungsi karena ketika PHP melihat ' (kutipan tunggal) setelah Sally, PHP akan berpikir bahwa ini adalah akhir dari string, menampilkan yang berikut:

```
Ini adalah Sally
```

Kita perlu memberi tahu PHP untuk menginterpretasikan tanda kutip tunggal (') sebagai apostrof, bukan sebagai akhir string. Kita dapat melakukannya dengan menggunakan garis miring terbalik (\) di depan tanda kutip tunggal. Garis miring terbalik memberi tahu PHP bahwa kutipan tunggal tidak memiliki arti khusus; itu hanya sebuah apostrof. Ini disebut melarikan diri dari karakter. Gunakan pernyataan berikut untuk menampilkan seluruh string:

```
$string = 'Ini adalah rumah Sally';
echo $string;
```

Demikian pula, ketika kita menyertakan string dalam tanda kutip ganda, kita juga harus menggunakan garis miring terbalik di depan tanda kutip ganda dalam string.

Menggunakan tanda kutip tunggal dan ganda dengan string

String dengan tanda kutip tunggal dan tanda kutip ganda ditangani secara berbeda, sebagai berikut:

- String dengan tanda kutip tunggal disimpan secara harfiah — dengan pengecualian \’, yang disimpan sebagai tanda kutip.
- Dalam string yang dikutip ganda, variabel dan beberapa karakter khusus dievaluasi sebelum string disimpan.

Berikut adalah perbedaan terpenting dalam penggunaan tanda kutip ganda atau tunggal dalam kode:

- Handlingvariabel: Jika kita menyertakan variabel dalam tanda kutip ganda, PHP menggunakan nilai variabel. Namun, jika kita menyertakan variabel dalam tanda kutip

tunggal, PHP menggunakan nama variabel literal. Misalnya, jika kita menggunakan pernyataan berikut:

```
$month = 12;
    $result1 = “$month”;
    $result2 = '$month';
    echo $result;
    echo “<br />”;
    echo $result2;
```

Outputnya adalah:

```
12
$month
```

- Memulai baris baru: Karakter khusus \n memberitahu PHP untuk memulai baris baru. Saat kita menggunakan tanda kutip ganda, PHP memulai baris baru di \n; dengan tanda kutip tunggal, \n adalah string literal. Misalnya, ketika menggunakan pernyataan berikut:

```
$string1 = “String in \ndouble quotes”;
$string2 = 'String in \nsingle quotes ';
```

Output string1 adalah:

```
String in
tanda kutip ganda
```

dan output string2 adalah:

```
String in \nsingle quotes
```

- Menyisipkan tab: Karakter khusus \t memberi tahu PHP untuk menyisipkan tab. Saat kita menggunakan tanda kutip ganda, PHP menyisipkan tab di \t, tetapi dengan tanda kutip tunggal, \t adalah string literal. Misalnya, ketika menggunakan pernyataan berikut:

```
$string1 = “String in \tdouble quotes”;
$string2 = 'String in \tsingle quotes';
```

Output string1 adalah:

```
String in double quotes
```

Dan output string2 adalah:

```
String in \tsingle quotes
```

Tanda kutip yang menyertakan seluruh string menentukan perlakuan variabel dan karakter khusus, bahkan jika kumpulan tanda kutip lainnya ada di dalam string. Sebagai contoh, perhatikan pernyataan berikut:

```
$number = 10;
$string1 = "Ada '$number' orang dalam antrean.";
$string2 = 'Ada “$number” orang yang menunggu.';
echo $string1,”<br />\n”;
echo $string2;
```

Outputnya adalah sebagai berikut:

```
Ada '10' orang dalam antrean.
Ada “$number” orang yang menunggu.
```

Menggabungkan string

Kita dapat menggabungkan string, sebuah proses yang disebut penggabungan, dengan menggunakan titik (.). Misalnya, kita dapat menggabungkan string dengan pernyataan berikut:

```
$string1 = 'Hello';
$string2 = 'World!';
$stringall = $string1.$string2;
echo $stringall;
```

Output pernyataan echo adalah:

```
HelloWorld!
```

Perhatikan bahwa tidak ada spasi yang muncul antara Hello dan World. Itu karena tidak ada spasi yang disertakan dalam dua string yang digabungkan. Kita dapat menambahkan spasi di antara kata-kata dengan menggunakan pernyataan gabungan berikut daripada pernyataan sebelumnya:

```
$stringall = $string1.” “.$string2;
```

Kita dapat menggunakan .= untuk menambahkan karakter ke string yang ada. Misalnya, kita dapat menggunakan pernyataan berikut sebagai pengganti pernyataan sebelumnya:

```
$stringall = “Hello”;
$stringall .= “World!”;
echo $stringall;
```

Output pernyataan echo adalah ini:

```
Hello World!
```

Kita juga dapat memisahkan string. Kita dapat memisahkannya pada karakter tertentu atau mencari substring dalam string. Kita menggunakan fungsi untuk melakukan ini dan operasi lainnya pada string.

Menyimpan string yang sangat panjang

PHP menyediakan fitur yang disebut heredoc yang berguna untuk menetapkan nilai yang terdiri dari string yang sangat panjang yang mencakup beberapa baris. Heredoc memungkinkan kita memberi tahu PHP di mana harus memulai dan mengakhiri membaca string. Pernyataan heredoc memiliki format berikut:

```
$varname = <<<ENDSTRING
teks
ENDSTRING;
```

ENDSTRING dapat menyertakan string apa pun. Kita melampirkan teks yang ingin disimpan dalam variabel $varname dengan mengetik ENDSTRING di awal dan di akhir. Saat PHP memproses heredoc, ia membaca ENDSTRING pertama dan mulai membaca teks menjadi $varnama. Itu terus membaca teks ke $varname sampai menemukan ENDSTRING yang sama lagi. Pada saat itu, itu mengakhiri string. String dibuat oleh pernyataan heredoc mengevaluasi variabel dan karakter khusus di dengan cara yang sama seperti string yang dikutip ganda. Pernyataan berikut membuat string dengan metode heredoc:

```
$distance = 10;
$herevariable = <<<ENDOFTEXT
Jarak antara
Semarang dan Ungaran
Apakah $jarak Km.
ENDOFTEXT;
echo $herevariable;
```

Output dari pernyataan echo adalah sebagai berikut:

```
Jarak antara Semarang dan Ungaran adalah 10 km.
```

Tetapi berhati-hatilah. PHP pilih-pilih tentang ENDSTRING-nya. Saat pertama kali muncul, ENDSTRING (ENDOFTEXT dalam contoh ini) harus muncul di akhir baris pertama, tanpa diikuti apa pun, bahkan spasi. Dan ENDSTRING pada baris terakhir harus muncul di awal baris, tanpa apa pun sebelumnya, bahkan spasi dan tidak ada yang mengikutinya selain titik koma. Jika aturan ini dilanggar, PHP tidak akan mengenali string akhir dan akan terus mencarinya di seluruh skrip. Ini pada akhirnya akan menampilkan kesalahan parse yang menunjukkan nomor baris yang merupakan baris terakhir dalam skrip.

2.7 MENGGUNAKAN TIPE DATA BOOLEAN

Tipe data Boolean hanya mengambil nilai true atau false. Kita dapat menetapkan nilai Boolean ke variabel sebagai berikut:

```
$var1 = true;
```

PHP menetapkan variabel ke tipe data Boolean. Nilai Boolean digunakan saat membandingkan nilai dan ekspresi untuk pernyataan bersyarat, seperti pernyataan if.
Nilai-nilai berikut dievaluasi sebagai salah oleh PHP:

- Kata *false*
- Bilangan bulat *0*

- Angka floating-point *0,0*
- String kosong
- String dengan nilai *0*
- Array kosong
- Benda kosong
- Nilai *NULL*

Jika suatu variabel berisi nilai yang tidak dievaluasi sebagai salah, maka variabel tersebut diberi nilai benar.

Menggunakan tipe data NULL

Satu-satunya nilai yang merupakan tipe data NULL adalah NULL. Kita dapat menetapkan nilai ke variabel sebagai berikut:

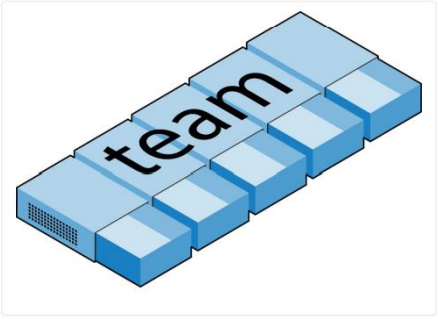
```
$var1 = NULL;
```

Variabel dengan nilai NULL tidak mengandung nilai.

2.8 ARRAY

Anggap saja beberapa kotak korek api yang direkatkan. Katakanlah kita ingin menyimpan nama pemain untuk tim sepak bola lima orang dalam array yang disebut $team. Untuk melakukan ini, kita bisa merekatkan lima kotak korek api secara berdampingan dan menuliskan nama semua pemain di kertas terpisah, menempatkan satu di setiap kotak korek api. Di seluruh bagian atas rakitan kotak korek api, kita akan menulis kata tim. Setara dengan ini di PHP adalah:

```
$team = array('Danial', 'Maria', 'Niko', 'Chris', 'Josh')
```



Gambar 2.1 Array seperti peumpamaan beberapa kotak korek api yang direkatkan

Sintaks ini lebih rumit daripada yang telah saya jelaskan sejauh ini. Kode pembuatan array terdiri dari konstruksi berikut:

```
array();
```

dengan lima senar di dalamnya. Setiap string diapit oleh apostrof. Jika kita ingin tahu siapa pemain 4, kita bisa menggunakan perintah ini:

```
echo $team[3]; // Displays the name Niko
```

Alasan pernyataan sebelumnya memiliki angka 3, bukan 4, adalah karena elemen pertama dari array PHP sebenarnya adalah elemen ke-nol, sehingga angka pemain akan menjadi 0 hingga 4.

Array dua dimensi

Ada lebih banyak lagi yang dapat kita lakukan dengan array. Misalnya, alih-alih menjadi garis kotak korek api satu dimensi, mereka dapat berupa matriks dua dimensi atau bahkan dapat memiliki tiga dimensi atau lebih. Sebagai contoh larik dua dimensi, misalkan kita ingin melacak permainan tic-tac-toe, yang membutuhkan struktur data sembilan sel yang disusun dalam kotak 3x3. Untuk menggambarkannya dengan kotak korek api, bayangkan sembilan di antaranya direkatkan satu sama lain dalam matriks tiga baris dengan tiga kolom.

Gambar 2.2 Array multidimensi yang disimulasikan dengan kotak korek api

Kita sekarang dapat menempatkan selembar kertas dengan "x" atau "o" di kotak korek api yang benar untuk setiap gerakan yang dimainkan. Untuk melakukan ini dalam kode PHP, kita harus menyiapkan larik yang berisi tiga larik lagi.

Contoh Mendefinisikan array dua dimensi

```
<?php
   $oxo      = array(array('x', ' ', ', 'o'),
                     array('o', 'o', 'x'),
                     array('x', 'o', ' '));
?>
```

Ada tiga konstruksi array() yang bersarang di dalam konstruksi array() luar. Untuk kemudian mengembalikan elemen ketiga di baris kedua dari array ini, kita akan menggunakan perintah PHP berikut, yang akan menampilkan x:

```
echo $oxo[1][2]
```

Catatan: Ingatlah bahwa indeks larik (penunjuk pada elemen dalam larik) dimulai dari nol, bukan satu, jadi [1] pada perintah sebelumnya mengacu pada larik kedua dari tiga larik, dan [2] merujuk pada posisi ketiga dalam larik tersebut Himpunan. Ini akan mengembalikan isi kotak korek api tiga bersama dan dua ke bawah.

Seperti yang disebutkan, kita dapat mendukung array dengan lebih banyak dimensi hanya dengan membuat lebih banyak array di dalam array. Namun, kita tidak akan membahas susunan lebih dari dua dimensi dalam buku ini.

Akses Dasar

Kita telah melihat susunan seolah-olah mereka adalah kumpulan kotak korek api yang direkatkan. Cara lain untuk memikirkan array adalah seperti untaian manik-manik, dengan manik-manik mewakili variabel yang dapat berupa numerik, string, atau bahkan array lainnya. Mereka seperti string manik-manik, karena setiap elemen memiliki lokasinya sendiri dan (dengan pengecualian yang pertama dan terakhir) masing-masing memiliki elemen lain di kedua sisi. Beberapa array direferensikan oleh indeks numerik; yang lain mengizinkan

pengidentifikasi alfanumerik. Fungsi bawaan memungkinkan kita mengurutkannya, menambah atau menghapus bagian, dan menelusurinya untuk menangani setiap item melalui jenis loop khusus. Dan dengan menempatkan satu atau lebih array di dalam array lain, kita dapat membuat array dua, tiga, atau sejumlah dimensi.

Array Terindeks Numerik

Mari kita asumsikan bahwa kita telah ditugaskan untuk membuat situs web sederhana untuk sebuah perusahaan peralatan kantor lokal dan saat ini kita sedang mengerjakan bagian yang dikhususkan untuk kertas. Salah satu cara untuk mengelola berbagai item stok dalam kategori ini adalah dengan menempatkannya dalam array numerik. Kita dapat melihat cara paling sederhana untuk melakukannya dalam contoh dibawah ini.

Contoh Menambahkan item ke array

```
<?php
    $paper[] = "Copier";
    $paper[] = "Inkjet";
    $paper[] = "Laser";
    $paper[] = "Photo";
    print_r($paper);
?>
```

Dalam contoh ini, setiap kali kita menetapkan nilai ke array $paper, lokasi kosong pertama dalam array tersebut digunakan untuk menyimpan nilai, dan pointer internal ke PHP ditambahkan untuk menunjuk ke lokasi bebas berikutnya, siap untuk penyisipan di masa mendatang. Fungsi print_r yang sudah dikenal (yang mencetak konten variabel, larik, atau objek) digunakan untuk memverifikasi bahwa larik telah diisi dengan benar. Ini mencetak yang berikut:

```
Array
(
    [0] => Copier
    [1] => Inkjet
    [2] => Laser
    [3] => Photo
)
```

Kode sebelumnya juga bisa ditulis seperti yang ditunjukkan pada contoh setelah paragraf ini, di mana lokasi yang tepat dari setiap item dalam array ditentukan. Namun, seperti yang kita lihat, pendekatan itu memerlukan pengetikan ekstra dan membuat kode kita lebih sulit untuk dipelihara jika kita ingin menyisipkan atau menghapus persediaan dari larik kecuali jika kita ingin menentukan urutan yang berbeda, biasanya lebih baik membiarkan PHP menangani nomor lokasi yang sebenarnya.

Contoh Menambahkan item ke array menggunakan lokasi eksplisit

```
<?php
    $paper[0] = "Copier";
    $paper[1] = "Inkjet";
    $paper[2] = "Laser";
    $paper[3] = "Photo";
    print_r($paper);
```

```
?>
```

Output dari contoh-contoh ini identik, tetapi kita kemungkinan tidak akan menggunakan print_r di situs web yang dikembangkan, jadi contoh dibawah ini menunjukkan bagaimana kita dapat mencetak berbagai jenis kertas yang ditawarkan situs web menggunakan for loop.

Contoh Menambahkan item ke array dan mengambilnya

```
<?php
    $paper[] = "Copier";
    $paper[] = "Inkjet";
    $paper[] = "Laser";
    $paper[] = "Photo";
    for ($j = 0 ; $j < 4 ; ++$j)
        echo "$j: $paper[$j]<br>";
?>
```

Contoh ini mencetak yang berikut:

```
0: Copier
1: Inkjet
2: Laser
3: Photo
```

Sejauh ini, kita telah melihat beberapa cara di mana kita dapat menambahkan item ke array dan salah satu cara untuk mereferensikannya, tetapi PHP menawarkan lebih banyak lagi—yang akan segera saya bahas. Tapi pertama-tama, kita akan melihat tipe array yang lain.

Array Asosiatif

Melacak elemen array berdasarkan indeks berfungsi dengan baik, tetapi dapat membutuhkan kerja ekstra dalam hal mengingat nomor mana yang merujuk ke produk mana. Itu juga dapat membuat kode sulit untuk diikuti oleh programmer lain. Di sinilah array asosiatif menjadi miliknya. Dengan menggunakannya, kita dapat mereferensikan item dalam array berdasarkan nama, bukan dengan nomor. Contoh dibwah ini adalah contoh untuk memperluas kode sebelumnya dengan memberi setiap elemen dalam larik nama pengenal dan nilai string yang lebih panjang dan lebih jelas.

Contoh Menambahkan item ke array asosiatif dan mengambilnya

```
<?php
    $paper['copier'] = "Copier & Multipurpose";
    $paper['inkjet'] = "Inkjet Printer";
    $paper['laser'] = "Laser Printer";
    $paper['photo'] = "Photographic Paper";
    echo $paper['laser'];
?>
```

Di tempat nomor (yang tidak menyampaikan informasi yang berguna, selain dari posisi item dalam array), setiap item sekarang memiliki nama unik yang dapat kita gunakan untuk referensi di tempat lain, seperti dengan pernyataan echo—yang hanya mencetak Printer Laser. Nama-nama (copier, inject, dll.) disebut indeks atau kunci dan item yang ditetapkan

untuk mereka (seperti "Printer Laser") disebut nilai. Fitur PHP yang sangat kuat ini sering digunakan saat kita mengekstrak informasi dari XML dan HTML. Misalnya, pengurai HTML seperti yang digunakan oleh mesin telusur dapat menempatkan semua elemen laman web ke dalam larik asosiatif yang namanya mencerminkan struktur laman:

```
$html['title'] = "My web page";
$html['body'] = "... body of web page ...";
```

Program juga mungkin akan memecah semua tautan yang ditemukan di dalam sebuah halaman ke dalam larik lain, dan semua judul dan subjudul menjadi yang lain. Saat kita menggunakan array asosiatif daripada numerik, kode untuk merujuk ke semua item ini mudah ditulis dan di-debug.

Menggunakan Kata Kunci Array

Sejauh ini, kita telah melihat cara menetapkan nilai ke array hanya dengan menambahkan item baru satu per satu. Apakah kita menentukan kunci, menentukan pengidentifikasi numerik, atau membiarkan PHP menetapkan pengidentifikasi numerik secara implisit, ini adalah pendekatan yang bertele-tele. Metode penugasan yang lebih ringkas dan lebih cepat menggunakan kata kunci array. Contoh dibawah ini menunjukkan array numerik dan asosiatif yang ditetapkan menggunakan metode ini.

Contoh Menambahkan item ke array menggunakan kata kunci array

```
<?php
    $p1 = array("Copier", "Inkjet", "Laser", "Photo");
    echo "p1 element: " . $p1[2] . "<br>";
    $p2 = array('copier' => "Copier & Multipurpose",
                    'inkjet' => "Inkjet Printer",
                    'laser' => "Laser Printer",
                    'photo' => "Photographic Paper");
    echo "p2 element: " . $p2['inkjet'] . "<br>";
?>
```

Paruh pertama cuplikan ini menetapkan deskripsi produk lama yang dipersingkat ke larik $p1. Ada empat item, sehingga mereka akan menempati slot 0 hingga 3. Oleh karena itu, pernyataan echo mencetak yang berikut:

p1 element: Laser

Babak kedua memberikan pengidentifikasi asosiatif dan deskripsi produk yang lebih panjang yang menyertai ke array $p2 menggunakan format indeks => nilai. Penggunaan => mirip dengan operator penugasan = reguler, kecuali bahwa kita menetapkan nilai ke indeks dan bukan ke variabel. Indeks kemudian terkait erat dengan nilai itu, kecuali jika diberi nilai baru. Oleh karena itu perintah echo akan mencetak:

p2 element: Inkjet Printer

Kita dapat memverifikasi bahwa $p1 dan $p2 adalah tipe larik yang berbeda, karena kedua perintah berikut, ketika ditambahkan ke kode, akan menyebabkan indeks tidak terdefinisi atau kesalahan offset tidak terdefinisi, karena pengidentifikasi larik untuk masing-masing salah:

```
echo $p1['inkjet']; // Undefined index
echo $p2[3]; // Undefined offset
```

Foreach ... sebagai Loop

Pembuat PHP telah berusaha keras untuk membuat bahasa ini mudah digunakan. Jadi, tidak puas dengan struktur loop yang sudah disediakan, mereka menambahkan satu lagi khusus untuk array: foreach ... as loop. Dengan menggunakan ini, kita dapat menelusuri semua item dalam array, satu per satu, dan melakukan sesuatu dengannya. Prosesnya dimulai dengan item pertama dan diakhiri dengan item terakhir, jadi kita bahkan tidak perlu tahu berapa banyak item yang ada dalam array.

***Contoh Berjalan melalui array numerik menggunakan* foreach ... as**

```
?php
    $paper = array("Copier", "Inkjet", "Laser", "Photo");
    $j = 0;
    foreach($paper as $item)
    {
        echo "$j: $item";
        ++$j;
    }
?>
```

Ketika PHP menemukan pernyataan foreach, PHP mengambil item pertama dari array dan menempatkannya di variabel setelah kata kunci as; dan setiap aliran kontrol waktu kembali ke foreach, elemen array berikutnya ditempatkan di kata kunci as. Dalam hal ini, variabel $item diatur ke masing-masing dari empat nilai secara bergantian dalam array $paper. Setelah semua nilai telah digunakan, eksekusi loop berakhir.

***Contoh Berjalan melalui array asosiatif menggunakan* foreach ... as**

```
<?php
    $paper = array('copier' => "Copier & Multipurpose",
                   'inkjet' => "Inkjet Printer",
                   'laser' => "Laser Printer",
                   'photo' => "Photographic Paper");
    foreach($paper as $item => $description)
    echo "$item: $description";
?>
```

Ingat bahwa array asosiatif tidak memerlukan indeks numerik, jadi variabel $j tidak digunakan dalam contoh ini. Sebagai gantinya, setiap item dari array $paper dimasukkan ke dalam pasangan kunci/nilai variabel $item dan $description, dari mana mereka di print. Hasil yang ditampilkan dari kode ini adalah sebagai berikut:

```
copier: Copier & Multipurpose
inkjet: Inkjet Printer
laser: Laser Printer
photo: Photographic Paper
```

***Contoh Berjalan melalui array asosiatif menggunakan* each *dan* list**

```
<?php
    $paper = array('copier' => "Copier & Multipurpose",
                   'inkjet' => "Inkjet Printer",
```

```
                    'laser' => "Laser Printer",
                    'photo' => "Photographic Paper");
    while (list($item, $description) = each($paper))
    echo "$item: $description";
?>
```

Dalam contoh ini, while loop diatur dan akan terus perulangan sampai fungsi each mengembalikan nilai FALSE. Fungsi each bertindak seperti foreach: mengembalikan array yang berisi pasangan kunci/nilai dari array $paper dan kemudian memindahkan pointer bawaannya ke pasangan berikutnya dalam array itu. Ketika tidak ada lagi pasangan untuk dikembalikan, masing-masing mengembalikan FALSE. Fungsi list menggunakan larik sebagai argumennya (dalam hal ini, pasangan kunci/nilai masing-masing dikembalikan oleh fungsi) dan kemudian menetapkan nilai larik ke variabel yang tercantum dalam tanda kurung. Kita dapat melihat bagaimana fungsi list bekerja sedikit lebih jelas dalam Contoh dibawah ini, di mana sebuah array dibuat dari dua string Sarwo dan Edy kemudian diteruskan ke fungsi list, yang menetapkan string tersebut sebagai nilai ke variabel $a dan $b.

***Contoh Menggunakan fungsi* list**

```
<?php
    list($a, $b) = array('Sarwo', 'Edy');
    echo "a=$a b=$b";
?>
```

Output dari kode ini adalah :

```
a=Sarwo b=Edy
```

Jadi kita dapat memilih saat berjalan melalui array. Gunakan foreach ... untuk membuat loop yang mengekstrak nilai ke variabel mengikuti as, atau gunakan fungsi each dan buat sistem loop kita sendiri.

Array Multidimensi

Fitur desain sederhana dalam sintaks array PHP memungkinkan untuk membuat array lebih dari satu dimensi. Faktanya, mereka dapat memiliki dimensi sebanyak yang kita suka (walaupun ini adalah aplikasi langka yang lebih dari tiga). Fitur itu memungkinkan untuk memasukkan seluruh array sebagai bagian dari yang lain, dan untuk dapat terus melakukannya, seperti sajak lama: “Kutu besar memiliki kutu kecil di punggungnya untuk menggigit mereka. Kutu kecil memiliki kutu yang lebih kecil” dan seterusnya.

Contoh Membuat array asosiatif multidimensi

```
<?php
    $products = array(
        'paper' => array(
            'copier' => "Copier & Multipurpose",
            'inkjet' => "Inkjet Printer",
            'laser' => "Laser Printer",
            'photo' => "Photographic Paper"),
        'pens' => array(
            'ball' => "Ball Point",
            'hilite' => "Highlighters",
```

```
            'marker' => "Markers"),
        'misc' => array(
            'tape' => "Sticky Tape",
            'glue' => "Adhesives",
            'clips' => "Paperclips"
        )
    );
    echo "<prev>";
    foreach($products as $section => $items)
        foreach($items as $key => $value)
            echo "$section:\t$key\t($value)";
    echo "/prev>";
?>
```

Sekarang kode mulai berkembang, saya telah mengganti nama beberapa elemen. Misalnya, karena larik sebelumnya $paper sekarang hanya merupakan subbagian dari larik yang lebih besar, larik utama sekarang disebut $product. Dalam larik ini, ada tiga item — papper, pen, dan misc — masing-masing berisi larik lain dengan pasangan kunci/nilai. Jika perlu, subarray ini bisa berisi array lebih lanjut. Misalnya, di bawah ball mungkin ada banyak jenis dan warna pulpen yang tersedia di toko online. Tetapi untuk saat ini, saya telah membatasi kode hingga kedalaman hanya dua.

Setelah data array ditetapkan, saya menggunakan sepasang foreach ... sebagai loop untuk mencetak berbagai nilai. Loop luar mengekstrak bagian utama dari tingkat atas array, dan loop dalam mengekstrak pasangan kunci/nilai untuk kategori dalam setiap bagian. Selama kita ingat bahwa setiap level array bekerja dengan cara yang sama (ini adalah pasangan kunci/nilai), kita dapat dengan mudah menulis kode untuk mengakses elemen apa pun di level mana pun. Pernyataan echo menggunakan karakter escape PHP \t, yang menampilkan tab. Meskipun tab biasanya tidak signifikan bagi browser web, saya membiarkannya digunakan untuk tata letak dengan menggunakan tag <pre> ... </pre>, yang memberi tahu browser web untuk memformat teks sebagai terformat dan monospasi, dan bukan untuk abaikan karakter spasi putih seperti tab dan umpan baris. Output dari kode ini terlihat seperti berikut:

```
paper: copier (Copier & Multipurpose)
paper: inkjet (Inkjet Printer)
paper: laser (Laser Printer)
paper: photo (Photographic Paper)
pens: ball (Ball Point)
pens: hilite (Highlighters)
pens: marker (Markers)
misc: tape (Sticky Tape)
misc: glue (Adhesives)
misc: clips (Paperclips)
```

Kita dapat langsung mengakses elemen tertentu dari array menggunakan tanda kurung siku, seperti ini:

```
echo $products['misc']['glue'];
```

yang menampilkan nilai Adhesives. Kita juga dapat membuat array multidimensi numerik yang diakses secara langsung oleh indeks identifikator alfanumerik. Contoh membuat papan untuk permainan catur dengan bidak di posisi awalnya.

Contoh Membuat array numerik multidimensi

```
<?php
   $chessboard = array(
          array('r', 'n', 'b', 'q', 'k', 'b', 'n', 'r'),
          array('p', 'p', 'p', 'p', 'p', 'p', 'p', 'p'),
          array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
          array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
          array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
          array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
          array('P', 'P', 'P', 'P', 'P', 'P', 'P', 'P'),
          array('R', 'N', 'B', 'Q', 'K', 'B', 'N', 'R')
   );
   echo "<pre>";
   foreach($chessboard as $row)
   {
       foreach ($row as $piece)
           echo "$piece ";
       echo "<br>";
   }
   echo "</pre>";
?>
```

Dalam contoh ini, huruf kecil mewakili potongan hitam dan huruf besar putih. Kuncinya adalah r = rook (benteng), n = knight (ksatria), b = bishop (uskup), k = king (raja), q = queen (ratu), dan p = pion. Sekali lagi, sepasang foreach... saat loop berjalan melalui array dan menampilkan isinya. Loop luar memproses setiap baris menjadi variabel $row, yang merupakan array, karena array $chessboard menggunakan subarray untuk setiap baris. Loop ini memiliki dua pernyataan di dalamnya, jadi kurung kurawal mengapitnya. Loop dalam kemudian memproses setiap kotak dalam satu baris, mengeluarkan karakter ($ piece) yang disimpan di dalamnya, diikuti dengan spasi (untuk mengkuadratkan hasil print). Loop ini memiliki satu pernyataan, jadi kurung kurawal tidak diperlukan untuk mengapitnya. Tag <pre> dan </pre> memastikan bahwa output ditampilkan dengan benar, seperti ini:

```
r n b q k b n r
p p p p p p p p

P P P P P P P P
R N B Q K B N R
```

Kita juga dapat langsung mengakses elemen apa pun dalam array ini menggunakan tanda kurung siku, seperti ini:

```
echo $chessboard[7][3];
```

Pernyataan ini menampilkan huruf besar Q, elemen kedelapan ke bawah dan keempat sepanjang (mengingat bahwa indeks array dimulai dari 0, bukan 1).

Menggunakan Fungsi Array

Kita telah melihat daftar dan setiap fungsi, tetapi PHP hadir dengan banyak fungsi lain untuk menangani array. Daftar lengkapnya ada di http://tinyurl.com/arraysinphp. Namun, beberapa fungsi ini sangat mendasar sehingga perlu meluangkan waktu untuk melihatnya di sini.

is_array

Array dan variabel berbagi namespace yang sama. Ini berarti kita tidak dapat memiliki variabel string yang disebut $fred dan array yang juga disebut $fred. Jika kita ragu dan kode kita perlu memeriksa apakah suatu variabel adalah array, kita dapat menggunakan fungsi is_array seperti ini:

```
echo (is_array($fred)) ? "Is an array" : "Is not an array";
```

Perhatikan bahwa jika $fred belum diberi nilai, pesan variabel yang tidak ditentukan akan dibuat.

count

Meskipun fungsi each dan foreach ... sebagai struktur loop adalah cara terbaik untuk menelusuri isi array, terkadang kita perlu tahu persis berapa banyak elemen yang ada di array Anda, terutama jika kita akan mereferensikannya secara langsung. Untuk menghitung semua elemen di tingkat atas array, gunakan perintah seperti berikut:

```
echo count($fred);
```

Jika kita ingin mengetahui berapa banyak elemen yang ada dalam array multidimensi, kita dapat menggunakan pernyataan seperti:

```
echo count($fred, 1);
```

Parameter kedua adalah opsional dan mengatur mode yang akan digunakan. Itu harus berupa 0 untuk membatasi penghitungan hanya ke tingkat atas, atau 1 untuk memaksa penghitungan rekursif dari semua elemen subarray juga.

sort

Penyortiran sangat umum sehingga PHP menyediakan fungsi bawaan. Dalam bentuknya yang paling sederhana, kita akan menggunakannya seperti ini:

```
sort($fred);
```

Tidak seperti beberapa fungsi lainnya, sort akan bertindak langsung pada array yang disediakan daripada mengembalikan array baru dari elemen yang diurutkan. Sebagai gantinya, ia mengembalikan TRUE pada keberhasilan dan FALSE pada kesalahan dan juga mendukung beberapa flag, tetapi dua utama yang kita mungkin ingin menggunakan penyortiran paksa untuk dibuat baik secara numerik atau sebagai string, seperti ini:

```
sort($fred, SORT_NUMERIC);
sort($fred, SORT_STRING);
```

Kita juga dapat mengurutkan array dalam urutan terbalik menggunakan fungsi rsort, seperti ini:

```
rsort($fred, SORT_NUMERIC);
rsort($fred, SORT_STRING);
```

shuffle

Mungkin ada saat-saat ketika kita membutuhkan elemen array untuk diurutkan secara acak, seperti saat kita membuat permainan kartu remi:

```
shuffle($cards);
```

Seperti sort, shuffle bertindak langsung pada array yang disediakan dan mengembalikan TRUE jika berhasil atau FALSE jika error.

explode

Ini adalah fungsi yang sangat berguna yang dapat digunakan untuk mengambil string yang berisi beberapa item yang dipisahkan oleh satu karakter (atau string karakter) dan kemudian menempatkan masing-masing item ini ke dalam array. Salah satu contoh praktis adalah membagi kalimat menjadi larik yang berisi semua kata-katanya.

Contoh* explode *string ke dalam array menggunakan spasi

```
<?php
    $temp = explode(' ', "Ini adalah sebuah kalimay dengan enam kata ");
    print_r($temp);
?>
```

Contoh ini mencetak yang berikut (pada satu baris saat dilihat di browser):

```
Array (
[0] => Ini
[1] => adalah
[2] => sebuah
[3] => kalimat
[4] => dengan
[5] => enam
[6] => kata
)
```

Contoh Meledak string yang dibatasi dengan *** ke dalam array

```
<?php
    $temp = explode('***', "A***sentence***with***asterisks");
    print_r($temp);
?>
```

Kode dalam contoh diatas mencetak ini:

```
Array
(
    [0] => sebuah
    [1] => kalimat
    [2] => dengan
```

```
    [3] => tanda bintang
)
```

extract

Terkadang akan lebih mudah untuk mengubah pasangan kunci/nilai dari array menjadi variabel PHP. Salah satunya mungkin saat kita memproses variabel $_GET atau $_POST seperti yang dikirim ke skrip PHP dengan formulir. Saat formulir dikirimkan melalui Web, server web membongkar variabel ke dalam larik global untuk skrip PHP. Jika variabel dikirim menggunakan metode GET, variabel tersebut akan ditempatkan dalam array asosiatif yang disebut $_GET; jika dikirim menggunakan POST, mereka akan ditempatkan dalam array asosiatif yang disebut $_POST. Kita dapat, tentu saja, menelusuri susunan asosiatif seperti itu dengan cara yang ditunjukkan dalam contoh sejauh ini. Namun, terkadang kita hanya ingin menyimpan nilai yang dikirim ke variabel untuk digunakan nanti. Dalam hal ini, kita dapat meminta PHP melakukan pekerjaan secara otomatis untuk Anda:

```
extract($_GET);
```

Jadi, misalnya, jika parameter string kueri q dikirim ke skrip PHP bersama dengan nilai terkait Hi there, variabel baru bernama $q akan dibuat dan diberi nilai itu. Namun, berhati-hatilah dengan pendekatan ini, karena jika ada variabel yang diekstraksi bertentangan dengan variabel yang telah kita tetapkan, nilai kita yang ada akan ditimpa. Untuk menghindari kemungkinan ini, kita dapat menggunakan salah satu dari banyak parameter tambahan yang tersedia untuk fungsi ini, seperti ini:

```
extract($_GET, EXTR_PREFIX_ALL, 'fromget');
```

Dalam hal ini, semua variabel baru akan dimulai dengan string awalan yang diberikan diikuti dengan garis bawah, jadi $q akan menjadi $fromget_q. Saya sangat menyarankan kita menggunakan versi fungsi ini saat menangani array $_GET dan $_POST, atau array lain yang kuncinya dapat dikontrol oleh pengguna, karena pengguna jahat dapat mengirimkan kunci yang dipilih dengan sengaja untuk menimpa nama variabel yang umum digunakan dan kompromi situs Anda.

compact

Ada juga saat kita ingin menggunakan compact, kebalikan dari ekstrak, untuk membuat array dari variabel dan nilainya. Contoh dibawah ini menunjukkan bagaimana kita dapat menggunakan fungsi ini.

Contoh Menggunakan fungsi compact

```
<?php
    $fname = "Doctor";
    $sname = "Who";
    $planet = "Gelliea";
    $system = "Gallileo";
    $constellation = "Kasterborous";
    $contact = compact('fname', 'sname', 'planet', 'system', 'constellation');
    print_r($contact);
?>
```

Hasil larinya adalah:

```
Array
```

```
(
    [fname] => Doctor
    [sname] => Who
    [planet] => Gallilea
    [system] => Gallileo
    [constellation] => Kasterborous
)
```

Perhatikan bagaimana kompak mengharuskan nama variabel yang akan diberikan dalam tanda kutip, tidak didahului dengan simbol $. Ini karena compact sedang mencari daftar nama variabel.

Contoh Menggunakan compact untuk membantu debugging

```
<?php
    $j = 23;
    $temp = "Hello";
    $address = "Jalan Siliwangi";
    $age = 61; print_r(compact(explode(' ', 'j temp address age')));
?>
```

Cara bekerjanya dengan fungsi blast untuk mengekstrak semua kata dari string ke dalam array, yang kemudian diteruskan ke fungsi compact, yang pada gilirannya mengembalikan array ke print_r, yang akhirnya menunjukkan isinya. Jika kita menyalin dan menempelkan baris kode print_r, kita hanya perlu mengubah variabel yang disebutkan di sana untuk mencetak cepat sekelompok nilai variabel. Dalam contoh ini, outputnya adalah:

```
Array
(
    [j] => 23
    [temp] => Hello
    [address] => Jalan Siliwangi
    [age] => 61
)
```

reset

Ketika foreach ... sebagai konstruk atau setiap fungsi berjalan melalui array, ia menyimpan pointer PHP internal yang mencatat elemen array mana yang harus dikembalikan berikutnya. Jika kode kita perlu kembali ke awal array, kita dapat mengeluarkan reset, yang juga mengembalikan nilai elemen tersebut. Contoh cara menggunakan fungsi ini adalah:

```
reset($fred); // Throw away return value
$item = reset($fred); // Keep first element of the array in $item
```

end

Seperti reset, kita dapat memindahkan pointer array internal PHP ke elemen terakhir dalam array menggunakan fungsi akhir, yang juga mengembalikan nilai elemen, dan dapat digunakan seperti dalam contoh berikut:

```
end($fred); $item = end($fred);
```

Menggunakan Array

Array adalah variabel yang kompleks. Array menyimpan sekelompok nilai di bawah satu nama variabel, dan berguna untuk menyimpan nilai terkait. Misalnya, kita dapat menyimpan informasi tentang bunga (seperti varietas, warna, dan biaya) dalam satu larik bernama $flowerinfo. Informasi dalam array dapat ditangani, diakses, dan dimodifikasi dengan mudah. Misalnya, PHP memiliki beberapa metode untuk mengurutkan array. Bagian berikut memberi kita lowdown pada array.

Membuat array

Cara paling sederhana untuk membuat array adalah dengan menetapkan nilai ke variabel dengan tanda kurung siku ([ ]) di akhir namanya. Misalnya, dengan asumsi bahwa kita belum mereferensikan $cities pada titik sebelumnya dalam skrip, pernyataan berikut membuat array yang disebut $cities:

```
$cities[1] = “Jakarta”;
```

Pada titik ini, array bernama $cities telah dibuat dan hanya memiliki satu nilai: Jakarta. Selanjutnya, kita menggunakan pernyataan berikut:

```
$kota[2] = “Semarang”;
$kota[3] = “Salatiga”;
```

Sekarang array $cities berisi tiga nilai: Jakarta, Semarang, dan Salatiga.

Array dapat dilihat sebagai daftar pasangan kunci/nilai. Setiap pasangan kunci/nilai disebut elemen. Untuk mendapatkan nilai tertentu, kita menentukan kunci dalam tanda kurung. Dalam larik sebelumnya, kuncinya adalah angka — 1, 2, dan 3. Namun, kita juga bisa menggunakan kata untuk kunci. Misalnya, pernyataan berikut membuat larik ibu kota negara:

```
$capitals[‘ID] = “Jakarta”;
$capitals[‘TH] = “Bangkok”;
$capitals[KR] = “Seoul”;
```

Kita dapat menggunakan pintasan daripada menulis pernyataan tugas terpisah untuk setiap nomor. Salah satu jalan pintas menggunakan pernyataan berikut:

```
$city[] = “Jakarta”;
$city[] = “Semarang”;
$city[] = “Salatiga”;
```

Saat kita membuat larik menggunakan pintasan ini, nilainya secara otomatis diberikan kunci yang merupakan nomor seri, dimulai dengan angka 0. Misalnya, pernyataan berikut menampilkan Phoenix.

```
echo “$city[0]”;
```

Nilai pertama dalam array dengan indeks bernomor adalah 0 kecuali jika kita sengaja menyetelnya ke angka yang berbeda. Satu kesalahan umum saat bekerja dengan array adalah menganggap angka pertama sebagai 1 daripada 0.

Jalan pintas yang lebih baik adalah dengan menggunakan pernyataan berikut:

```
$City = array(“Jakarta”,”Semarang”,”Salatiga”);
```

Pernyataan ini membuat larik yang sama, dengan tombol bernomor, seperti pintasan sebelumnya. Kita dapat menggunakan pernyataan serupa untuk membuat array dengan kata-kata sebagai kunci. Misalnya, pernyataan berikut membuat larik ibu kota negara:

```
$capital = array( “ID” => “Jakarta”, “TH” => “Bangkok”,  “KR” => “Seoul” );
```

Melihat array

Kita dapat menggemakan nilai array seperti ini:

```
echo $capitals['TH'];
```

Jika kita menyertakan nilai larik dalam pernyataan echo yang lebih panjang yang diapit oleh tanda kutip ganda, kita mungkin perlu menyertakan nama nilai larik dalam kurung kurawal:

```
echo “Ibukota Thailand adalah {$capitals[‘TH’]}<br />”;
```

Kita dapat melihat struktur dan nilai larik apa pun dengan menggunakan pernyataan print_r atau var_dump. Untuk menampilkan array $capitals, gunakan salah satu pernyataan berikut:

```
print_r($kapital);
var_dump($kapital);
```

Pernyataan print_r ini memberikan output berikut:

Himpunan

```
(
 [ID] => Jakarta
 [TH] => Bangkok
 [KR] => Seoul
)
```

Pernyataan var_dump memberikan output berikut:

```
line(3) {
 [“ID”]=>
 string(10) "Jakarta"
 [“TH”]=>
 string(6) “Bangkok”
 [“KR”]=>
 string(5) "Seoul"
}
```

Output print_r menunjukkan kunci dan nilai untuk setiap elemen dalam array. Output var_dump menunjukkan tipe data, serta kunci dan nilai. Saat kita menampilkan output dari print_r atau var_dump pada halaman web, ini ditampilkan dengan HTML, yang berarti ditampilkan dalam satu baris panjang. Untuk melihat output di web dalam format berguna

yang kita jelaskan di sini, kirim tag HTML yang memberi tahu browser untuk menampilkan teks seperti yang diterima, tanpa mengubahnya, dengan menggunakan pernyataan berikut:

```
echo “<pra>”;
print_r($kapital);
echo “</pre>”;
```

Menghapus nilai dari array

Terkadang kita perlu menghapus elemen dari array sepenuhnya. Misalnya, kita memiliki larik berikut dengan lima elemen:

```
$cities[0] = “Jepara”;
$cities[1] = “Demak”;
$cities[2] = “Semarang”;
$cities[3] = “Kendal”;
$cities[4] = “Pekalongan”;
```

Sekarang kita memutuskan bahwa kita tidak ingin lagi memasukkan Kota Kendal, jadi untuk mencoba menghapus Kendal kita dapat menggunakan pernyataan berikut dari baris maka:

```
$cities[3] = “”;
```

Meskipun pernyataan ini menetapkan $cities[3] ke string kosong, pernyataan ini tidak menghapus elemen dari array. Kita masih memiliki larik dengan lima elemen, tetapi salah satu dari lima nilai tersebut kosong. Untuk benar-benar menghapus elemen dari array, kita perlu menghapusnya dengan pernyataan berikut:

```
tidak disetel($kota[3]);
```

Sekarang array kita hanya memiliki empat elemen di dalamnya, sebagai berikut:

```
$cities[0] = “Jepara”;
$kota[1] = “Demak”;
$cities[2] = “Semarang”;
$kota[4] = “Pekalongan”;
```

Mengurutkan array

Salah satu fitur array yang paling berguna adalah PHP dapat mengurutkannya untuk Anda. PHP awalnya menyimpan elemen array sesuai urutan pembuatannya. Jika kita menampilkan seluruh larik tanpa mengubah urutannya, elemen akan ditampilkan sesuai urutan saat kita membuatnya. Seringkali, kita ingin mengubah urutan ini. Misalnya, kita mungkin ingin menampilkan larik dalam urutan abjad berdasarkan nilai atau kunci. PHP dapat mengurutkan larik dalam berbagai cara. Untuk mengurutkan array yang memiliki angka sebagai kunci, gunakan fungsi sort, sebagai berikut:

```
sort($kota);
```

Pernyataan ini mengurutkan berdasarkan nilai dan menetapkan kunci baru yang merupakan nomor yang sesuai. Nilai diurutkan dengan angka terlebih dahulu, huruf besar berikutnya, dan huruf kecil terakhir. Misalnya, pertimbangkan array $cities yang dibuat di bagian sebelumnya:

```
$cities[0] = “Jepara”;
$cities[1] = “Demak”;
$cities[2] = “Semarang”;
```

Setelah pernyataan pengurutan berikut:

```
sort($cities);
```

array menjadi:

```
$cities[0] = “Jepara”;
$cities[1] = “Demak”;
$cities[2] = “Semarang”;
```

Jika kita menggunakan sort() untuk mengurutkan array dengan kata-kata sebagai kunci, kunci akan diubah menjadi angka, dan kunci kata akan dibuang. Untuk mengurutkan array yang memiliki kata untuk kunci, gunakan fungsi sort. Pernyataan ini mengurutkan huruf besar berdasarkan nilai dan menyimpan kunci asli untuk setiap nilai. Misalnya, pertimbangkan array ibukota negara yang dibuat di bagian sebelumnya:

```
$capitals[‘ID] = “Jakarta”;
$capitals[‘TH’] = “Bangkok”;
$capitals['KR'] = “Seoul”;
```

Setelah pernyataan asort berikut:

```
asort($kapital);
```

array menjadi:

```
$capitals[‘TH’] = “Bangkok”;
$capitals[‘ID’] = “Jakarta”;
$capitals['KR'] = “Seoul”;
```

Perhatikan bahwa kunci tetap dengan nilai ketika elemen disusun ulang. Sekarang elemen dalam urutan abjad, dan kunci negara yang benar masih dengan ibukota negara yang sesuai. Jika kuncinya adalah angka, nomornya sekarang akan berada dalam urutan yang berbeda. Sepertinya kita tidak ingin menggunakan asort pada array dengan angka sebagai kunci. Beberapa pernyataan sortir lainnya mengurutkan dengan cara lain. Tabel dibawah ini mencantumkan semua pernyataan sortir yang tersedia.

Tabel 2.4 Daftar pernyataan sortir

Pernyataan *Short*	Apa yang dilakukan
sort*($arrayname)*	Sort melalui nilai, tetapkan angka baru sebagai kunci
asort*($arrayname)*	Sort melalui nilai, simpan kunci sama

rsort*($arrayname)*	Sort melalui nilai yang diminta, tetapkan angka baru sebagai kunci
arsort*($arrayname)*	Sort melalui nilai yang dimininta, tetapkan angka baru sebagai kunci
ksort*($arrayname*)	Sort melalui kunci
krsort(*$arrayname*)	Sort melalui kunci pada permintaan,
usort*($arrayname*, functionname)	Sort melalui fungsi

Mendapatkan nilai dari array

Kita dapat mengambil nilai individual apa pun dalam array dengan mengaksesnya secara langsung, sebagai berikut:

```
$CApital = $capitals['ID'];
echo $Capital ;
```

Output dari pernyataan tersebut adalah:

```
Jakarta
```

Jika kita menggunakan elemen array yang tidak ada, pemberitahuan akan ditampilkan. Misalnya, kita menggunakan pernyataan berikut:

```
$CAcapital = $capitals['IDx'];
```

Jika array $capitals ada tetapi tidak ada elemen yang memiliki kunci IDx, kita akan melihat pemberitahuan berikut:

Pemberitahuan: Indeks tidak terdefinisi: IDx di **d:\testarray.php** pada baris **9**

Pemberitahuan tidak menyebabkan skrip berhenti. Pernyataan setelah pemberitahuan terus dijalankan. Tetapi karena tidak ada nilai yang dimasukkan ke dalam $CAcapital, setiap pernyataan echo berikutnya akan menggemakan ruang kosong. Kita dapat mencegah agar pemberitahuan tidak ditampilkan dengan menggunakan simbol @:

```
@$Capital = $capitals['IDx'];
```

Kita bisa mendapatkan beberapa nilai sekaligus dari larik menggunakan pernyataan daftar atau semua nilai dari larik dengan menggunakan pernyataan ekstrak. Fungsi daftar mendapatkan nilai dari array dan memasukkannya ke dalam variabel.
Pernyataan berikut termasuk pernyataan daftar:

```
$flowerInfo = array (“Mawar”, “merah”, 12.000);
list($firstvalue,$secondvalue) = $flowerInfo;
echo $secondvalue,”<br />";
echo $secondvalue,”<br />”;
```

Baris pertama membuat array $flowerInfo. Baris ketiga menyiapkan dua variabel bernama $firstvalue dan $secondvalue akan menyalin dua nilai pertama di $flowerInfo ke dalam dua variabel baru, seolah-olah kita telah menggunakan dua pernyataan:

```
$firstvalue=$flowerInfo[0];
```

```
$nilaikedua=$infobunga[1];
```

Nilai ketiga dalam $flowerInfo tidak disalin ke dalam variabel karena pernyataan daftar hanya menyertakan dua variabel. Output dari pernyataan echo adalah:

```
Mawar
merah
```

Kita dapat mengambil semua nilai dari array dengan kata-kata sebagai kunci dengan menggunakan ekstrak. Setiap nilai disalin ke dalam variabel bernama untuk kunci. Misalnya, array $flowerinfo dibuat sebagai berikut:

```
$flowerInfo = array ("variety"=>"Mawar", "warna"=>"merah",
"biaya"=>12.000);
```

Pernyataan berikut mendapatkan semua informasi dari $flowerInfo dan echo itu:

```
ekstract($flowerinfo);
echo "variasi adalah $variety; warna adalah $color; biaya adalah $cost";
```

Output dari pernyataan-pernyataan tersebut adalah:

```
varietas adalah Mawar; warnanya merah; biaya adalah 12.00;
```

Berjalan melalui array

Seringkali kita ingin melakukan sesuatu untuk setiap nilai dalam array. Kita mungkin ingin mengulang setiap nilai, menyimpan setiap nilai dalam database, atau menambahkan 6 ke setiap nilai dalam array. Dalam pembicaraan teknis, berjalan melalui setiap nilai dalam array, secara berurutan, adalah iterasi. Itu juga kadang-kadang disebut melintasi. Berikut adalah dua cara untuk berjalan melalui array:

- **Secara manual**: Memindahkan pointer dari satu nilai array ke nilai lainnya.
- **Menggunakan foreach**: Secara otomatis menelusuri array, dari awal hingga akhir, satu nilai pada satu waktu

Berjalan secara manual melalui array

Kita dapat menelusuri array secara manual dengan menggunakan pointer. Untuk melakukan ini, anggap array kita sebagai daftar. Bayangkan sebuah pointer menunjuk ke sebuah nilai dalam daftar. Pointer tetap pada nilai sampai kita memindahkannya. Setelah kita memindahkannya, itu tetap di sana sampai kita memindahkannya lagi. Kita dapat memindahkan penunjuk dengan petunjuk berikut:

- current*($arrayname):* Mengacu pada nilai saat ini di bawah pointer; tidak menggerakkan penunjuk.
- next*($arrayname):* Memindahkan pointer ke nilai setelah nilai saat ini.
- sebelumnya($arrayname): Memindahkan penunjuk ke nilai sebelum lokasi penunjuk saat ini.
- end*($arrayname):* Memindahkan pointer ke nilai terakhir dalam array.
- reset*($arrayname):* Memindahkan pointer ke nilai pertama dalam array.

Pernyataan berikut secara manual berjalan melalui array yang berisi ibukota negara:

```
$value = saat ini ($capitals);
```

```
echo “$value<br />”;
$value = berikutnya ($capitals);
echo “$value<br />”;
$value = berikutnya ($capitals);
echo “$value<br />”;
```

Kecuali kita telah memindahkan penunjuk sebelumnya, penunjuk itu terletak di elemen pertama saat kita mulai menelusuri larik. Jika kita berpikir bahwa penunjuk larik mungkin telah dipindahkan lebih awal dalam skrip atau jika output kita dari larik tampaknya dimulai di suatu tempat di tengah, gunakan pernyataan reset sebelum kita mulai berjalan, sebagai berikut:

```
reset($kapital);
```

Saat menggunakan metode ini untuk menelusuri larik, kita memerlukan pernyataan penetapan dan pernyataan echo untuk setiap nilai dalam larik — untuk masing-masing dari 50 status. Outputnya adalah daftar semua ibu kota negara.

Metode ini memberi kita fleksibilitas. Kita dapat menelusuri larik dengan cara apa pun — bukan hanya satu nilai pada satu waktu. Kita dapat bergerak mundur, langsung ke akhir, melewatkan setiap nilai lain dengan menggunakan dua pernyataan berikutnya berturut-turut, atau metode apa pun yang berguna. Namun, jika kita ingin menelusuri array dari awal hingga akhir, satu nilai pada satu waktu, PHP menyediakan foreach, yang melakukan persis apa yang kita butuhkan dengan jauh lebih efisien. foreach dijelaskan di bagian berikutnya.

Menggunakan foreach untuk berjalan melalui array

Pernyataan foreach berjalan melalui array satu nilai pada satu waktu. Kunci dan nilai larik saat ini dapat digunakan dalam blok pernyataan setiap kali blok dijalankan. Bentuk umumnya adalah:

```
foreach( $arrayname sebagai $keyname => $valuename )
{
 Block of statement;
}
```

Isi informasi berikut:

- *arrayname:* Nama larik yang kita lalui.
- *keyname:* Nama variabel tempat kita ingin menyimpan kunci. *keyname* adalah opsional. Jika kita meninggalkan $keyname =>, hanya nilai yang dimasukkan ke dalam variabel yang dapat digunakan di blok pernyataan.
- *valuename:* Nama variabel tempat kita ingin menyimpan nilainya.

Misalnya, pernyataan foreach berikut menelusuri contoh larik ibu kota negara dan menggemakan daftar:

```
$capitals = array(“ID” => “Jakarta”, “TH” => “Bangkok”,
 “KR” => “Seoul” );
ksort($kapitals);
foreach( $capitals sebagai $country => $city )
{
 echo “$kota, $country<br />”;
}
```

Pernyataan sebelumnya memberikan output halaman web berikut:

Jakarta, ID
Seoul, KR
Bangkok, TH

Kita dapat menggunakan baris berikut sebagai pengganti baris foreach pada pernyataan sebelumnya:

foreach($capital sebagai $city)

Saat menggunakan pernyataan foreach ini, hanya kota yang tersedia untuk output. Kita kemudian akan menggunakan pernyataan echo berikut:

```
echo “$kota<br />”;
```

Output dengan perubahan ini adalah:

Jakarta
Seoul
Bangkok

Ketika foreach mulai berjalan melalui array, itu memindahkan pointer ke awal array. Kita tidak perlu mengatur ulang array sebelum berjalan melaluinya dengan foreach

Menyimpan nilai dengan array multidimensi

Di bagian awal bab ini, kita menjelaskan array yang merupakan daftar tunggal pasangan kunci/nilai. Namun, pada beberapa kesempatan, kita mungkin ingin menyimpan nilai dengan lebih dari satu kunci. Misalnya, kita ingin menyimpan kota dan kabupaten, sebagai berikut:

```
$cities['AZ']['Semarang'] = Ungaran;
$cities['AZ']['Solo'] = Surakarta;
$cities['AZ']['DI Yogyakarta'] = Sleman;
$cities['OR']['Jakarta'] = Bintaro;
$cities['OR']['Jepara'] = Tahunan;
$cities['OR']['Kediri'] = pare;
```

Array jenis ini adalah array multidimensi karena seperti array dari array dengan struktur sebagai berikut:

$cities key	value		
		key	value
	AZ	Semarang	Ungaran
		Solo	Surakarta
	OR	Jakarta	Bintaro
		Jepara	Tahunan
		Kediri	Pare

$cities adalah array dua dimensi.

PHP juga dapat memahami array multidimensi yang memiliki kedalaman empat, lima, enam, atau lebih. Namun, orang cenderung sakit kepala ketika mereka mencoba memahami array yang lebih dari tiga level. Kemungkinan kebingungan meningkat ketika jumlah dimensi meningkat. Cobalah untuk menjaga agar array multidimensi kita dapat dikelola.

Kita bisa mendapatkan nilai dari array multidimensi dengan menggunakan prosedur yang sama yang kita gunakan dengan array satu dimensi. Misalnya, kita dapat mengakses nilai secara langsung dengan pernyataan ini:

```
$cities = $cities['AZ']['Surakarta']
```

Kita juga dapat menggemakan nilainya:

```
echo $cities['OR']['Kediri'];
```

Kita dapat menelusuri array multidimensi dengan menggunakan pernyataan foreach (dijelaskan di bagian sebelumnya). Kita memerlukan pernyataan foreach untuk setiap array. Satu pernyataan foreach berada di dalam pernyataan foreach lainnya. Menempatkan pernyataan di dalam pernyataan lain disebut *nesting*. Karena array dua dimensi, seperti $cities, berisi dua array, dibutuhkan dua pernyataan foreach untuk melewatinya. Pernyataan berikut mendapatkan nilai dari array multidimensi dan menampilkannya dalam tabel HTML:

```
foreach( $cities sebagai $state)
{
foreach( $state sebagai $county => $city )
    {
    echo “$city, $country country<br />”;
    }
}
```

2.9 MENGGUNAKAN TANGGAL DAN WAKTU

Tanggal dan waktu dapat menjadi elemen penting dalam aplikasi database web. PHP memiliki kemampuan untuk mengenali tanggal dan waktu dan menanganinya secara berbeda dari string karakter biasa. Tanggal dan waktu disimpan oleh komputer dalam format yang disebut *Timestamp*. Namun, ini bukan format di mana kita ingin melihat tanggal. PHP mengubah tanggal dari notasi kita menjadi timestamp yang dipahami komputer dan dari timestamp menjadi format yang familiar bagi orang-orang. PHP menangani tanggal dan waktu dengan fungsi bawaan.

Format timestamp adalah Unix Timestamp, yang merupakan bilangan bulat yang merupakan jumlah detik dari 1 Januari 1970, 00:00:00 GMT (Greenwich Mean Time) hingga waktu yang diwakili oleh timestamp. Format ini memudahkan untuk menghitung waktu antara dua tanggal — cukup kurangi satu timestamp dari yang lain.

Pengaturan waktu lokal

Dengan rilis PHP 5.1, PHP menambahkan pengaturan untuk zona waktu lokal default ke php.ini. Jika kita tidak menetapkan zona waktu default, PHP akan menebak, yang terkadang menghasilkan GMT. Selain itu, PHP menampilkan pesan yang menyarankan kita untuk mengatur zona waktu lokal Anda. Untuk menetapkan zona waktu default, ikuti langkah-langkah berikut:

1. Buka php.ini di editor teks.
2. Gulir ke bawah ke bagian menuju [Date].
3. Cari pengaturan date.timezone =.

4. Jika baris diawali dengan titik koma (;), hilangkan titik koma.
5. Tambahkan kode zona waktu setelah tanda sama dengan.

Kita dapat melihat daftar kode zona waktu di Lampiran H manual online PHP di www.php.net/manual/en/timezones.php. Misalnya, kita dapat mengatur zona waktu default kita ke waktu Pasifik dengan pengaturan:

```
date.timezone = Indonesia/Kota_Semarang
```

Jika kita tidak memiliki akses ke file php.ini, kita dapat mengatur zona waktu default di setiap skrip yang hanya berlaku untuk skrip tersebut, sebagai berikut:

```
date_default_timezone_set("timezonecode");
```

Kita dapat melihat zona waktu mana yang saat ini menjadi zona waktu default kita dengan menggunakan pernyataan ini:

```
$def = date_default_timezone_get()
echo $def;
```

Memformat tanggal

Fungsi yang paling sering kita gunakan adalah date, yang mengubah tanggal atau waktu dari format timestamp menjadi format yang kita tentukan. Bentuk umumnya adalah:

```
$mydate = date(“format”,$timestamp);
```

$timestamp adalah variabel dengan timestamp yang tersimpan di dalamnya. Kode ini mengasumsikan bahwa kita sebelumnya menyimpan timestamp dalam variabel, yang mungkin kita lakukan menggunakan fungsi PHP (seperti yang akan kita jelaskan nanti di bagian ini). Jika $timestamp tidak disertakan, waktu saat ini diperoleh dari sistem operasi dan digunakan. Dengan demikian, kita bisa mendapatkan tanggal hari ini dengan yang berikut:

```
$today = date(“Y/m/d”);
```

Jika hari ini adalah 17 Agustus 2022, pernyataan ini kembali:

```
2022/08/17
```

Format adalah string yang menentukan format tanggal yang ingin kita simpan dalam variabel. Misalnya, format "y-m-d" mengembalikan 13-08-10, dan "M.d.Y" mengembalikan 10 Agustus 2013. Tabel dibawah ini mencantumkan beberapa simbol yang dapat kita gunakan dalam format string. (Untuk daftar lengkap simbol, lihat dokumentasi di www.php.net/manual/en/function.date.php.) Bagian tanggal dapat dipisahkan dengan tanda hubung (-), titik (.), garis miring (/), atau spasi.

Tabel 2.5 Simbol tanggal yang dapat digunakan dalam format string

Simbol	Makna	Contoh
F	Bulan dalan teks, tidak di singkat	Januari
M	Bulan dalam teks, disingkat	Jan
m	Bulan dalam angka dengan awalan nol	02, 12

n	Bulan dalam angka tanpa awalan nol	1, 12
d	Hari pada bulan, dua digit dengan awalan nol	01, 14
j	Hari pada bulan tanpa awalan nol	3,30

Menyimpan timestamp dalam variabel

Kita dapat menetapkan timestamp dengan tanggal dan waktu saat ini ke variabel dengan pernyataan berikut:

```
$today = time();
```

Cara lain untuk menyimpan timestamp saat ini adalah dengan pernyataan

```
$today = strtotime("today");
```

Kita dapat menyimpan timestamp tertentu dengan menggunakan strtotime dengan berbagai kata kunci dan singkatan yang mirip dengan bahasa Inggris. Misalnya, kita dapat membuat timestamp untuk 15 Juli 2022, sebagai berikut:

```
$importantDate = strtotime("Juli 15 2022");
```

Pernyataan strtotime mengenali kata-kata dan singkatan berikut:

- Nama bulan: Nama dan singkatan dua belas bulan
- Hari dalam seminggu: Tujuh hari dan beberapa singkatan
- Satuan waktu: tahun, bulan, dua minggu, minggu, hari, jam, menit, detik, pagi, sore
- Beberapa kata bahasa Inggris yang berguna: lalu, sekarang, terakhir, selanjutnya, ini, besok, kemarin
- Plus dan minus: + atau -
- Semua nomor
- Zona waktu: Misalnya, gmt (Greenwich Mean Time)

Kita dapat menggabungkan kata-kata dan singkatan dalam berbagai cara. Pernyataan berikut ini semua valid:

```
$importantDate = strtotime("besok"); #24 hours from now
$importantDate = strtotime("sekarang + 24 jam");
$importantDate = strtotime("Sabtu terakhir");
$importantDate = strtotime("8pm + 3 hari");
$importantDate = strtotime("2 minggu yang lalu"); # current time
$importantDate = strtotime("tahun depan gmt");
$importantDate = strtotime("ini jam 4 pagi"); #4 today
```

Jika kita ingin tahu berapa lama tanggal $importantDate, kita bisa kurangi dari $today. Misalnya:

```
$timeSpan = $today - $importantDate;
```

Ini memberi kita jumlah detik antara tanggal penting dan hari ini. Atau gunakan pernyataan:

```
$timeSpan =(($today - $importantDate)/60)/60
```

untuk mengetahui jumlah jam sejak tanggal penting.

2.10 MEMAHAMI PESAN KESALAHAN PHP (*PHP ERROR MESSAGE*)

PHP mencoba membantu ketika masalah muncul. Ini memberikan berbagai jenis pesan kesalahan dan peringatan dengan informasi sebanyak mungkin. Di sini kita memberi tahu kita tentang berbagai jenis pesan kesalahan tersebut dan memberi kita beberapa tip untuk mengatasinya. Kita juga memberi tahu kita cara menampilkan atau mematikan pesan kesalahan, dan cara menyimpan pesan kesalahan dalam file log.

Jenis pesan kesalahan PHP

PHP dapat menampilkan lima jenis pesan. Setiap jenis pesan menampilkan nama file tempat kesalahan ditemukan dan nomor baris tempat PHP mengalami masalah. Jenis kesalahan yang berbeda memberikan informasi tambahan dalam pesan kesalahan. Jenis-jenis pesan adalah:

- **Parse Error/Kesalahan penguraian**: Kesalahan penguraian adalah kesalahan sintaksis yang ditemukan PHP saat memindai skrip sebelum menjalankannya.
- **Fatal Error/Kesalahan fatal:** PHP mengalami kesalahan serius yang menghentikan eksekusi skrip.
- **Warning/Peringatan:** PHP melihat masalah, tetapi masalahnya tidak cukup serius untuk mencegah skrip berjalan.
- **Notice/Pemberitahuan:** PHP melihat kondisi yang mungkin error atau mungkin baik-baik saja.
- **Strict/Ketat:** Pesan ketat, ditambahkan dalam PHP 5, memperingatkan tentang standar pengkodean. Kita mendapatkan pesan ketat ketika kita menggunakan bahasa yang merupakan praktik pengkodean yang buruk atau telah digantikan oleh kode yang lebih baik.

Memahami Parse Error/kesalahan penguraian

Sebelum mulai menjalankan skrip, PHP memindai skrip untuk mencari kesalahan sintaks. Ketika menemukan kesalahan, ini akan menampilkan pesan kesalahan parse. Kesalahan parse adalah kesalahan fatal, mencegah skrip bahkan mulai berjalan. Kesalahan parse terlihat mirip dengan berikut ini:

Parse error: parse error, error, in c:\test.php on line 6

Seringkali, kita menerima pesan kesalahan ini karena kita lupa titik koma, tanda kurung, atau kurung kurawal. Kesalahan yang ditampilkan memberikan informasi sebanyak mungkin. Misalnya, berikut ini mungkin ditampilkan:

Parse error: parse error, unexpected T_ECHO, expecting ',' or ';',
in c:\test.php on line 6

Kesalahan ini berarti bahwa PHP menemukan pernyataan echo di mana ia mengharapkan koma atau titik koma, yang mungkin berarti kita lupa titik koma di akhir baris sebelumnya. T_ECHO adalah tanda. Token mewakili berbagai bagian dari bahasa PHP. Beberapa, seperti T_ECHO atau T_IF, cukup jelas. Lainnya lebih tidak jelas. Lihat lampiran token di manual online PHP (www.php.net/manual/en/tokens.php) untuk daftar token parser beserta artinya.

Memahami Fatal Error/kesalahan fatal

Pesan kesalahan fatal ditampilkan ketika PHP menemukan kesalahan serius selama eksekusi skrip yang mencegah skrip melanjutkan eksekusi. Skrip berhenti berjalan dan menampilkan pesan yang berisi informasi sebanyak mungkin untuk membantu kita mengidentifikasi masalah. Salah satu masalah yang menghasilkan pesan kesalahan fatal adalah memanggil fungsi yang tidak ada. Jika kita salah mengeja nama fungsi dalam skrip PHP, kita akan melihat pesan kesalahan fatal yang mirip dengan berikut ini:

Fatal error: Call to undefined function xxx() in **C:\Program Files\Apache Group\Apache2\htdocs\PHPandMySQL\info.php** on line **10**

Dalam hal ini, PHP tidak dapat menemukan fungsi bernama xxx yang kita panggil pada baris 10.

Kita menggunakan istilah kesalahan fatal untuk membedakan jenis kesalahan ini dari kesalahan lainnya. Namun, PHP hanya menyebutnya kesalahan (membingungkan). Kita tidak akan menemukan istilah kesalahan fatal dalam manual. Juga, kata kunci yang diperlukan untuk menampilkan jenis kesalahan ini adalah E_ERROR.

Memahami Warning/peringatan

Pesan peringatan ditampilkan saat skrip mengalami masalah tetapi masalahnya tidak cukup serius untuk mencegah skrip berjalan. Pesan peringatan tidak berarti bahwa skrip tidak dapat berjalan; skrip terus berjalan. Sebaliknya, pesan peringatan memberi tahu kita bahwa PHP percaya bahwa ada sesuatu yang mungkin salah. Kita harus mengidentifikasi sumber peringatan dan kemudian memutuskan apakah itu perlu diperbaiki. Biasanya demikian. Jika kita mencoba menyambung ke database MySQL dengan nama pengguna atau kata sandi yang tidak valid, kita akan melihat pesan peringatan berikut:

Warning: mysql_connect() [function.mysql-connect]: Access denied for user 'root'@'localhost' (using password: YES) in **C:\Program Files\Apache Group\Apache2\htdocs\test.php** on line **9**

Upaya untuk ***menyambung*** ke database gagal, tetapi skrip tidak berhenti berjalan. Itu terus mengeksekusi pernyataan PHP tambahan dalam skrip. Namun, karena pernyataan selanjutnya mungkin bergantung pada koneksi database yang dibuat, pernyataan selanjutnya tidak akan dijalankan dengan benar. Pernyataan ini perlu diperbaiki. Sebagian besar pernyataan yang menghasilkan pesan peringatan perlu diperbaiki.

Memahami Notice/pemberitahuan

Pemberitahuan ditampilkan ketika PHP melihat kondisi yang mungkin error atau mungkin baik-baik saja. Pemberitahuan, seperti peringatan, tidak menyebabkan skrip berhenti berjalan. Pemberitahuan jauh lebih kecil kemungkinannya daripada peringatan untuk menunjukkan masalah serius. Pemberitahuan hanya memberi tahu kita bahwa kita melakukan sesuatu yang tidak biasa dan untuk melihat kembali apa yang kita lakukan untuk memastikan bahwa kita benar-benar ingin melakukannya.

Salah satu alasan umum mengapa kita mungkin menerima pemberitahuan adalah bahwa kita mengulangi variabel yang tidak ada. Berikut adalah contoh dari apa yang mungkin kita lihat dalam contoh itu.

Notice: Undefined variable: age in **testing.php** on line **9**

Memahami pesan yang ketat

Pesan ketat memperingatkan tentang standar pengkodean. Mereka menunjukkan bahasa yang merupakan praktik pengkodean yang buruk atau telah digantikan oleh kode yang lebih baik. Jenis kesalahan ketat ditambahkan di PHP 5. Pesan ketat tidak menghentikan eksekusi skrip. Namun, mengubah kode kita sehingga kita tidak melihat pesan ketat apa pun akan membuat skrip lebih andal di masa mendatang. Beberapa bahasa yang disorot oleh pesan ketat mungkin akan dihapus seluruhnya di masa mendatang.

Beberapa pesan ketat merujuk pada fitur bahasa PHP yang sudah tidak digunakan lagi. Fungsi yang tidak digunakan lagi adalah fungsi lama yang telah digantikan oleh fungsi yang

lebih baru. Fungsi yang tidak digunakan lagi masih didukung, tetapi akan dihapus di masa mendatang. PHP mungkin menambahkan jenis kesalahan terpisah E_DEPRECATED untuk mengidentifikasi jenis kesalahan ini sehingga E_STRICT dan E_DEPRECATED pesan akan mengidentifikasi berbagai jenis masalah.

Menampilkan pesan kesalahan (Error Messages)

Kita dapat menangani pesan kesalahan dengan salah satu cara berikut:

- Tampilkan beberapa atau semua pesan kesalahan di halaman web Anda.
- Jangan tampilkan pesan kesalahan apa pun.
- Menekan satu pesan kesalahan.

Kita dapat memberi tahu PHP apakah akan menampilkan pesan kesalahan atau pesan kesalahan mana yang akan ditampilkan dengan pengaturan di file php.ini atau dengan pernyataan PHP di skrip Anda. Pengaturan di php.ini mengatur penanganan kesalahan untuk semua skrip Anda. Pernyataan dalam skrip mengatur penanganan kesalahan untuk skrip itu saja.

Mematikan pesan kesalahan

Pesan kesalahan ditampilkan di halaman web kita secara default. Menampilkan pesan kesalahan pada halaman web kita adalah risiko keamanan. Kita dapat mengaktifkan pesan kesalahan saat kita mengembangkan situs web, sehingga kita dapat memperbaiki kesalahan, tetapi ketika halaman web kita selesai dan siap untuk dilihat publik, kita harus mematikan pesan kesalahan. Kita dapat mematikan semua pesan kesalahan untuk semua skrip di situs web kita di file php.ini. Temukan pengaturan berikut:

```
display_errors = On
```

Ubah On ke Off.

Kita dapat menonaktifkan kesalahan dalam skrip individual dengan pernyataan berikut:

```
ini_set(“display_errors”,”off”);
```

Mengubah pengaturan tidak mengubah kesalahan dengan cara apa pun; itu hanya mengubah apakah pesan kesalahan ditampilkan. Kesalahan fatal masih menghentikan skrip; itu hanya tidak menampilkan pesan di halaman web. Salah satu cara untuk menangani pesan kesalahan adalah dengan mematikannya di php.ini dan mengaktifkannya di setiap skrip individu selama pengembangan. Kemudian, ketika situs web siap untuk dilihat publik, kita dapat menghapus pernyataan ini_set yang mengaktifkan pesan kesalahan.

Menampilkan pesan yang dipilih

Kita dapat menentukan jenis pesan kesalahan yang ingin kita tampilkan dengan pengaturan berikut di php.ini:

```
error_reporting =
```

Kita menggunakan salah satu dari beberapa kode untuk memberi tahu PHP pesan mana yang akan ditampilkan. Beberapa pengaturan yang mungkin adalah:

```
error_reporting = E_ALL | E_STRICT
error_reporting = 0
error_reporting = E_ALL & ~ E_NOTICE
```

Pengaturan pertama menampilkan E_ALL, yaitu semua kesalahan, peringatan, dan pemberitahuan kecuali aturan; dan E_STRICT, yang menampilkan pesan ketat. Pengaturan kedua tidak menampilkan pesan kesalahan. Pengaturan ketiga menampilkan semua pesan kesalahan kecuali batasan dan pemberitahuan, karena & ~ berarti "dan tidak". Kode lain yang dapat kita gunakan adalah E_WARNING, yang berarti semua peringatan, dan E_ERROR, yang berarti semua kesalahan runtime yang fatal.

Kita juga dapat mengatur jenis pesan yang akan ditampilkan untuk masing-masing skrip. Kita dapat menambahkan pernyataan ke skrip yang menetapkan tingkat pelaporan kesalahan hanya untuk skrip tersebut. Tambahkan pernyataan berikut di awal skrip:

```
error_reporting(errorSetting);
```

Misalnya, untuk melihat semua kesalahan kecuali ketat, gunakan yang berikut ini:

```
error_reporting(E_ALL);
```

Menekan satu pesan kesalahan

Kita dapat menghentikan tampilan satu pesan kesalahan dalam pernyataan PHP. Secara umum, ini bukan ide yang bagus. Kita ingin melihat pesan kesalahan kita dan memperbaiki masalah. Namun, terkadang, menekan satu pemberitahuan adalah metode paling sederhana untuk mencegah pesan yang tidak sedap dipandang ditampilkan di halaman web.

Kita dapat menghentikan tampilan pesan kesalahan dengan menempatkan tanda (@) di mana kita mengharapkan pesan kesalahan dihasilkan. Misalnya, @ dalam pernyataan berikut menekan pesan kesalahan:

```
echo @$news1;
```

Jika variabel $news1 belum disetel sebelumnya, pernyataan ini akan menghasilkan pemberitahuan berikut:

Notice: Undefined variable: news1 in C:\Program Files\Apache Group\Apache2\htdocs\PHPandMySQL\info.php on line 10

Namun, @ di depan nama variabel membuat pemberitahuan tidak ditampilkan. Fitur ini seharusnya jarang digunakan, tetapi dapat berguna dalam beberapa situasi.

Pesan Logging Error

Kita dapat menyimpan pesan kesalahan dalam file log. Ini menghasilkan catatan permanen dari kesalahan, apakah itu ditampilkan di halaman web atau tidak. Mencatat pesan memerlukan dua setelan:

- log_errors: Atur ini ke On atau Off untuk mengirim kesalahan ke file log.
- error_log: Tentukan nama file tempat kesalahan akan dicatat.

Logging Error

Kita dapat memberitahu PHP untuk mencatat kesalahan dengan pengaturan di php.ini. Temukan pengaturan berikut:

```
log_errors = Off
```

Ubah pengaturan ke On. Setelah kita menyimpan file php.ini yang diubah dan memulai ulang server web Anda, PHP mencatat kesalahan ke file teks. Kita dapat memberi tahu PHP ke mana harus mengirim kesalahan dengan pengaturan error_log yang dijelaskan di bagian berikutnya. Jika kita tidak menentukan file dengan pengaturan error_log, pesan kesalahan ditulis ke log kesalahan Apache, yang terletak di subdirektori log di direktori tempat Apache diinstal. Log kesalahan memiliki ekstensi file .err.

Kita dapat mencatat kesalahan untuk skrip individual dengan menyertakan pernyataan berikut di awal skrip:

```
ini_set("log_errors","On")
```

Pernyataan ini menetapkan pencatatan kesalahan untuk skrip ini saja.

Menentukan file log

Kita menentukan file tempat PHP mencatat pesan kesalahan dengan pengaturan di php.ini. Temukan pengaturannya:

```
;error_log = filename
```

Hapus titik koma dari awal baris. Ganti nama file dengan jalur dan nama file tempat kita ingin PHP mencatat pesan kesalahan, seperti:

```
error_log = "c:\php\logs\errs.log"
```

File yang kita tentukan tidak perlu ada. Jika tidak ada, PHP akan membuatnya. Setelah kita menyimpan file php.ini yang telah diedit dan memulai ulang server web Anda, pesan kesalahan dicatat dalam file yang ditentukan. Setiap pesan kesalahan dicatat pada baris terpisah, dengan tanggal dan waktu di awal baris. Kita dapat menentukan file log untuk skrip individual dengan menyertakan pernyataan berikut di awal skrip:

```
ini_set("error_log"," c:\php\logs\errs.log ");
```

Pernyataan ini menetapkan file log untuk skrip ini saja.

2.11 MENAMBAHKAN KOMENTAR KE SKRIP PHP ANDA

Komentar adalah catatan yang disematkan dalam skrip itu sendiri. Menambahkan komentar di skrip kita yang menjelaskan tujuan dan apa yang mereka lakukan sangat penting. Ini penting untuk faktor lotere — yaitu, jika kita memenangkan lotre dan melarikan diri ke kehidupan mewah di French Riviera, orang lain harus menyelesaikan aplikasi. Penerus kita perlu mengetahui apa yang seharusnya dilakukan oleh skrip kita dan bagaimana ia melakukan tugasnya. Sebenarnya, komentar juga bermanfaat bagi Anda. Kita mungkin perlu merevisi naskah tahun depan ketika detailnya sudah lama terkubur dalam pikiran kita di bawah pemikiran proyek yang lebih baru. Gunakan komentar secara bebas. PHP mengabaikan komentar; komentar adalah untuk manusia. Kita dapat menyematkan komentar di skrip kita di mana saja selama kita memberi tahu PHP bahwa itu adalah komentar. Format komentar adalah:

```
/* comment text
more comment text */
```

Komentar kita bisa sepanjang atau sesingkat yang kita butuhkan. Ketika PHP melihat kode yang menunjukkan awal komentar (/*), PHP mengabaikan semuanya sampai melihat kode yang menunjukkan akhir komentar (*/). Salah satu format yang mungkin untuk komentar di awal setiap skrip adalah sebagai berikut :

```
/* name:              catalog.php
* description:  Script that displays descriptions of
*                    products. The descriptions are stored
*                    in a database. The product descriptions
*                    are selected from the database based on
*                    the category the user entered into a form.
* written by: Lola Designer
* created: 2/1/13
* modified: 3/15/13
*/
```

Kita harus menggunakan komentar di seluruh skrip untuk menjelaskan apa yang dilakukan skrip. Komentar sangat penting ketika pernyataan skrip rumit. Gunakan komentar seperti berikut ini sesering mungkin:

```
/* Get the information from the database */
/* Check whether the customer is over 18 years old */
/* Add shipping charges to the order total */
```

PHP juga memiliki format komentar pendek. Kita dapat menentukan bahwa satu baris adalah komentar dengan menggunakan tanda pound (#) atau dua garis miring (//) dengan cara berikut:

```
# This is comment line 1
// This is comment line 2
```

Semua teks dari # atau // hingga akhir baris adalah komentar. Kita juga dapat menggunakan # atau // di tengah baris untuk menandai awal komentar. PHP akan mengabaikan semuanya mulai dari # atau // hingga akhir baris. Ini berguna untuk mengomentari pernyataan tertentu, seperti dalam contoh berikut:

```
$average = $orderTotal/$nItems; // compute average price
```

Terkadang kita ingin menekankan sebuah komentar. Format berikut membuat komentar sangat terlihat:

```
########################################
##        Double-Check This Section       ##
########################################
```

Komentar PHP tidak disertakan dalam kode HTML yang dikirim ke browser pengguna. Pengguna tidak melihat komentar ini. Gunakan komentar sesering yang diperlukan dalam skrip untuk memperjelas. Namun, menggunakan terlalu banyak komentar adalah sebuah kesalahan. Jangan mengomentari setiap baris atau semua yang kita lakukan dalam skrip. Jika

skrip kita terlalu penuh dengan komentar, komentar penting bisa hilang di labirin. Gunakan komentar untuk memberi label pada bagian dan menjelaskan kode yang tidak biasa atau rumit — bukan kode yang jelas.

2.12 CASTING IMPLISIT DAN EKSPLISIT

PHP adalah bahasa yang diketik secara longgar yang memungkinkan kita untuk mendeklarasikan variabel dan tipenya hanya dengan menggunakannya. Itu juga secara otomatis mengubah nilai dari satu jenis ke jenis lainnya kapan pun diperlukan. Ini disebut *casting implisit*. Namun, mungkin ada kalanya casting implisit PHP bukanlah yang kita inginkan. Dalam contoh berikut, perhatikan bahwa input ke pembagian adalah bilangan bulat. Secara default, PHP mengonversi output menjadi floating point sehingga dapat memberikan nilai yang paling tepat— 4,66 berulang.

Contoh ekspresi ini mengembalikan angka floating-point.

```
<?php
    $a = 56;
    $b = 12;
    $c = $a / $b;
    echo $c;
?>
```

Tetapi bagaimana jika kita ingin $c menjadi bilangan bulat? Ada berbagai cara di mana kita bisa mencapai ini, salah satunya adalah memaksa hasil $a/$b menjadi cast ke nilai integer menggunakan tipe cast integer (int), seperti ini:

```
$c = (int) ($a / $b);
```

Ini disebut *casting eksplisit*. Perhatikan bahwa untuk memastikan bahwa nilai seluruh ekspresi dilemparkan ke bilangan bulat, kita menempatkan ekspresi di dalam tanda kurung. Jika tidak, hanya variabel $a yang akan dimasukkan ke bilangan bulat — latihan yang sia-sia, karena pembagian dengan $b masih akan mengembalikan angka titik-mengambang.

Catatan: kita dapat secara eksplisit melakukan cast ke tipe yang ditunjukkan pada *tabel jenis cast* dibawah ini, tetapi kita biasanya dapat menghindari keharusan menggunakan cast dengan memanggil salah satu fungsi bawaan PHP. Misalnya, untuk mendapatkan nilai integer, kita bisa menggunakan fungsi intval. Seperti beberapa bagian lain dalam buku ini, bagian ini terutama untuk membantu kita memahami kode pihak ketiga yang mungkin kita temui.

Tabel 2.6 Jenis pemeran PHP

Jenis cast	Deskripsi
(int) (integer)	Cast ke bilangan bulat dengan menjatuhkan bagian desimal.
(bool) (boolean)	Transmisikan ke Boolean.
(float) (double) (real)	Cast ke nomor floating-point.
(string)	Dilemparkan ke string.
(array)	Transmisikan ke array.
(object)	Dilemparkan ke suatu objek.

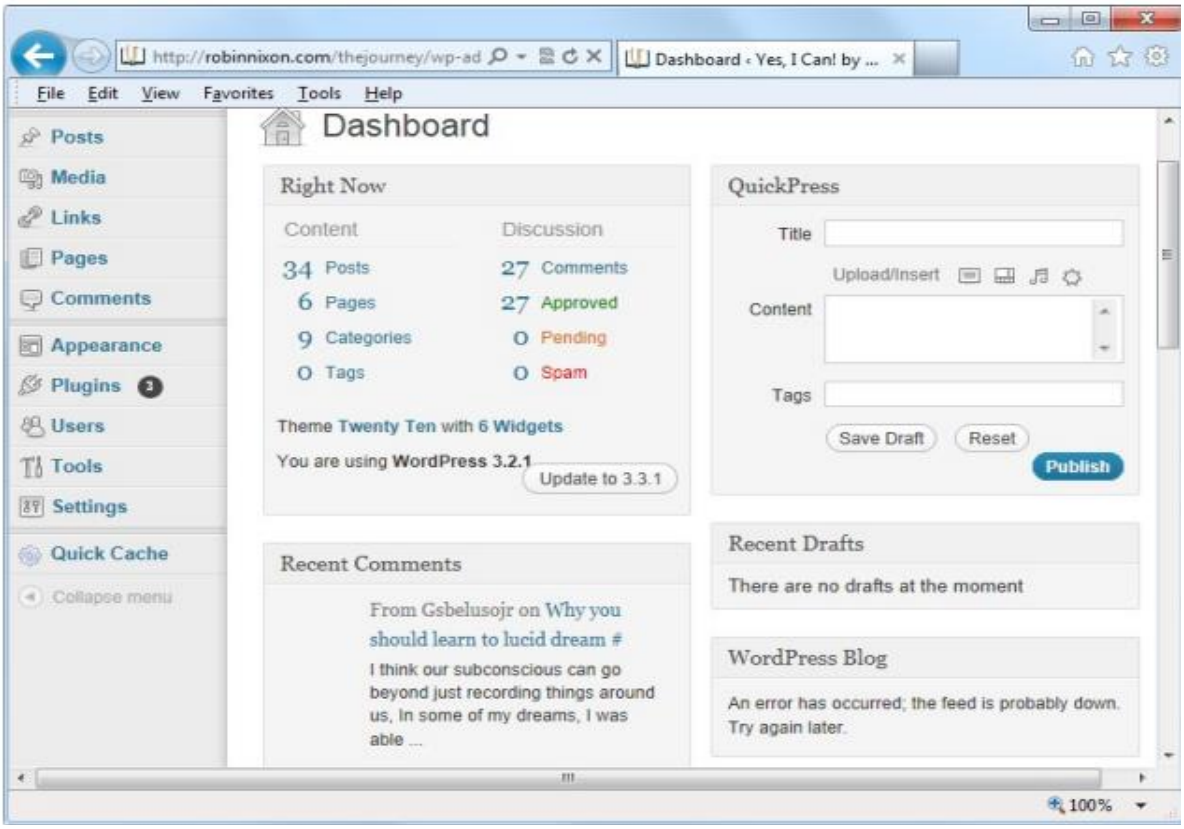
2.13 DYNAMIC LINKING PHP

Karena PHP adalah bahasa pemrograman, dan output darinya bisa sangat berbeda untuk setiap pengguna, mungkin saja seluruh situs web dijalankan dari satu halaman web PHP. Setiap kali pengguna mengklik sesuatu, detailnya dapat dikirim kembali ke halaman web yang sama, yang memutuskan apa yang harus dilakukan selanjutnya sesuai dengan berbagai cookie dan/atau detail sesi lain yang mungkin disimpannya. Meskipun dimungkinkan untuk membangun seluruh situs web dengan cara ini, itu tidak disarankan, karena source code kita akan tumbuh dan berkembang dan mulai menjadi berat, karena harus memperhitungkan setiap tindakan yang mungkin dilakukan pengguna. Alih-alih, jauh lebih masuk akal untuk membagi pengembangan situs web kita menjadi beberapa bagian. Misalnya, satu proses berbeda adalah mendaftar ke situs web, bersama dengan semua pemeriksaan yang diperlukan untuk memvalidasi alamat email, menentukan apakah nama pengguna sudah digunakan, dan seterusnya.

Modul kedua mungkin merupakan modul untuk memasukkan pengguna sebelum menyerahkannya ke bagian utama situs web Anda. Kemudian kita mungkin memiliki modul pesan dengan fasilitas bagi pengguna untuk meninggalkan komentar, modul yang berisi tautan dan informasi berguna, modul lain untuk memungkinkan pengunggahan gambar, dan banyak lagi. Selama kita telah membuat cara untuk melacak pengguna kita melalui situs web kita melalui cookie atau variabel sesi, kita dapat membagi situs web kita menjadi beberapa bagian yang masuk akal dari kode PHP , masing-masing mandiri, dan karena itu manjakan diri kita dengan masa depan yang jauh lebih mudah dengan mengembangkan setiap fitur baru dan memelihara fitur lama.

Dynamic Linking dalam Tindakan

Salah satu aplikasi berbasis PHP yang lebih populer di Web saat ini adalah platform blogging WordPress. Sebagai blogger atau pembaca blog, kita mungkin tidak menyadarinya, tetapi setiap bagian utama telah diberikan file PHP utamanya sendiri, dan seluruh fungsi umum yang dibagikan telah ditempatkan di file terpisah yang disertakan oleh halaman PHP utama. seperlunya. Seluruh platform disatukan dengan pelacakan sesi di belakang layar, sehingga kita hampir tidak tahu kapan kita beralih dari satu subbagian ke subbagian lainnya. Jadi, sebagai web developer, jika kita ingin men-tweak WordPress, mudah untuk menemukan file tertentu yang kita butuhkan, memodifikasinya, dan menguji serta men-debug-nya tanpa mengotak-atik bagian program yang tidak terhubung. Lain kali jika kita menggunakan WordPress, perhatikan bilah alamat browser Anda, terutama jika kita mengelola blog, dan kita akan melihat beberapa file PHP berbeda yang digunakannya.



Gambar 2.3 Platform blogging WordPress ditulis dalam PHP

BAB 3
MEMBANGUN SKRIP PHP

3.1 OBJEK DAN FUNGSI PHP

Persyaratan dasar dari setiap bahasa pemrograman termasuk tempat untuk menyimpan data, sarana untuk mengarahkan aliran program, dan beberapa bagian seperti evaluasi ekspresi, manajemen file, dan output teks. PHP memiliki semua ini, ditambah alat seperti yang lain dan yang lain untuk membuat hidup lebih mudah. Tetapi bahkan dengan semua ini di toolkit kita, pemrograman bisa menjadi canggung dan membosankan, terutama jika kita harus menulis ulang bagian dari kode yang sangat mirip setiap kali kita membutuhkannya. Di situlah fungsi dan objek masuk. Seperti yang kita duga, fungsi adalah sekumpulan pernyataan yang menjalankan fungsi tertentu dan—secara opsional—mengembalikan nilai. Kita dapat mengeluarkan bagian kode yang telah kita gunakan lebih dari sekali, menempatkannya ke dalam suatu fungsi, dan memanggil fungsi tersebut dengan nama ketika kita menginginkan kode tersebut. Fungsi memiliki banyak keunggulan dibandingkan kode sebaris yang bersebelahan. Misalnya, mereka:

- Libatkan lebih sedikit mengetik
- Kurangi sintaks dan kesalahan pemrograman lainnya
- Kurangi waktu pemuatan file program
- Kurangi waktu eksekusi, karena setiap fungsi hanya dikompilasi sekali, tidak peduli seberapa sering kita memanggilnya
- Terima argumen dan karena itu dapat digunakan untuk kasus umum maupun khusus

Objek mengambil konsep ini selangkah lebih maju. Sebuah objek menggabungkan satu atau lebih fungsi, dan data yang mereka gunakan, ke dalam satu struktur yang disebut kelas. Dalam bab ini, kita akan mempelajari semua tentang penggunaan fungsi, mulai dari mendefinisikan dan memanggilnya hingga meneruskan argumen bolak-balik. Dengan pengetahuan di bawah ikat pinggang Anda, kita akan mulai membuat fungsi dan menggunakannya di objek kita sendiri.

Fungsi PHP

PHP hadir dengan ratusan fungsi bawaan yang siap pakai, menjadikannya bahasa yang sangat kaya. Untuk menggunakan fungsi, panggil dengan nama. Misalnya, kita dapat melihat fungsi print beraksi di sini:

```
print("print is a pseudo-function");
```

Tanda kurung memberi tahu PHP bahwa kita merujuk ke suatu fungsi. Jika tidak, ia menganggap kita mengacu pada konstanta. Kita mungkin melihat peringatan seperti ini:

```
Notice: Use of undefined constant fname - assumed 'fname'
```

diikuti oleh string teks fname, dengan asumsi bahwa kita pasti ingin memasukkan string literal ke dalam kode Anda. (Segalanya menjadi lebih membingungkan jika sebenarnya ada konstanta bernama fname, dalam hal ini PHP menggunakan nilainya.)

Catatan: Sebenarnya print adalah fungsi semu, biasa disebut konstruk. Perbedaannya adalah kita dapat menghilangkan tanda kurung, sebagai berikut:

```
print "print doesn't require parentheses";
```

Kita harus meletakkan tanda kurung setelah fungsi lain yang kita panggil, meskipun fungsi tersebut kosong (yaitu, jika kita tidak memberikan argumen apa pun ke fungsi).

Fungsi dapat mengambil sejumlah argumen, termasuk nol. Misalnya, phpinfo, seperti yang ditunjukkan di sini, menampilkan banyak informasi tentang instalasi PHP saat ini dan tidak memerlukan argumen:

```
phpinfo();
```

Catatan: Fungsi phpinfo sangat berguna untuk mendapatkan informasi tentang instalasi PHP kita saat ini, tetapi informasi itu juga bisa sangat berguna bagi calon peretas. Oleh karena itu, jangan pernah meninggalkan panggilan ke fungsi ini dalam kode siap-web apa pun.

Gambar 3.1 Output dari fungsi phpinfo bawaan PHP

Contoh Tiga fungsi string

```
<?php
    echo strrev(" .dlrow olleH"); // Reverse string
    echo str_repeat("Hip ", 2); // Repeat string
    echo strtoupper("hooray!");  // String to uppercase
?>
```

Contoh ini menggunakan tiga fungsi string untuk menampilkan teks berikut:

```
Hello world.  Hip Hip HOORAY!
```

Seperti yang kita lihat, fungsi strrev membalik urutan karakter dalam string, str_repeat mengulangi string "Hip" dua kali (seperti yang dipersyaratkan oleh argumen kedua), dan strtoupper mengonversi "hore!" untuk huruf besar.

3.2 SKRIP PHP

Skrip PHP adalah serangkaian instruksi dalam file bernama dengan ekstensi yang memberitahu server web untuk mencari bagian PHP dalam file. (Ekstensi biasanya .php atau .phtml, tetapi dapat berupa apa saja yang diharapkan oleh server web.) PHP dimulai di bagian atas file dan mengeksekusi setiap instruksi, sesuai urutannya.

Instruksi, yang disebut pernyataan, bisa sederhana atau kompleks. Skrip yang membentuk aplikasi basis data web tidak sesederhana itu. Mereka dinamis dan berinteraksi dengan pengguna dan database. Akibatnya, skrip memerlukan pernyataan yang lebih kompleks.

Pernyataan kompleks mengeksekusi satu atau lebih blok pernyataan. Sebuah blok pernyataan terdiri dari sekelompok pernyataan sederhana yang diapit oleh kurung kurawal ({ dan }). PHP mencari kurung kurawal akhir dari blok terakhir dalam pernyataan kompleks.
Pernyataan kompleks:

- Pernyataan bersyarat: Pernyataan yang dijalankan hanya jika kondisi tertentu terpenuhi. Pernyataan kondisional PHP adalah pernyataan if dan switch.
- Loop: Pernyataan yang berulang kali mengeksekusi blok pernyataan. Empat jenis loop adalah foreach, for, while, dan do.. while loop.
- Fungsi: Pernyataan yang dapat digunakan kembali berkali-kali. Banyak tugas dilakukan di lebih dari satu bagian aplikasi. PHP memungkinkan kita untuk menggunakan kembali blok pernyataan dengan membuat fungsi.

Pernyataan bersyarat dan loop mengeksekusi blok pernyataan berdasarkan suatu kondisi. Artinya, jika suatu kondisi benar, blok pernyataan dijalankan. Jadi, untuk menggunakan pernyataan dan loop bersyarat, kita perlu mengatur kondisi.

Mendefinisikan Fungsi

Sintaks umum untuk suatu fungsi adalah:

```
function function_name([parameter [, ...]]) { // Statements
}
```

Saya akan menjelaskan semua tanda kurung siku, jika kita merasa bingung. Baris pertama sintaks menunjukkan bahwa:

- Definisi dimulai dengan kata fungsi.
- Nama mengikuti, yang harus dimulai dengan huruf atau garis bawah, diikuti dengan sejumlah huruf, angka, atau garis bawah.
- Tanda kurung diperlukan.
- Satu atau beberapa parameter, dipisahkan dengan koma, bersifat opsional.

Nama fungsi tidak peka huruf besar/kecil, sehingga semua string berikut dapat merujuk ke fungsi print: PRINT, Print, dan PRINT. Kurung kurawal pembuka memulai pernyataan yang akan dieksekusi saat kita memanggil fungsi; kurung kurawal yang cocok harus menutupnya. Pernyataan ini dapat mencakup satu atau lebih pernyataan kembali, yang memaksa fungsi untuk menghentikan eksekusi dan kembali ke kode panggilan. Jika sebuah nilai dilampirkan ke pernyataan pengembalian, kode panggilan dapat mengambilnya, seperti yang akan kita lihat selanjutnya.

3.3 MENYIAPKAN KONDISI

Kondisi adalah ekspresi yang diuji atau dievaluasi PHP untuk melihat apakah benar atau salah. Kondisi digunakan dalam pernyataan kompleks untuk menentukan apakah blok pernyataan sederhana harus dieksekusi. Untuk mengatur kondisi, kita membandingkan nilai. Berikut adalah beberapa pertanyaan yang dapat kita ajukan untuk membandingkan nilai untuk kondisi:

- Apakah dua nilai sama? Apakah nama belakang Mars sama dengan nama belakang Novi? Atau, apakah Josh berusia 35 tahun?
- Apakah satu nilai lebih besar atau lebih kecil dari yang lain? Apakah Josh lebih muda dari Joni? Atau, apakah rumah Mars berharga lebih dari satu miliar?
- Apakah string cocok dengan pola? Apakah nama Josh dimulai dengan huruf S? Apakah kode pos memiliki lima karakter numerik?

Kita juga dapat mengatur kondisi di mana kita mengajukan dua pertanyaan atau lebih. Misalnya, kita mungkin bertanya: Apakah Josh lebih tua dari Joni dan apakah Josh lebih muda dari Mars? Atau kita mungkin bertanya: Apakah hari ini hari Minggu dan hari ini cerah? Atau kita mungkin bertanya: Apakah hari ini hari Minggu atau hari ini Senin?

Mengembalikan Nilai

Mari kita lihat fungsi sederhana untuk mengubah nama lengkap seseorang menjadi huruf kecil dan kemudian menggunakan huruf besar untuk huruf pertama dari setiap nama. Kita telah melihat contoh fungsi strtoupper bawaan PHP pada contoh dibawah. Untuk fungsi kita saat ini, kita akan menggunakan mitranya, strtolower:

```
$lowered = strtolower("aNY # of Letters and Punctuation you WANT");echo $lowered;
```

Output dari percobaan ini adalah:

```
any # of letters and punctuation you want
```

Kita tidak ingin nama semua huruf kecil; kita ingin huruf pertama dari setiap nama dikapitalisasi. Untungnya, PHP juga menyediakan fungsi ucfirst yang menetapkan karakter pertama dari string menjadi huruf besar:

```
$ucfixed = ucfirst("any # of letters and punctuation you want");echo $ucfixed;
```

Outputnya adalah:

```
any # of letters and punctuation you want
```

Sekarang kita dapat melakukan desain program pertama kita: untuk mendapatkan kata dengan huruf awal yang dikapitalisasi, kita memanggil strtolower pada string terlebih dahulu, lalu ucfirst. Cara melakukannya adalah dengan membuat panggilan ke strtolower di dalam ucfirst. Mari kita lihat alasannya, karena penting untuk memahami urutan kode yang dievaluasi. Jika kita membuat panggilan sederhana ke fungsi print:

```
print(5-8);
```

Ekspresi 5-8 dievaluasi terlebih dahulu, dan hasilnya adalah -3. Jika ekspresi berisi fungsi, fungsi itu juga dievaluasi terlebih dahulu:

```
print(abs(5-8));
```

PHP melakukan beberapa hal dalam mengeksekusi pernyataan singkat itu:

1. Evaluasi 5-8 untuk menghasilkan -3.
2. Gunakan fungsi abs untuk mengubah -3 menjadi 3.
3. Ubah hasilnya menjadi string dan keluarkan menggunakan fungsi print.

Semuanya berfungsi, karena PHP mengevaluasi setiap elemen dari dalam ke luar. Prosedur yang sama beroperasi ketika kita memanggil yang berikut ini:

```
ucfirst(strtolower("aNY # of Letters and Punctuation you WANT"))
```

PHP meneruskan string kita ke strtolower dan kemudian ke ucfirst, menghasilkan (seperti yang telah kita lihat ketika kita bermain dengan fungsi secara terpisah):

```
any # of letters and punctuation you want
```

Sekarang mari kita definisikan sebuah fungsi yang mengambil tiga nama dan membuat masing-masing huruf kecil dengan huruf kapital awal.

Contoh Membersihkan nama lengkap

```
<?php
    echo fix_names("JOSEPH", "teguh", "SantoSO");
    function fix_names($n1, $n2, $n3)
    {
        $n1 = ucfirst(strtolower($n1));
        $n2 = ucfirst(strtolower($n2));
        $n3 = ucfirst(strtolower($n3));
        return $n1 . " " . $n2 . " " . $n3;
    }
?>
```

Kita mungkin menemukan diri kita menulis kode jenis ini, karena pengguna sering membiarkan tombol Caps Lock mereka, tanpa sengaja memasukkan huruf kapital di tempat yang salah, dan bahkan melupakan kapital sama sekali. Output dari contoh ini adalah:

```
Joseph Teguh Santoso
```

3.4 MEMBANDINGKAN NILAI

Kita dapat membandingkan angka atau string untuk melihat apakah mereka sama, apakah satu lebih besar dari yang lain, atau apakah mereka tidak sama. Kita membandingkan nilai dengan operator perbandingan. PHP mengevaluasi perbandingan dan mengembalikan true atau false. Sebagai contoh, berikut ini adalah perbandingan sederhana:

```
$result = $a == $b;
```

Operator perbandingan == memeriksa apakah dua nilai sama. Jika $a dan $b sama, $result diberikan nilai Boolean yang benar. Jika $a dan $b tidak sama, $result diberikan false. Jadi, $a == $b adalah kondisi sederhana yang bernilai benar atau salah. PHP menawarkan beberapa

operator perbandingan yang dapat kita gunakan untuk membandingkan nilai. Tabel dibawah ini menunjukkan operator pembanding ini.

Tabel 3.1 Operator Pembanding

Operator	Maksud
==	Apakah dua nilai itu nilainya sama?
===	Apakah dua nilai sama diantara nilai dan jenis data?
>	Apakah nilai pertama lebih besar dari nilai kedua?
>=	Apakah nilai pertama lebih besar atau sama dengan nilai kedua
<	Apakah nilai pertama lebih kecil dari nilai kedua?
<=	Apakah nilai pertama lebih kecil atau sama dengan nilai kedua?
!=, <>	Apakah dua nilai tidak sama pada setiap niliai lainnya?
!==	Apakah nilai tidak sama satu sama lain pada nilai lainnya atau tipe data?

Kita dapat membandingkan angka dan string. String dibandingkan menurut abjad, dengan semua karakter huruf besar datang sebelum karakter huruf kecil. Misalnya, SS datang sebelum Sa. Karakter tanda baca juga memiliki urutan, dan satu karakter dapat ditemukan lebih besar dari karakter lain. Namun, membandingkan koma dengan titik tidak memiliki banyak nilai praktis.

String dibandingkan berdasarkan kode ASCII mereka. Dalam rangkaian karakter ASCII, setiap karakter diberi kode ASCII yang sesuai dengan angka antara 0 dan 127. Ketika string dibandingkan, string tersebut dibandingkan berdasarkan kode ini. Misalnya, angka yang mewakili koma adalah 44. Periode sama dengan 46. Oleh karena itu, jika periode dan koma dibandingkan, periode dievaluasi lebih besar.

Berikut ini adalah beberapa perbandingan valid yang dapat diuji PHP untuk menentukan kebenarannya:

- $a == $b
- $age != 21
- $ageJosh < $ageJoni
- $house_price >= 1000000

Operator perbandingan yang menanyakan apakah dua nilai sama terdiri dari dua tanda sama dengan (==). Salah satu kesalahan paling umum adalah menggunakan tanda sama dengan tunggal untuk perbandingan. Satu tanda sama dengan menempatkan nilai ke dalam variabel. Jadi, pernyataan seperti jika ($weather = "hujan") akan menetapkan $weather menjadi hujan daripada memeriksa apakah sudah menyamai hujan, dan akan selalu benar. Jika kita menulis negatif (dengan menggunakan !), kondisi negatifnya benar. Perhatikan perbandingan berikut ini:

$age != 21

Kondisi yang diuji adalah $age tidak sama dengan 21. Oleh karena itu, jika $age sama dengan 20, perbandingannya benar.

Mengembalikan Array

Kita baru saja melihat fungsi yang mengembalikan satu nilai. Ada juga cara untuk mendapatkan beberapa nilai dari suatu fungsi. Metode pertama adalah mengembalikannya ke dalam array. Array seperti sekelompok variabel yang disatukan dalam satu baris. Contoh ini menunjukkan bagaimana kita dapat menggunakan array untuk mengembalikan nilai fungsi.

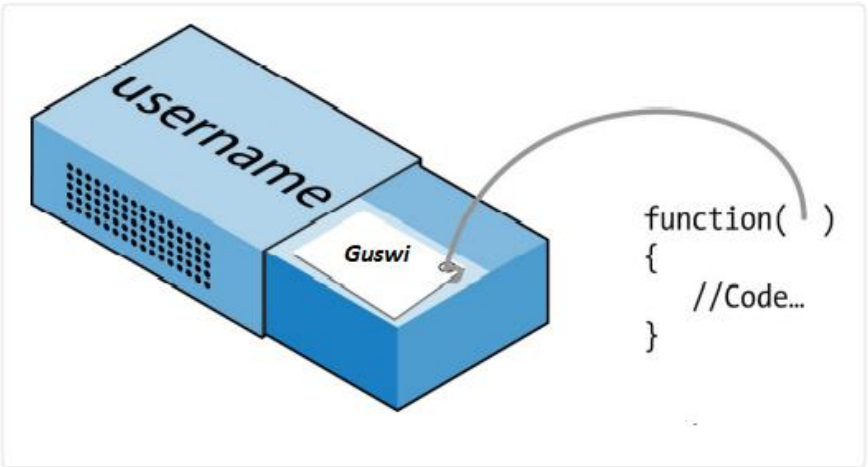
Contoh mengembalikan beberapa nilai dalam array

```
<?php
    $names = fix_names("JOSH", "henry", "fiKA");
    echo $names[0] . " " . $names[1] . " " . $names[2];
    function fix_names($n1, $n2, $n3)
    {
        $n1 = ucfirst(strtolower($n1));
        $n2 = ucfirst(strtolower($n2));
        $n3 = ucfirst(strtolower($n3));
        return array($n1, $n2, $n3);
    }
?>
```

Metode ini memiliki manfaat untuk memisahkan ketiga nama tersebut, daripada menggabungkannya menjadi satu string, sehingga kita dapat merujuk ke pengguna mana pun hanya dengan nama depan atau belakang, tanpa harus mengekstrak salah satu nama dari string yang dikembalikan.

Melewati Referensi

Di PHP, mengawali variabel dengan simbol & memberi tahu parser untuk meneruskan referensi ke nilai variabel, bukan nilai itu sendiri. Konsep ini mungkin sulit dipahami, jadi mari kembali ke metafora kotak korek api. Bayangkan bahwa, alih-alih mengambil selembar kertas dari kotak korek api, membacanya, menyalinnya ke selembar kertas lain, meletakkan bagian belakang asli, dan meneruskan salinan ke suatu fungsi, kita cukup menempelkan seutas benang ke selembar kertas asli dan meneruskan salah satu ujungnya ke fungsi.



Gambar 3.2 membayangkan referensi sebagai utas yang dilampirkan ke variabel

Sekarang fungsinya bisa mengikuti thread untuk mencari data yang akan diakses. Ini menghindari semua overhead untuk membuat salinan variabel hanya untuk penggunaan fungsi. Terlebih lagi, fungsi sekarang dapat mengubah nilai variabel. Ini berarti kita dapat menulis ulang contoh mengembalikan beberapa nilai dalam array untuk meneruskan referensi ke semua parameter, dan kemudian fungsi dapat memodifikasinya secara langsung.

Contoh Mengembalikan nilai dari suatu fungsi dengan referensi

```
<?php
    $a1 = "JOSEPH";
    $a2 = "teguh";
    $a3 = "santoSO";
```

```
    echo $a1 . " " . $a2 . " " . $a3 . "<br>";
    fix_names($a1, $a2, $a3);
    echo $a1 . " " . $a2 . " " . $a3;
    function fix_names(&$n1, &$n2, &$n3)
    {
        $n1 = ucfirst(strtolower($n1));
        $n2 = ucfirst(strtolower($n2));
        $n3 = ucfirst(strtolower($n3));
    }
?>
```

Daripada meneruskan string langsung ke fungsi, kita terlebih dahulu menetapkannya ke variabel dan mencetaknya untuk melihat nilai "sebelumnya". Kemudian kita memanggil fungsi seperti sebelumnya, tetapi letakkan simbol & di depan setiap parameter, yang memberi tahu PHP untuk meneruskan referensi variabel saja. Sekarang variabel $n1, $n2, dan $n3 dilampirkan ke "utas" yang mengarah ke nilai $a1, $a2, dan $a3. Dengan kata lain, ada satu grup nilai, tetapi dua set nama variabel diizinkan untuk mengaksesnya. Oleh karena itu, fungsi fix_names hanya perlu menetapkan nilai baru ke $n1, $n2, dan $n3 untuk memperbarui nilai $a1, $a2, dan $a3. Output dari kode ini adalah:

```
JOSEPH teguh santoSO
Joseph Tegus Santoso
```

Seperti yang kita lihat, kedua pernyataan echo hanya menggunakan nilai $a1, $a2, dan $a3.
Catatan: Hati-hati saat memberikan nilai dengan referensi. Jika kita perlu mempertahankan nilai aslinya, buat salinan variabel kita dan kemudian berikan salinannya dengan referensi.

Mengembalikan Variabel Global

Kita juga dapat memberikan akses fungsi ke variabel yang dibuat secara eksternal dengan mendeklarasikannya sebagai variabel global dari dalam fungsi. Kata kunci global diikuti dengan nama variabel memberi setiap bagian kode kita akses penuh ke sana.

Contoh Mengembalikan nilai dalam variabel global

```
<?php
    $a1 = "JOSEPH";
    $a2 = "teguh";
    $a3 = "santoSO";
    echo $a1 . " " . $a2 . " " . $a3 . "<br>";
    fix_names(); echo $a1 . " " . $a2 . " " . $a3;
    function fix_names()
    {
        global $a1; $a1 = ucfirst(strtolower($a1));
        global $a2; $a2 = ucfirst(strtolower($a2));
        global $a3; $a3 = ucfirst(strtolower($a3));
    }
?>
```

Sekarang kita tidak perlu meneruskan parameter ke fungsi, dan tidak harus menerimanya. Setelah dideklarasikan, variabel-variabel ini tetap bersifat global dan tersedia untuk program kita lainnya, termasuk fungsinya.

Catatan: Untuk mempertahankan cakupan lokal sebanyak mungkin, kita harus mencoba mengembalikan array atau menggunakan variabel dengan asosiasi. Jika tidak, kita akan mulai kehilangan beberapa manfaat fungsi.

Rekap Lingkup Variabel

Pengingat singkat tentang apa yang kita ketahui:

- *Variabel lokal* dapat diakses hanya dari bagian kode tempat kita mendefinisikannya. Jika mereka berada di luar suatu fungsi, mereka dapat diakses oleh semua kode di luar fungsi, kelas, dan sebagainya. Jika sebuah variabel berada di dalam suatu fungsi, hanya fungsi tersebut yang dapat mengakses variabel tersebut, dan nilainya akan hilang saat fungsi tersebut kembali.
- *Variabel global* dapat diakses dari semua bagian kode Anda.
- *Variabel statis* hanya dapat diakses dalam fungsi yang mendeklarasikannya tetapi mempertahankan nilainya melalui beberapa panggilan.

Benar jika variabel diset, bahkan jika tidak ada yang disimpan di dalamnya.

```
empty($varname)                    # Benar jika nilainya 0 atau string tanpa
                                     karakter di dalamnya atau tidak diset.
```

Kita juga dapat menguji jenis data apa yang ada dalam variabel. Misalnya, untuk melihat apakah nilainya bilangan bulat, kita dapat menggunakan yang berikut ini:

```
is_int($number)
```

Perbandingan benar jika nilai dalam $number adalah bilangan bulat. Beberapa tes lain yang disediakan oleh PHP adalah sebagai berikut:

- is_array($var2): Memeriksa apakah $var2 adalah array.
- is_float($number): Memeriksa apakah $number adalah bilangan floating point.
- is_null($var1): Memeriksa apakah $var1 sama dengan 0.
- is_numeric($string): Memeriksa apakah $string adalah string numerik.
- is_string($string): Memeriksa apakah $string adalah string.

Kita juga dapat menguji kondisi negatif dengan menggunakan tanda seru (!) di depan ekspresi. Ini benar-benar kondisi NOT yang logis, seperti dalam "Jika kondisi ini TIDAK benar, lakukan sesuatu." Misalnya, pernyataan berikut mengembalikan nilai true jika variabel tidak ada sama sekali:

```
!isset($varname)
```

Kita dapat menganggapnya dalam bahasa sederhana sebagai “Jika $varname tidak diset . .. .”

3.5 PENCOCOKAN POLA DENGAN EKSPRESI REGULER

Terkadang kita perlu membandingkan string karakter untuk melihat apakah mereka cocok dengan karakteristik tertentu, daripada melihat apakah mereka cocok dengan nilai yang tepat. Misalnya, kita mungkin ingin mengidentifikasi string yang dimulai dengan S atau string yang memiliki angka di dalamnya. Untuk jenis perbandingan ini, kita membandingkan string dengan sebuah pola. Pola-pola ini disebut ekspresi reguler.

Kita mungkin pernah menggunakan beberapa bentuk pencocokan pola di masa lalu. Saat kita menggunakan tanda bintang (*) sebagai wild card saat mencari file (dir ex*.doc, misalnya), kita sedang mencocokkan pola. Misalnya, ex*.txt adalah sebuah pola.

Setiap string yang dimulai dengan ex dan diakhiri dengan .txt, dengan karakter apa pun di antara ex dan .txt, cocok dengan polanya. String exam.txt, ex33.txt, dan ex3x4.txt semuanya

cocok dengan polanya. Menggunakan ekspresi reguler hanyalah variasi yang lebih kuat dari menggunakan kartu liar. Salah satu penggunaan umum untuk pencocokan pola adalah untuk memeriksa input dari formulir halaman web. Jika input informasi tidak cocok dengan pola tertentu, mungkin itu bukan sesuatu yang ingin kita simpan di database Anda. Misalnya, jika pengguna mengetikkan kode ZIP ke formulir Anda, kita tahu formatnya harus lima angka atau ZIP + 4. Jadi, kita dapat memeriksa input untuk melihat apakah itu cocok dengan polanya. Jika tidak, kita tahu itu bukan kode ZIP yang valid, dan kita dapat meminta pengguna untuk mengetikkan informasi yang benar.

Ekspresi reguler digunakan untuk pencocokan pola dalam banyak situasi. Banyak perintah dan program Linux, seperti grep, vi, atau sed, menggunakan ekspresi reguler. Banyak aplikasi, seperti editor teks dan pengolah kata, memungkinkan pencarian menggunakan ekspresi reguler. PHP menyediakan dukungan untuk ekspresi reguler yang kompatibel dengan Perl. Bagian berikut menjelaskan beberapa ekspresi reguler dasar yang kompatibel dengan Perl, tetapi pencocokan pola yang jauh lebih kompleks dan kuat dimungkinkan. Lihat www.php.net/manual/en/reference.pcre.pattern.syntax.php untuk penjelasan lebih lanjut tentang ekspresi reguler yang kompatibel dengan Perl.

Menggunakan karakter khusus dalam pola

Pola terdiri dari karakter literal dan karakter khusus.

- Karakter literal adalah karakter normal, tanpa arti khusus. *E* adalah *e,* misalnya, tanpa arti selain itu adalah salah satu dari 26 huruf dalam alfabet.
- Karakter khusus, di sisi lain, memiliki arti khusus dalam pola, seperti tanda bintang (*) saat digunakan sebagai kartu liar. Tabel dibawah ini menunjukkan karakter khusus yang dapat kita gunakan dalam pola.

Tabel 3.2 Karakter khusus yang dapat digunakan dalam pola

Karakter	Arti	Contoh	Kecocokan	Ketidak cocokan
^	Awalan baris	^k	kucing	kucingku
$	Akhir baris	c$	tic	Stik
.	Ada karakter tunggal	. .	Strig yang memuat setidaknya dua karakter	a, I
?	Karakter sebelumnya opsional	mea?n	mean, men	moan
()	Group karakter literal bersama	m(ea)n	mean	men, mn
[]	Mencakup satu set dari opsi karakter literal	m[ea]n	men, man	mean, mn
-	Merepresentasikan semua karakter diantara dua karakter	m[a-c]n	man, mbn, mcn	mdn, mun, maan
+	Satu atau lebih pada item yang mendahului	pintu[1-3]+	pintu111, pintu131	Pintu, pintu55

*	Nol atau lebih pada item yang mendahului	pintu[1-3]*	pintu, pintu311	Pintu4, pintu445
{ , }	Mengawali atau mengakhiri tingkat pada repetisi	a{2,5}	aa,aaaaa	A, xx3
\	Mengikuri karakter adalah literal	m*n	m*n	men, mean
(|)	Set string alternatif	(Gus|Guswi)	Gus, Guswi	Gisele, Gi

Mempertimbangkan beberapa contoh pola

Karakter literal dan khusus digabungkan untuk membuat pola, terkadang pola yang panjang dan rumit. Sebuah string dibandingkan dengan pola, dan jika cocok, perbandingannya benar. Beberapa contoh pola mengikuti, dengan perincian pola dan beberapa contoh string yang cocok dan tidak cocok.

Contoh 1

^[A-Za-z].*

Pola ini mendefinisikan string yang dimulai dengan huruf dan memiliki dua bagian ini:

- ^[A-Za-z]: Bagian pertama dari pola menentukan bahwa awal string harus berupa huruf (baik huruf besar atau kecil).
- .*: Bagian kedua dari pola memberi tahu PHP bahwa string karakter dapat terdiri dari satu atau lebih karakter, termasuk angka, spasi, atau apa pun.

Ekspresi ^[A-Za-z].* cocok dengan string berikut: mainkan lagi, Sam, 4 kali, dan I.

Ekspresi ^[A-Za-z].* tidak cocok dengan string berikut: 123 dan ?

Contoh 2

Dear (Kimi|Riki)

Pola ini mendefinisikan dua string alternatif dan memiliki dua bagian ini:

- Dear: Bagian pertama dari pola hanyalah karakter literal diikuti dengan spasi.
- (Kimi|Riki): Bagian kedua mendefinisikan Kimi atau Riki sebagai string yang cocok.

Ekspresi Dear (Kimi|Riki) cocok dengan string berikut: Dear Kimi dan Dear Riki.

Ekspresi Dear (Kimi|Riki) tidak cocok dengan string berikut: Dear Riki dan Kimi.

Contoh 3

^[0-9]{5}(\-[0-9]{4})?$

Pola ini mendefinisikan kode ZIP apa pun dan memiliki beberapa bagian:

- ^[0-9]{5}: Bagian pertama dari pola menggambarkan setiap string lima
- angka.
- \-: Garis miring terbalik menunjukkan bahwa tanda hubung adalah literal.
- [0-9]{4}: Bagian pola ini memberi tahu PHP bahwa karakter berikutnya harus berupa rangkaian angka yang terdiri dari empat karakter.
- ( )?: Karakter ini mengelompokkan dua bagian terakhir dari pola dan menjadikannya opsional.
- $: Tanda dolar menyatakan bahwa string ini harus diakhiri (tidak ada karakter yang diizinkan setelah pola).

Ekspresi ^[0-9]{5}(\-[0-9]{4})?$ cocok dengan string berikut:
90001 dan 90002-4323.

Ekspresi ^[0-9]{5}(\-[0-9]{4})?$ tidak cocok dengan string berikut:
9001 dan 12-4321.

Contoh 4

^.+@.+\.com$

Pola ini mendefinisikan string apa pun dengan @ yang disematkan yang berakhiran .com. Dengan kata lain, ini mendefinisikan format umum (tetapi bukan satu-satunya) untuk alamat email. Ungkapan ini memiliki beberapa bagian:

- ^.+: Bagian pertama dari pola menjelaskan setiap string dari satu atau lebih karakter yang mendahului @.
- @: Ini adalah literal @ (tanda "at"). @ bukan karakter khusus dan tidak perlu didahului dengan \.
- .+: Ini adalah string dari satu atau lebih karakter.
- \.: Garis miring menunjukkan bahwa PHP harus mencari titik literal.
- com$: Ini mendefinisikan string literal com di akhir string, dan $ menandai akhir string.

Ekspresi ^.+@.+\.com$ cocok dengan string berikut: you@yourcompany.com dan guswi@somedomain.com.
Ekspresi ^.+@.+\.com$ tidak cocok dengan string berikut: you@yourcompany.net, you@.com, dan @you.com.

Menggunakan fungsi PHP untuk pencocokan pola

Kita dapat membandingkan apakah suatu pola cocok dengan string dengan fungsi preg_match. Format umumnya adalah sebagai berikut:

preg_match("*pattern",value*);

Pola harus diapit oleh sepasang pembatas — karakter yang melingkupi pola. Seringkali, garis miring (/) digunakan sebagai pembatas. Namun, kita dapat menggunakan karakter nonalfanumerik apa pun, kecuali garis miring terbalik (\). Misalnya, untuk memeriksa nama yang diketik pengguna dalam formulir, cocokkan pola dengan nama (disimpan dalam variabel $name), sebagai berikut:

preg_match("/^[A-Za-z' -]+$/",$name)

Pola dalam pernyataan ini melakukan hal berikut:

- Melampirkan pola dalam garis miring (/).
- Menggunakan ^ dan $ untuk menandakan awal dan akhir string, masing-masing. Itu berarti bahwa semua karakter dalam string harus sesuai dengan polanya.
- Melampirkan semua karakter literal yang diperbolehkan dalam string di [ ]. Tidak ada karakter lain yang diperbolehkan. Karakter yang diperbolehkan adalah huruf besar dan kecil, tanda kutip ('), spasi, dan tanda hubung (-). Kita dapat menentukan rentang karakter dengan menggunakan tanda hubung di dalam [ ]. Saat kita melakukannya, seperti dalam A-Z, tanda hubung tidak mewakili karakter literal. Karena kita juga ingin tanda hubung disertakan sebagai karakter karakter literal yang diizinkan dalam string Anda, kita perlu menambahkan tanda hubung yang tidak berada di antara dua karakter lainnya. Dalam hal ini, tanda hubung disertakan di akhir daftar karakter literal.
- Mengikuti daftar karakter literal di [ ] dengan +. Tanda plus berarti bahwa string dapat berisi sejumlah karakter di dalam [ ], tetapi harus berisi setidaknya satu karakter.

Jika pola itu sendiri berisi garis miring ke depan, pembatas tidak boleh berupa garis miring ke depan. Kita harus menggunakan karakter lain untuk pembatas, seperti:

preg_match("#^[A-Za-z' -/]+$#",$name)

Bergabung dengan beberapa perbandingan

Seringkali kita perlu mengajukan lebih dari satu pertanyaan untuk menentukan kondisi Anda. Misalnya, perusahaan kita menawarkan katalog untuk produk yang berbeda dalam bahasa yang berbeda. Kita perlu mengetahui jenis katalog produk yang ingin dilihat pelanggan dan bahasa apa yang mereka butuhkan untuk melihatnya. Ini mengharuskan kita untuk bergabung dengan perbandingan, yang memiliki format umum berikut:

comparison1 and|or|xor comparison2 and|or|xor comparison3 and|or|xor ...

Operator logika

Operator logika menghasilkan hasil benar atau salah, dan karena itu juga dikenal sebagai operator Boolean. Ada empat dari mereka.

Tabel 3.3 *Operator logika*

Operator Logika	Deskripsi
AND	TRUE jika salah satu operan adalah TRUE
OR	TRUE jika salah satu dari dua operan adalah TRUE
XOR	TRUE jika operan FALSE, atau FALSE jika operan TRUE
NOT	TRUE jika kedua operan adalah TRUE

Kita dapat melihat operator ini digunakan dalam Contoh dibawah ini. Perhatikan bahwa simbol ! diperlukan oleh PHP sebagai pengganti kata NOT. Selanjutnya, operator bisa huruf kecil atau besar.

Contoh operator logika yang digunakan

```
<?php

    $a = 1; $b = 0;

    echo ($a AND $b) . "<br>";
    echo ($a or $b) . "<br>";
    echo ($a XOR $b) . "<br>";
    echo !$a . "<br>";
?>
```

Contoh ini menghasilkan NULL, 1, 1, NULL, yang berarti bahwa hanya pernyataan echo kedua dan ketiga yang dievaluasi sebagai TRUE. (Ingat bahwa NULL—atau tidak sama sekali—mewakili nilai FALSE.) Hal ini karena pernyataan AND membutuhkan kedua operan menjadi TRUE jika akan mengembalikan nilai TRUE, sedangkan pernyataan keempat melakukan NOT pada nilai $a, mengubahnya dari TRUE (nilai 1) menjadi FALSE. Jika kita ingin bereksperimen dengan ini, cobalah kodenya, berikan nilai $a dan $b yang bervariasi dari 1 dan 0.Perbandingan dihubungkan oleh salah satu dari tiga kata berikut:

- and: Kedua perbandingan itu benar.
- or: Satu atau kedua perbandingan benar.
- xor: Hanya satu dari perbandingan yang benar.

Kita dapat merangkai sebanyak mungkin perbandingan yang diperlukan. Perbandingan menggunakan dan diuji terlebih dahulu, perbandingan menggunakan xor diuji selanjutnya, dan perbandingan menggunakan atau diuji terakhir. Misalnya, kondisi berikut mencakup tiga perbandingan:

$resCity == “Semarang” or $resState == “ID” and $name == “Guswi”

Jika nama pelanggan Guswi dan dia tinggal di Indonesia(ID), pernyataan ini benar. Pernyataan itu juga benar jika dia tinggal di Semarang, apa pun namanya. Kondisi ini tidak benar jika dia tinggal di ID tapi namanya bukan Guswi. Kita mendapatkan hasil ini karena skrip memeriksa kondisi dalam urutan berikut:

1. and dibandingkan.
   Script memeriksa $resState untuk melihat apakah itu sama dengan ID dan memeriksa $name untuk melihat apakah itu sama dengan Guswi. Jika keduanya cocok, kondisinya benar, dan skrip tidak perlu memeriksa atau. Jika hanya satu atau tidak satu pun dari variabel yang sama dengan nilai yang ditentukan, pengujian dilanjutkan.
2. or dibandingkan. Skrip memeriksa $resCity untuk melihat apakah sama dengan Semarang. Jika ya, maka kondisinya benar. Jika tidak, kondisinya salah.
   Kita dapat mengubah urutan perbandingan yang dibuat dengan menggunakan tanda kurung. Kata penghubung di dalam tanda kurung dievaluasi terlebih dahulu. Misalnya, kita dapat menulis ulang pernyataan sebelumnya dengan tanda kurung, sebagai berikut:

($resCity == “Semarang or $resState == “ID”) and $name == “Guswi”

Tanda kurung mengubah urutan kondisi diperiksa. Sekarang or dicentang terlebih dahulu karena berada di dalam tanda kurung. Pernyataan kondisi ini benar jika nama pelanggan Guswi dan dia tinggal di Semarang atau ID. Kita mendapatkan hasil ini karena skrip memeriksa kondisi sebagai berikut:

1. or dibandingkan.
   Script memeriksa untuk melihat apakah $resCity sama dengan Semarang atau $resStateequals ID. Jika tidak, seluruh kondisi salah, dan pengujian berhenti. Jika ya, bagian dari kondisi ini benar. Namun, perbandingan di sisi lain dan juga harus benar, sehingga pengujian berlanjut.
2. and dibandingkan. Script memeriksa $name untuk melihat apakah sama dengan Guswi. Jika ya, maka kondisinya benar. Jika tidak, kondisinya salah.

Gunakan tanda kurung secara bebas, bahkan ketika kita yakin kita tahu urutan perbandingannya. Tanda kurung yang tidak perlu tidak ada salahnya, tetapi perbandingan yang memiliki hasil yang tidak terduga bisa. Jika kita terbiasa dengan bahasa lain, seperti C, kita mungkin telah menggunakan ||(untuk or) dan && (untuk and) sebagai pengganti kata. || dan && bekerja di PHP juga. Pernyataan $a < $b && $c > $b sama validnya dengan pernyataan $a < $b dan $c > $b. || dicentang sebelum atau, dan && dicentang.

Catatan: Saat coding, ingatlah bahwa AND dan OR memiliki prioritas yang lebih rendah daripada versi operator lainnya, && dan ||. Dalam ekspresi kompleks, mungkin lebih aman menggunakan && dan || untuk alasan ini.

Operator OR dapat menyebabkan masalah yang tidak disengaja dalam pernyataan if, karena operan kedua tidak akan dievaluasi jika yang pertama dievaluasi sebagai TRUE.

Contoh pernyataan menggunakan operator OR

```
<?php
   if ($finished == 1 OR getnext() == 1) exit;
?>
```

Jika kita perlu getnext dipanggil pada setiap pernyataan if, kita dapat menulis ulang kode seperti yang telah dilakukan pada Contoh dibawah ini.

Contoh Pernyataan "if ... OR" dimodifikasi untuk memastikan pemanggilan getnext

```
<?php
    $gn = getnext();
    if ($finished == 1 OR $gn == 1) exit;
?>
```

Dalam hal ini, kode dalam fungsi getnext akan dieksekusi dan nilai yang dikembalikan akan disimpan di $gn sebelum pernyataan if.

Catatan: Solusi lain adalah dengan mengganti kedua klausa untuk memastikan getnext dieksekusi, karena akan muncul pertama kali dalam ekspresi.

Tabel di bawah ini menunjukkan semua kemungkinan variasi penggunaan operator logika. Kita juga harus diperhatikan bahwa !TRUE sama dengan FALSE dan !FALSE sama dengan TRUE.

Tabel 3.4 Semua kemungkinan ekspresi logika PHP

Input		**Operator dan Hasil**		
a	**b**	**AND**	**OR**	**XOR**
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	TRUE
FALSE	TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	FALSE

3.6 BERSYARAT

Kondisi mengubah alur program. Mereka memungkinkan kita untuk mengajukan pertanyaan tentang hal-hal tertentu dan menanggapi jawaban yang kita dapatkan dengan cara yang berbeda. Conditional adalah inti dari halaman web dinamis—tujuan penggunaan PHP sejak awal—karena conditional memudahkan untuk membuat output yang berbeda setiap kali halaman dilihat. Ada tiga jenis kondisional non-perulangan: pernyataan if, pernyataan switch, dan operator?. Dengan non-perulangan, maksud saya bahwa tindakan yang diprakarsai oleh pernyataan berlangsung dan aliran program kemudian bergerak, sedangkan persyaratan perulangan (yang akan segera kita bahas) mengeksekusi kode berulang-ulang sampai suatu kondisi terpenuhi.

Menggunakan Pernyataan Bersyarat

Pernyataan bersyarat mengeksekusi blok pernyataan hanya jika kondisi tertentu benar. Berikut adalah dua jenis pernyataan bersyarat yang berguna:

- Pernyataan if: Mengatur kondisi dan mengujinya. Jika kondisinya benar, blok pernyataan akan dieksekusi.
- Pernyataan switch: Mengatur daftar kondisi alternatif. Ini menguji kondisi sebenarnya dan mengeksekusi blok pernyataan yang sesuai.

Kita memberi tahu kita cara menggunakan kedua pernyataan bersyarat itu dalam teks yang mengikuti

Menggunakan pernyataan if

Pernyataan if menguji kondisi, mengeksekusi blok pernyataan saat kondisi benar. Bagian berikut membahas cara membangun pernyataan if menggunakan format yang sesuai,

membuat pernyataan if untuk kondisi salah, dan menyusun satu pernyataan if di dalam yang lain.

Membangun pernyataan if

Format umum dari pernyataan bersyarat if adalah sebagai berikut:

```
if ( condition )
{
        block of statements
}
elseif ( condition )
{
        block of statements
}
else
{
        block of statements
}
```

Pernyataan if terdiri dari tiga bagian:

- if: Bagian ini diperlukan. Hanya satu jika diizinkan. Ini menguji suatu kondisi:
    - *Jika kondisinya benar*: Blok pernyataan dieksekusi. Setelah pernyataan dieksekusi, skrip pindah ke instruksi berikutnya mengikuti pernyataan bersyarat; jika pernyataan bersyarat berisi bagian elseif atau else, skrip akan melewatinya.
    - *Jika kondisinya tidak benar*: Blok pernyataan tidak dieksekusi. Script melompat ke instruksi berikutnya, yang dapat berupa elseif, an else, atau instruksi berikutnya setelah pernyataan kondisi if.
- elseif: Bagian ini opsional. Kita dapat menggunakan lebih dari satu jika kita mau. Elseif juga menguji suatu kondisi:
    - *Jika kondisinya benar:* Blok pernyataan dieksekusi. Setelah exe memotong blok pernyataan, skrip pergi ke instruksi berikutnya mengikuti pernyataan bersyarat; jika pernyataan if berisi bagian elseif tambahan atau bagian lain, skrip akan melewatinya.
    - *Jika kondisinya tidak benar*: Blok pernyataan tidak dieksekusi. Script melompat ke instruksi berikutnya, yang dapat berupa elseif, an else, atau instruksi berikutnya setelah pernyataan kondisi if.
- else: Bagian ini juga opsional. Hanya satu lagi yang diperbolehkan. Bagian ini tidak menguji suatu kondisi, melainkan mengeksekusi blok pernyataan. Script memasuki bagian lain hanya ketika bagian if dan semua bagian elseif tidak benar

Berikut adalah contoh cara membuat pernyataan if. Berpura-puralah kita seorang guru. Pernyataan if berikut, ketika diberi nilai ujian, mengirimkan nilai dan pesan teks singkat kepada siswa Anda. Ia menggunakan ketiga bagian dari pernyataan if (if, elseif, dan else), sebagai berikut:

```
if ($score > 92 )
{
$grade = “A”;
$message = “Excellent!”;
}
```

```
 elseif
($score <= 92 and $score > 83 )
{
$grade = “B”;
$message = “Good!”;
}
elseif ($score <= 83 and $score > 74 )
{
$grade = “C”;
$message = “Okay”;
}
elseif
($score <= 74 and $score > 62 )
{
$grade = “D”;
$message = “Uh oh!”;
}
else
{
$grade = “F”;
$message = “Doom is upon you!”;
}
echo $message.”\n”;
echo “Your grade is $grade\n”;
```

Pernyataan bersyarat if berlangsung sebagai berikut:

1. Nilai dalam $score dibandingkan dengan 92. Jika $score lebih besar dari 92, $grade diatur ke A, $message diatur ke Excellent!, dan skrip melompat ke pernyataan echo. Jika $score adalah 92 atau kurang, $grade dan $message tidak disetel, dan skrip melompat ke bagian elseif.
2. Nilai dalam $score dibandingkan dengan 92 dan 83.Jika $score adalah 92 atau kurang dan lebih besar dari 83, $grade dan $message disetel, dan skrip melompat ke pernyataan echo. Jika $score 83 atau kurang, $grade dan $message tidak disetel, dan skrip melompat ke bagian elseif kedua.
3. Nilai dalam $score dibandingkan dengan 83 dan 74.Jika $score adalah 83 atau kurang dan lebih besar dari 74, $grade dan $messageare set, dan skrip melompat ke pernyataan echo. Jika $score adalah 74 atau kurang, $grade dan $message tidak disetel, dan skrip melompat ke bagian elseif berikutnya.
4. Nilai dalam $score dibandingkan dengan 74 dan 62.Jika $score adalah 74 atau kurang dan lebih besar dari 62, $grade dan $message disetel, dan skrip melompat ke pernyataan echo. Jika $score 62 atau kurang, $grade dan $message tidak disetel, dan skrip melompat ke bagian lain.
5. $grade diatur ke F, dan $message diatur ke Doom ada di tangan Anda!. Script melanjutkan ke pernyataan echo.

Ketika blok yang akan dieksekusi oleh setiap bagian dari pernyataan kondisional if hanya berisi satu pernyataan, kurung kurawal tidak diperlukan. Misalnya, katakanlah contoh sebelumnya hanya memiliki satu pernyataan di blok, sebagai berikut:

```
if ($grade > 92 )
```

```
{
        $grade = “A”;
}
```

Kita bisa menuliskannya sebagai berikut:

```
if ($grade > 92 )
  $grade = “A”;
```

Pintasan ini dapat menghemat pengetikan. Namun, ketika kita menggunakan beberapa pernyataan if, kita harus menyertakan kurung kurawal karena mengabaikannya dapat menyebabkan kebingungan.

Negating pernyataan if

Kita dapat menulis pernyataan if agar blok pernyataan dijalankan ketika kondisi salah dengan meletakkan tanda seru (!) di awal kondisi. Misalnya, kita dapat menggunakan pernyataan if berikut:

```
if (preg_match(“/^S[a-z]*/”,$string))
{
        $list[]=$string.”\n”;
}
```

Pernyataan if ini membuat larik string yang dimulai dengan S. Lebih khusus lagi, jika $string cocok dengan pola yang menetapkan satu huruf besar S di awal, diikuti oleh sejumlah huruf kecil, kondisinya benar dan blok pernyataan dijalankan. Namun, jika kita menempatkan tanda seru di awal kondisi, banyak hal akan berubah. Misalnya, kita menggunakan pernyataan berikut sebagai gantinya:

```
if (!preg_match(“/^S[a-z]*/”,$string)
{
        $list[]=$string.”\n”;
}
```

Dalam hal ini, array $list berisi semua string kecuali yang dimulai dengan S. Karena ! muncul di awal kondisi, kondisinya adalah “$string tidak cocok dengan pola yang dimulai dengan S.” Jadi, ketika $string tidak dimulai dengan S, kondisinya benar

Nesting Pernyataan if

Kita dapat memiliki pernyataan bersyarat if di dalam pernyataan bersyarat if yang lain. Menempatkan satu pernyataan di dalam yang lain disebut *nesting*. Misalnya, kita perlu menghubungi semua pelanggan kita yang tinggal di Idaho. Kita berencana untuk mengirim email kepada mereka yang memiliki alamat email dan mengirim surat kepada mereka yang tidak memiliki alamat email. Kita dapat mengidentifikasi grup pelanggan dengan menggunakan pernyataan if nested berikut:

```
if ( $custState == “ID” )
{
        if ( $EmailAdd = “” )
        {
        $contactMethod = “letter”;
        }
```

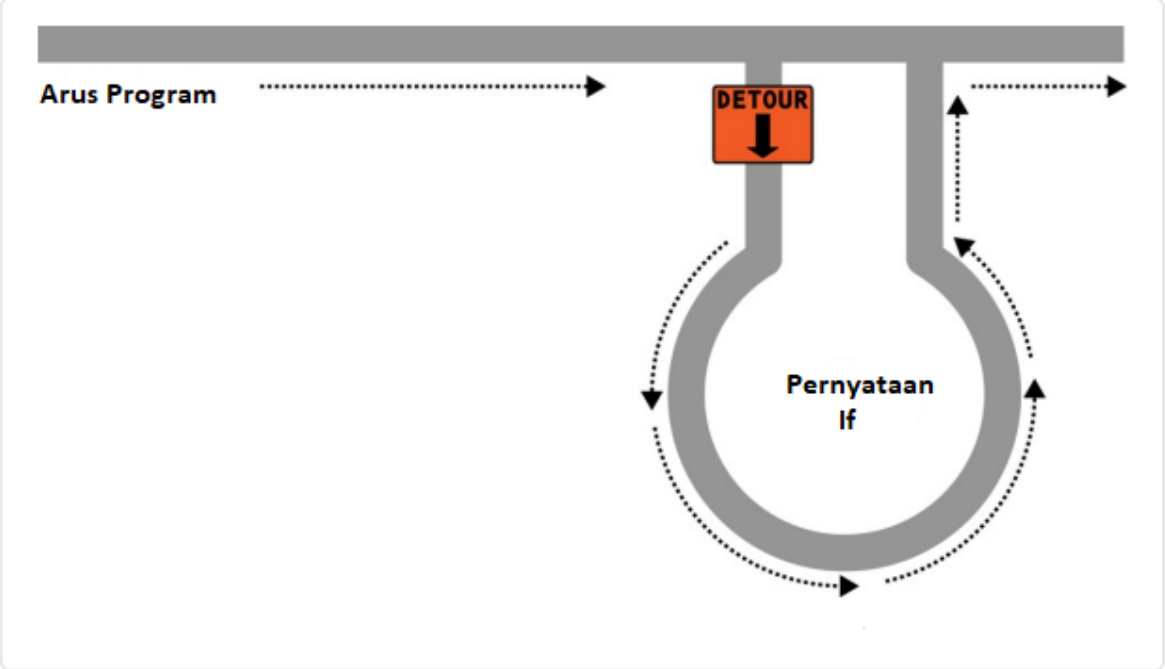
```
        else
        {
        $contactMethod = “email”;
        }
}
else
{
        $contactMethod = “none needed”;
}
```

Catatan: Pernyataan else menutup pernyataan if ... else atau if ... elseif ... else. Kita dapat meninggalkan final else jika tidak diperlukan, tetapi kita tidak dapat memilikinya sebelum elseif; kita juga tidak dapat memiliki elseif sebelum pernyataan if.

Pernyataan If

Salah satu cara berpikir tentang alur program ini adalah dengan membayangkannya sebagai jalan raya satu lajur yang kita lalui. Ini cukup banyak garis lurus, tetapi kadang-kadang kita menemukan berbagai tanda yang memberi tahu kita ke mana harus pergi.

Dalam kasus pernyataan if, kita dapat membayangkan menemukan tanda jalan memutar yang harus kita ikuti jika kondisi tertentu TRUE. Jika demikian, kita berkendara dan mengikuti jalan memutar sampai kita kembali ke tempat awalnya dan kemudian melanjutkan perjalanan kita ke arah semula. Atau, jika kondisinya tidak TRUE, kita mengabaikan jalan memutar dan terus mengemudi.



Gambar 3.3 Aliran program seperti jalan raya satu jalur

Isi kondisi if dapat berupa ekspresi PHP yang valid, termasuk kesetaraan, perbandingan, pengujian untuk 0 dan NULL, dan bahkan nilai yang dikembalikan oleh fungsi (baik fungsi bawaan atau yang kita tulis). Tindakan yang harus dilakukan ketika kondisi if TRUE umumnya ditempatkan di dalam kurung kurawal, {}. Namun, kita dapat mengabaikan kurung kurawal jika kita hanya memiliki satu pernyataan untuk dieksekusi. Tetapi jika kita selalu menggunakan kurung kurawal, kita akan terhindar dari pencarian bug yang sulit dilacak, seperti saat kita menambahkan baris tambahan ke suatu kondisi dan tidak dievaluasi karena kurangnya kurung kurawal. (Perhatikan bahwa untuk spasi dan kejelasan, banyak contoh

dalam buku ini mengabaikan saran ini dan menghilangkan tanda kurung kurawal untuk pernyataan tunggal.) Pada Contoh di bawah ini, bayangkan bahwa ini adalah akhir bulan dan semua tagihan kita telah dibayar, jadi kita sedang melakukan beberapa pemeliharaan rekening bank.

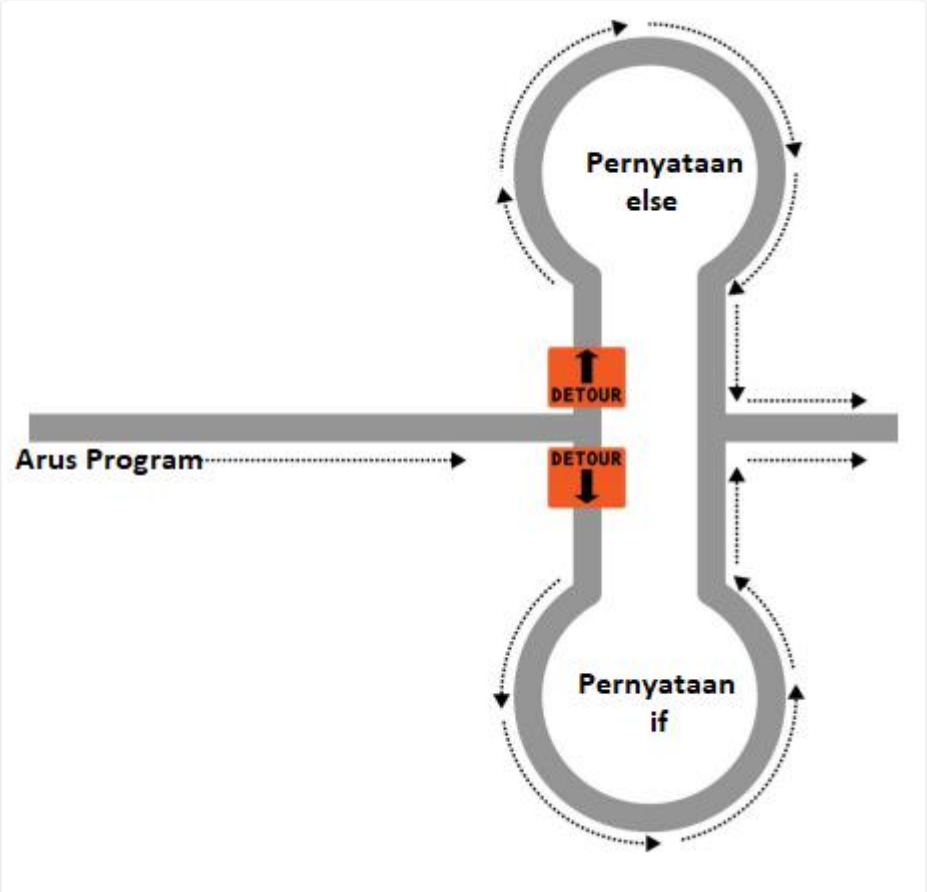
Contoh - pernyataan* if *dengan kurung kurawal

```
<?php
    if ($bank_balance < 100)
    {
        $money = 1000;
        $bank_balance += $money;
    }
?>
```

Dalam contoh ini, kita memeriksa saldo kita untuk melihat apakah kurang dari 100 (atau berapa pun mata uang Anda). Jika demikian, kita membayar sendiri 1.000 dan kemudian menambahkannya ke saldo. (Jika hanya menghasilkan uang sesederhana itu!) Jika saldo bank adalah 100 atau lebih, pernyataan kondisional diabaikan dan alur program melompat ke baris berikutnya (tidak ditampilkan). Dalam buku ini, kurung kurawal pembuka biasanya dimulai pada baris baru. Beberapa orang suka menempatkan kurung kurawal pertama di sebelah kanan ekspresi kondisional; yang lain memulai baris baru dengannya. Salah satu dari ini baik-baik saja, karena PHP memungkinkan kita untuk mengatur karakter spasi putih kita (spasi, baris baru, dan tab) dengan cara apa pun yang kita pilih. Namun, kita akan menemukan kode kita lebih mudah dibaca dan di-debug jika kita membuat indentasi setiap level kondisional dengan tab.

Pernyataan else

Kadang-kadang ketika suatu kondisi bukan TRUE, kita mungkin tidak ingin segera melanjutkan ke kode program utama tetapi mungkin ingin melakukan sesuatu yang lain sebagai gantinya. Di sinilah pernyataan else masuk. Dengan itu, kita dapat mengatur jalan memutar kedua di jalan raya Anda, perhatikan gambar di bawah ini.



Gambar 3.4 jalan raya sekarang memiliki jalan memutar jika dan jalan memutar lainnya

Dengan pernyataan if ... else, pernyataan kondisional pertama dijalankan jika kondisinya TRUE. Tetapi jika FALSE, yang kedua dieksekusi. Salah satu dari dua pilihan harus dijalankan. Dalam keadaan apa pun keduanya (atau tidak keduanya) dapat dieksekusi. Contoh dibawah ini menunjukkan penggunaan struktur if ... else.

Contoh Pernyataan if ... else dengan kurung kurawal

```
<?php
    if ($bank_balance < 100)
    {
    $money = 1000;
    $bank_balance += $money;
    }
    else
    {
        $savings += 50;
        $bank_balance −= 50;
}
?>
```

Dalam contoh ini, sekarang kita telah memastikan bahwa kita memiliki 100 atau lebih di bank, pernyataan else dijalankan, di mana kita menempatkan sebagian dari uang ini ke dalam rekening tabungan Anda. Seperti pernyataan if, jika pernyataan lain kita hanya memiliki satu pernyataan kondisional, kita dapat memilih untuk tidak menggunakan kurung kurawal. (Namun, kurung kurawal selalu disarankan. Pertama, kurung kurawal membuat kode lebih

mudah dipahami. Kedua, kurung kurawal memungkinkan kita menambahkan lebih banyak pernyataan ke cabang nanti.)

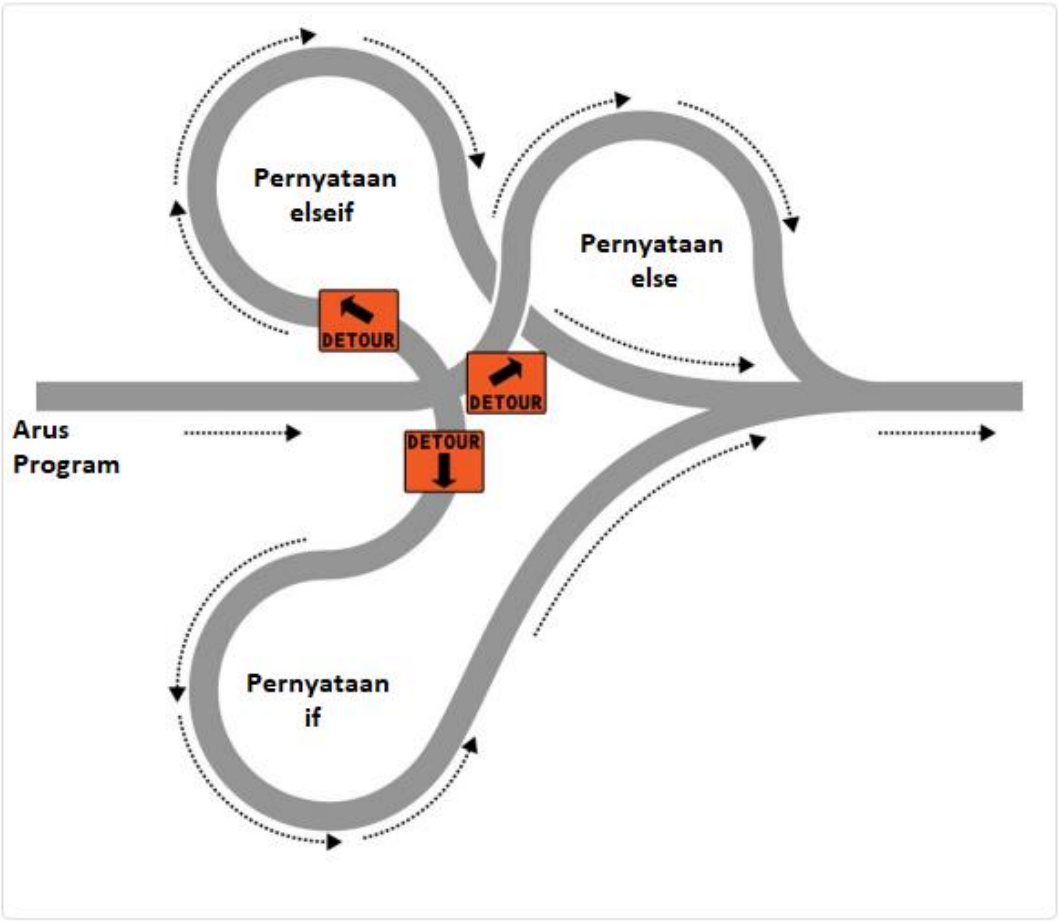
Pernyataan elseif

Ada juga saat-saat ketika kita ingin sejumlah kemungkinan yang berbeda terjadi, berdasarkan urutan kondisi. Kita dapat mencapai ini menggunakan pernyataan elseif. Seperti yang kita bayangkan, ini seperti pernyataan else, kecuali bahwa kita menempatkan ekspresi kondisional lebih lanjut sebelum kode kondisional. Pada contoh dibawah ini kita dapat melihat konstruksi if ... elseif ... else yang lengkap.

Contoh Pernyataan* if ... elseif ... else *dengan kurung kurawal.

```
<?php
    if ($bank_balance < 100)
    {
        $money = 1000;
        $bank_balance += $money;
    }
    elseif ($bank_balance > 200)
    {
    $savings += 100;
    $bank_balance −= 100;
    }
    else
    {
        $savings += 50;
        $bank_balance −= 50
}
?>
```

Dalam contoh, pernyataan elseif telah disisipkan di antara pernyataan if dan else. Ini memeriksa apakah saldo bank kita melebihi 200 dan, jika demikian, memutuskan bahwa kita mampu menghemat 100 bulan ini. Meskipun saya mulai meregangkan metafora agak terlalu jauh, kita dapat membayangkan ini sebagai rangkaian jalan memutar multi-arah.



Gambar 3.5 Jalan raya dengan if, elseif, dan else memutar

Kita dapat memiliki sebanyak mungkin pernyataan elseif yang kita inginkan. Tetapi karena jumlah pernyataan elseif meningkat, kita mungkin lebih disarankan untuk mempertimbangkan pernyataan switch jika sesuai dengan kebutuhan Anda. Kita akan melihat itu selanjutnya.

Pernyataan Switch

Pernyataan switch berguna dalam kasus di mana satu variabel atau hasil ekspresi dapat memiliki beberapa nilai, yang masing-masing akan memicu fungsi yang berbeda. Misalnya, pertimbangkan sistem menu berbasis PHP yang meneruskan string tunggal ke kode menu utama sesuai dengan permintaan pengguna. Katakanlah opsinya adalah Beranda, Tentang, Berita, Login, dan Tautan, dan kita menetapkan variabel $page ke salah satunya, sesuai dengan input pengguna.

***Contoh Pernyataan* multi-line if ... elseif ...**

```
<?php
    If          ($page == "Home")       echo "You selected Home";
    Elseif ($page == "About")   echo "You selected About";
    Elseif ($page == "News")        echo "You selected News";
    Elseif ($page == "Login")   echo "You selected Login";
    Elseif ($page == "Links")   echo "You selected Links";
?>
```

***Contoh Pernyataan* Switch**

```
<?php
    switch ($page)
    {
        case "Home":
            echo "You selected Home";
            break;
        case "About":
        echo "You selected About";
        break;
        case "News":
        echo "You selected News";
        break;
        case "Login":
        echo "You selected Login";
        break;
        case "Links":
            echo "You selected Links";
            break;
    }
?>
```

Seperti yang kita lihat, $page disebutkan hanya sekali di awal pernyataan switch. Setelah itu, perintah case akan memeriksa kecocokan. Ketika satu terjadi, pernyataan kondisional yang cocok dijalankan. Tentu saja, dalam program nyata kita akan memiliki kode di sini untuk ditampilkan atau melompat ke halaman, daripada hanya memberi tahu pengguna apa yang dipilih.

Menggunakan pernyataan switch

Untuk sebagian besar situasi, pernyataan kondisional if berfungsi paling baik. Namun, terkadang kita memiliki daftar kondisi dan ingin mengeksekusi pernyataan yang berbeda untuk setiap kondisi. Misalnya, skrip kita menghitung pajak penjualan. Bagaimana kita menangani tarif pajak penjualan negara yang berbeda? Pernyataan switch dirancang untuk situasi seperti itu. Pernyataan switch menguji nilai satu variabel dan mengeksekusi blok pernyataan untuk nilai yang cocok dari variabel. Format umumnya adalah sebagai berikut:

```
switch ( $variablename )
{
        case value :
                block of statements;
                break;
        case value :
                block of statements;
                break;
        ...
        default:
                block of statements;
                break;
}
```

Pernyataan switch menguji nilai $variablename. Script kemudian melompat ke bagian kasus untuk nilai itu dan mengeksekusi pernyataan hingga mencapai pernyataan break atau akhir dari pernyataan switch. Jika tidak ada bagian case untuk nilai $variablename, skrip akan mengeksekusi bagian default. Kita dapat menggunakan bagian kasus sebanyak yang kita butuhkan. Bagian default adalah opsional. Jika kita menggunakan bagian default, biasanya bagian default diletakkan di akhir, tetapi sejauh menyangkut PHP, itu bisa pergi ke mana saja. Pernyataan berikut menetapkan tarif pajak penjualan untuk negara yang berbeda:

```
switch ( $custState )
{
        case “KR” :
                $salestaxrate = 0;
                break;
        case “ID” :
                $salestaxrate = 1.0;
                break;
        default:
                $salestaxrate = .5;
                break;
}
$salestax = $orderTotalCost * $salestaxrate;
```

Dalam hal ini, tarif pajak untuk Semarang adalah 0, tarif pajak untuk Jepara adalah 100 persen, dan tarif pajak untuk semua kota lainnya adalah 50 persen. Pernyataan switch melihat nilai $custState dan melompat ke bagian yang cocok dengan nilainya. Misalnya, jika $custState adalah ID, skrip akan mengeksekusi bagian default dan menetapkan $salestaxrate ke .5. Setelah pernyataan peralihan, skrip menghitung $salestax sebesar .5 kali biaya pesanan.
Catatan: Dengan pernyataan switch, kita tidak menggunakan kurung kurawal di dalam perintah huruf besar/kecil. Sebaliknya, mereka mulai dengan titik dua dan diakhiri dengan pernyataan break. Seluruh daftar kasus dalam pernyataan switch terlampir dalam satu set kurung kurawal.

Keluar

Jika kita ingin keluar dari pernyataan switch karena kondisi telah terpenuhi, gunakan perintah break. Perintah ini memberitahu PHP untuk keluar dari switch dan melompat ke pernyataan berikut. Jika kita mengabaikan perintah break pada Contoh diatas dan kasus Home dievaluasi sebagai TRUE, maka kelima kasus tersebut akan dieksekusi. Atau jika $page memiliki nilai News, maka semua perintah case sejak saat itu akan dijalankan. Ini disengaja dan memungkinkan untuk beberapa pemrograman tingkat lanjut, tetapi umumnya kita harus selalu ingat untuk mengeluarkan perintah break setiap kali satu set case conditional selesai dieksekusi. Faktanya, mengabaikan pernyataan break adalah kesalahan umum.

Tindakan Default

Persyaratan umum dalam pernyataan switch adalah mundur ke tindakan default jika tidak ada kondisi kasus yang terpenuhi. Misalnya, dalam hal kode menu pada Contoh di atas, kita dapat menambahkan kode tepat sebelum tanda kurung kurawal terakhir seperti pada contoh dibawah ini.

Contoh Pernyataan* default *untuk ditambahkan ke Contoh sebelumnya

```
default:

        echo "Unrecognized selection";
```

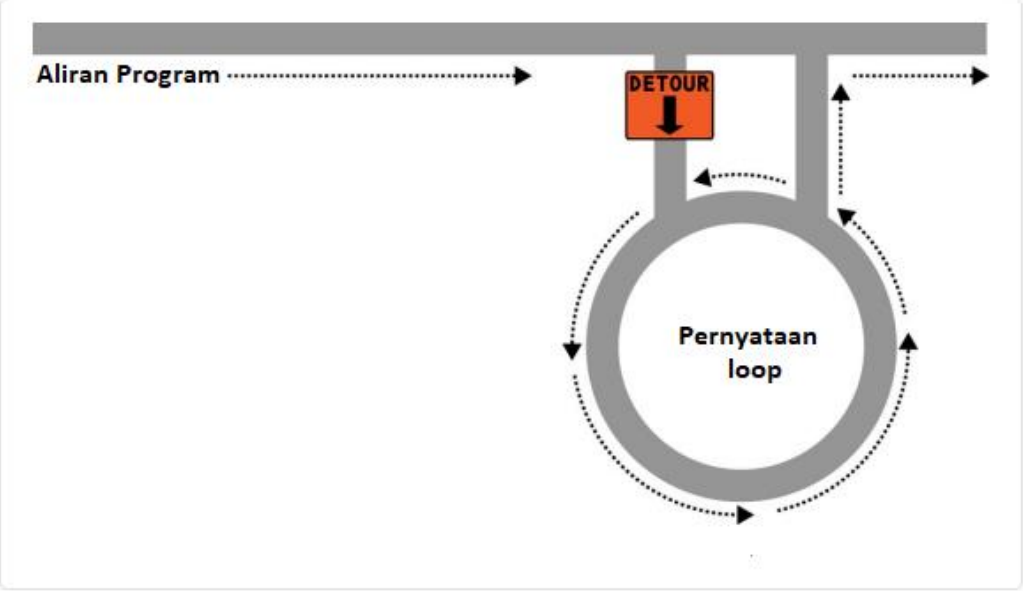
break;

Pernyataan break sangat penting untuk mengakhiri bagian kasus. Jika bagian case tidak menyertakan pernyataan break, skrip tidak berhenti mengeksekusi pernyataan di akhir bagian case. Script terus mengeksekusi pernyataan melewati akhir bagian case, ke bagian kasus berikutnya, dan berlanjut hingga mencapai pernyataan break atau akhir dari pernyataan switch. Ini adalah masalah untuk setiap bagian kasus kecuali yang terakhir karena akan mengeksekusi bagian yang mengikuti bagian yang sesuai. Dalam beberapa case yang jarang terjadi, kita mungkin ingin dua bagian kasus dijalankan ketika variabel switch cocok dengan nilai bagian kasus pertama, jadi kita dapat mengabaikan pernyataan break di bagian kasus pertama. Ini bukan situasi yang umum, tetapi kadang-kadang dapat memecahkan masalah. Bagian case terakhir dalam pernyataan switch sebenarnya tidak memerlukan pernyataan break. Kita dapat meninggalkannya. Namun, ada baiknya untuk memasukkannya untuk kejelasan dan konsistensi.

Meskipun perintah break tidak diperlukan di sini karena defaultnya adalah sub-pernyataan terakhir, dan alur program akan secara otomatis melanjutkan ke kurung kurawal penutup, jika kita memutuskan untuk menempatkan pernyataan default lebih tinggi, itu pasti memerlukan perintah break untuk mencegah program mengalir dari jatuh ke pernyataan berikut. Umumnya, praktik teraman adalah selalu menyertakan perintah break.

3.7 PERULANGAN (*LOOPING*)

Salah satu hal hebat tentang komputer adalah mereka dapat mengulangi tugas menghitung dengan cepat dan tanpa lelah. Seringkali kita mungkin ingin sebuah program mengulangi urutan kode yang sama berulang-ulang sampai sesuatu terjadi, seperti pengguna memasukkan nilai atau mencapai akhir alami. Berbagai struktur loop PHP menyediakan cara sempurna untuk melakukan ini. Untuk menggambarkan cara kerjanya, lihat gambar dibawah. Ini hampir sama dengan metafora jalan raya yang digunakan untuk mengilustrasikan pernyataan if, kecuali bahwa jalan memutar juga memiliki bagian lingkaran yang—setelah kendaraan masuk—hanya dapat keluar di bawah kondisi program yang tepat.



Gambar 3.6 Membayangkan loop sebagai bagian dari program tata letak jalan raya

Mengulangi Tindakan dengan Loop

Loop sering digunakan dalam skrip untuk mengatur blok pernyataan yang berulang. Loop dapat diulang beberapa kali. Misalnya, loop yang menggemakan semua ibu kota negara perlu diulang sebanyak 50 kali. Atau loop dapat diulang sampai kondisi tertentu terpenuhi. Misalnya, loop yang menggemakan nama semua file dalam direktori perlu diulang sampai kehabisan file, terlepas dari berapa banyak file yang ada.
Berikut adalah tiga jenis loop:

- A for loop: Mengatur penghitung; mengulangi blok pernyataan sampai penghitung mencapai nomor tertentu.
- a while loop: Mengatur kondisi; memeriksa kondisi, dan jika benar, mengulangi blok pernyataan sampai kondisi menjadi salah.
- Do..while loop: Mengatur kondisi; mengeksekusi blok pernyataan; memeriksa kondisi, dan jika benar, mengulangi blok pernyataan sampai kondisi menjadi salah.

Kita menjelaskan masing-masing loop ini secara rinci dalam beberapa bagian berikut.

for Loop

Jenis terakhir dari pernyataan loop, for loop, juga yang paling kuat, karena menggabungkan kemampuan untuk mengatur variabel saat kita memasuki loop, menguji kondisi saat while loop, dan memodifikasi variabel setelah setiap iterasi. Contoh dibawah ini menunjukkan bagaimana kita bisa menulis program tabel perkalian dengan for loop.

***Contoh Mengeluarkan tabel perkalian untuk 12 dari* for loop**

```
<?php
    for ($count = 1 ; $count <= 12 ; ++$count)
        echo "$count times 12 is " . $count * 12 . "<br>";
?>
```

Lihat bagaimana seluruh kode telah direduksi menjadi satu pernyataan for yang berisi satu pernyataan bersyarat? Inilah yang sedang terjadi. Setiap pernyataan for membutuhkan tiga parameter:

- Ekspresi inisialisasi
- Ekspresi kondisi
- Ekspresi modifikasi

Ini dipisahkan oleh titik koma seperti ini: for (expr1 ; expr2 ; expr3). Pada awal iterasi pertama dari loop, ekspresi inisialisasi dijalankan. Dalam kasus kode tabel perkalian, $count diinisialisasi ke nilai 1. Kemudian, setiap kali di sekitar loop, ekspresi kondisi (dalam hal ini, $count <= 12) diuji, dan loop dimasukkan hanya jika syaratnya BENAR. Akhirnya, pada akhir setiap iterasi, ekspresi modifikasi dieksekusi. Dalam kasus kode tabel perkalian, variabel $count bertambah. Semua struktur ini dengan rapi menghilangkan persyaratan apa pun untuk menempatkan kontrol untuk loop di dalam tubuhnya, membebaskannya hanya untuk pernyataan yang kita inginkan agar dilakukan loop. Ingatlah untuk menggunakan kurung kurawal dengan for loop jika akan berisi lebih dari satu pernyataan, seperti pada contoh dibawah ini.

***Menggunakan* for loop**

For loop yang paling dasar didasarkan pada penghitung. Kita mengatur nilai untuk penghitung, mengatur nilai akhir, dan mengatur bagaimana penghitung bertambah setiap kali blok pernyataan dieksekusi. Kita memberi tahu kita cara membangun loop dasar itu, dan kemudian bagaimana membuat satu loop di dalam loop lain dan juga membangun loop yang lebih canggih.

Contoh perulangan untuk dari contoh sebelumnya dengan menambahkan kurung kurawal

```
<?php
   for ($count = 1 ; $count <= 12 ; ++$count)
   {
      echo "$count times 12 is " . $count * 12;
      echo "<br>";
   }
?>
```

***Membangun* for loop**

Format umum for loop dasar adalah sebagai berikut:

- ***Startingvalue:*** Nilai awal adalah pernyataan yang menyiapkan variabel untuk menjadi penghitung kita dan menetapkannya ke nilai awal Anda. Misalnya, pernyataan $i=1; menetapkan $i sebagai variabel penghitung dan menyetelnya sama dengan 1. Seringkali, variabel penghitung dimulai pada 0 atau 1. Nilai awal dapat berupa angka, kombinasi angka (seperti 2 + 2), atau variabel.
- ***endingcondition***: Kondisi akhir adalah pernyataan yang menetapkan nilia akhir. Selama pernyataan ini benar, blok pernyataan terus berulang. Ketika pernyataan ini tidak benar, loop berakhir. Misalnya, pernyataan $i<10; menetapkan nilai akhir untuk perulangan menjadi 10. Ketika $i sama dengan 10, pernyataan tersebut tidak lagi benar (karena $i tidak kurang dari 10), dan perulangan berhenti berulang. Pernyataan dapat menyertakan variabel, seperti $i<$size;.
- ***increment***: Pernyataan ini menambah counter Anda. Misalnya, pernyataan $i++; menambahkan 1 ke penghitung kita di akhir setiap blok pernyataan. Kita dapat menggunakan pernyataan kenaikan lainnya, seperti $i=+1;atau $i--;.

Perulangan dasar untuk mengatur variabel, seperti $i, yang digunakan sebagai penghitung. Variabel ini memiliki nilai yang berubah selama setiap loop. Variabel $i dapat digunakan dalam blok pernyataan yang berulang. Misalnya, loop sederhana berikut menampilkan Hello World! tiga kali:

```
for ($i=1;$i<=3;$i++)
{
   echo “$i. Hello World!”<br>”;
}
```

Berikut adalah output dari pernyataan-pernyataan tersebut:

1. Hello World!
2. Hello World!
3. Hello World!

Nesting for loop

Kita dapat meletakkan loop didalam for loop. Kita dapat mengikuti pernyataan ini :

```
for($i=1;$i<=9;$i++)
{
   echo “\nMultiply by $i \n”;
   for($j=1;$j<=9;$j++)
   {
      $result = $i * $j;
      echo “$i x $j = $result\n”;
}
```

```
}
```

Outpunya akan mengikuti :

```
Dikalikan dengan 1
1 x 1 = 1
1 x 2 = 2
...
1 x 8 = 8
1 x 9 = 9

Dikalikan dengan 2
2 x 1 = 2
2 x 2 = 4
...
2 x 8 = 16
2 x 9 = 18

Dikalikan dengan 3
3 x 1 = 3
```

Dan seterusnya.

***Merancang lanjutan* for loop**

Struktur for loop cukup fleksibel dan memungkinkan kita membuat loop untuk hampir semua tujuan. Meskipun dasar for loop yang dibahas sejauh ini di bagian ini memiliki satu pernyataan di bagian awal, kondisional, dan kenaikannya, format umum memungkinkan lebih dari satu pernyataan di setiap bagian. Format umumnya adalah:

```
for (pernyataan awal; pernyataan bersyarat;
 pernyataan akhir)
{
 blok pernyataan;
}
```

Pernyataan dalam for loop memiliki peran berikut:

- Pernyataan awal dieksekusi sekali pada awal loop. Mereka dapat berupa pernyataan yang menetapkan nilai awal yang diperlukan atau pernyataan lain yang ingin kita jalankan sebelum loop kita mulai berjalan.
- Pernyataan bersyarat diuji untuk setiap iterasi dari loop Anda.
- Pernyataan akhir dieksekusi sekali di akhir loop. Itu bisa berupa pernyataan yang meningkatkan nilai kita atau pernyataan lain yang ingin kita jalankan di akhir loop Anda.

Setiap bagian pernyataan dipisahkan oleh titik koma (;). Setiap bagian dapat berisi pernyataan sebanyak yang diperlukan, dipisahkan dengan koma. Setiap bagian bisa kosong.

Loop berikut memiliki pernyataan di ketiga bagian:

```
$t = 0;
for ($i=0,$j=1;$t<=4;$i++,$j++)
{
    $t = $i + $j;
    echo “$t</br>”;
```

```
}
```

Dalam contoh ini, $i=0 dan $j=1 adalah pernyataan awal, $t<=4 adalah pernyataan bersyarat, dan $i++ dan $j++ adalah pernyataan akhir.
Output dari pernyataan-pernyataan tersebut adalah sebagai berikut:

```
1
3
5
```

Loop dijalankan dalam urutan berikut:

1. Bagian awal yang berisi dua pernyataan dieksekusi. $i diatur ke 0, dan $j diatur ke 1.
2. Bagian kondisional yang berisi satu pernyataan dievaluasi. Apakah $t kurang dari atau sama dengan 4? Ya, jadi pernyataan itu benar. Loop terus dieksekusi.
3. Pernyataan dalam blok pernyataan dieksekusi. $t menjadi sama dengan $i ditambah $j, yaitu 0 + 1, yang sama dengan 1. Kemudian $t digaungkan untuk memberikan output 1.
4. Bagian akhir yang berisi dua pernyataan ($i++ dan $j++) dieksekusi. Baik $i dan $j bertambah 1, jadi $i sekarang sama dengan 1, dan $j sekarang sama dengan 2.
5. Bagian kondisional dievaluasi. Apakah $t kurang dari atau sama dengan 4? Karena $t sama dengan 1 pada titik ini, pernyataan tersebut benar. Loop terus dieksekusi
1. Pernyataan dalam blok pernyataan dieksekusi. $t menjadi sama dengan $i ditambah $j, yaitu 1 + 2, yang sama dengan 3. Kemudian $t digaungkan untuk memberikan output 3.
6. Bagian akhir yang berisi dua pernyataan ($i++ dan $j++) dieksekusi. Baik $i dan $j bertambah 1, jadi $i sekarang sama dengan 2, dan $j sekarang sama dengan 3.
7. Bagian kondisional dievaluasi. Apakah $t kurang dari atau sama dengan 4? Karena $t sekarang sama dengan 3, pernyataan tersebut benar. Loop terus dieksekusi.
8. Pernyataan dalam blok pernyataan dieksekusi. $t menjadi sama dengan $i ditambah $j, yaitu 2 + 3, yang sama dengan 5. Kemudian $t digaungkan untuk memberikan output 5.
9. Bagian akhir yang berisi dua pernyataan ($i++ dan $j++) dieksekusi. Baik $i dan $j bertambah 1, jadi $i sekarang sama dengan 2, dan $j sekarang sama dengan 3.
10. Bagian kondisional dievaluasi. Apakah $t kurang dari atau sama dengan 4? Karena $t sekarang sama dengan 5, pernyataan tersebut tidak benar. Loop tidak terus dijalankan. Loop berakhir, dan skrip melanjutkan ke pernyataan berikutnya setelah akhir loop.

While loop

Menggunakan while loop

Perulangan while akan terus berulang selama kondisi tertentu benar. Loop bekerja sebagai berikut:

1. Kita mengatur kondisi.
2. Kondisi diuji di bagian atas setiap loop.
3. Jika kondisinya benar, perulangan akan berulang. Jika kondisi tidak benar, loop berhenti.

Berikut ini adalah format umum dari while loop:

```
while ( condition )
{
   block of statements
```

```
}
```

***Contoh* while loop:**

```
<?php
    $fuel = 10;
    while ($fuel > 1)
    {
        // Keep driving ...
        echo "There's enough fuel"; }
?>
```

Sebenarnya, kita mungkin lebih memilih untuk tetap menyalakan lampu hijau daripada teks output, tetapi intinya adalah bahwa indikasi positif apa pun yang ingin kita buat tentang tingkat bahan bakar ditempatkan di dalam while loop. Omong-omong, jika kita mencoba contoh ini sendiri, perhatikan bahwa itu akan terus mencetak string sampai kita mengklik tombol Stop di browser Anda.

Contoh* while loop *untuk mencetak tabel perkalian 12

```
<?php
$count = 1;
while ($count <= 12)
{
    echo "$count times 12 is " . $count * 12 . "<br>";
    ++$count;
}
?>
```

Di sini variabel $count diinisialisasi ke nilai 1, kemudian loop while dimulai dengan ekspresi komparatif $count <= 12. Loop ini akan terus dieksekusi hingga variabel lebih besar dari 12. Output dari kode ini adalah sebagai berikut :

```
1 times 12 is 12
2 times 12 is 24
3 times 12 is 36
and so on...
```

Di dalam loop, sebuah string diprint bersama dengan nilai $count dikalikan 12. Untuk kerapian, ini juga diikuti dengan tag
 untuk memaksa baris baru. Kemudian $count bertambah, siap untuk kurung kurawal terakhir yang memberitahu PHP untuk kembali ke awal loop. Pada titik ini, $count diuji lagi untuk melihat apakah lebih besar dari 12. Tidak, tetapi sekarang memiliki nilai 2, dan setelah 11 kali putaran berikutnya, ia akan memiliki nilai 13. Ketika itu terjadi, kode dalam while loop dilewati dan eksekusi diteruskan ke kode yang mengikuti loop, yang dalam hal ini adalah akhir dari program.

Jika pernyataan ++$count (yang bisa juga $count++) tidak ada, loop ini akan seperti yang pertama di bagian ini. Itu tidak akan pernah berakhir dan hanya hasil 1 * 12 yang akan diprint berulang-ulang. Tetapi ada cara yang lebih rapi untuk menulis loop ini, dan prediksi saya, kita akan menyukainya.

Contoh Versi singkat dari Contoh sebelumnya

```
<?php
    $count = 0;
    while (++$count <= 12)
        echo "$count times 12 is " . $count * 12 . "<br>";
?>
```

Dalam contoh ini, dimungkinkan untuk menghapus pernyataan ++$count dari dalam while loop dan menempatkannya langsung ke ekspresi kondisional loop. Apa yang sekarang terjadi adalah bahwa PHP menemukan variabel $count pada awal setiap iterasi dari loop dan, memperhatikan bahwa itu diawali dengan operator increment, pertama-tama menambah variabel dan baru kemudian membandingkannya dengan nilai 12. Oleh karena itu kita dapat melihat $count sekarang harus diinisialisasi ke 0, bukan 1, karena bertambah segera setelah loop dimasukkan. Jika kita mempertahankan inisialisasi pada 1, hanya hasil antara 2 dan 12 yang akan keluar.

Mari kita bandingkan kapan menggunakan for dan while loop. Perulangan for secara eksplisit dirancang di sekitar nilai tunggal yang berubah secara teratur. Biasanya kita memiliki nilai yang meningkat, seperti ketika kita melewati daftar pilihan pengguna dan ingin memproses setiap pilihan secara bergantian. Tetapi kita dapat mengubah variabel sesuka Anda. Bentuk pernyataan for yang lebih kompleks bahkan memungkinkan kita melakukan beberapa operasi di masing-masing dari tiga parameter:

```
for ($i = 1, $j = 1 ; $i + $j < 10 ; $i++ , $j++)
{
  // ...
}
```

Itu rumit dan tidak disarankan untuk pengguna pertama kali. Kuncinya adalah membedakan koma dari titik koma. Ketiga parameter tersebut harus dipisahkan dengan titik koma. Dalam setiap parameter, beberapa pernyataan dapat dipisahkan dengan koma. Jadi, pada contoh sebelumnya, parameter pertama dan ketiga masing-masing berisi dua pernyataan:

```
$i = 1, $j = 1 // Initialize $i and $j
$i + $j < 10 // Terminating condition
$i++ , $j++ // Modify $i and $j at the end of each iteration
```

Hal utama yang dapat diambil dari contoh ini adalah kita harus memisahkan tiga bagian parameter dengan titik koma, bukan koma (yang harus digunakan hanya untuk memisahkan pernyataan dalam bagian parameter). Jadi, kapan pernyataan while lebih tepat daripada pernyataan for? Ketika kondisi kita tidak bergantung pada perubahan sederhana dan teratur pada suatu variabel. Misalnya, jika kita ingin memeriksa beberapa input atau kesalahan khusus dan mengakhiri perulangan saat itu terjadi, gunakan pernyataan while.
Pernyataan berikut mengatur while loop yang melihat melalui array untuk sebuah apel:

```
$fruit = array ( “jeruk”, “apel”, “anggur” );
$testvar = “bukan”;
$k = 0;
while ( $testvar != “ya” )
{
```

```
    if ($fruit[$k] == “apel” )
    {
        $testvar = “ya”;
        echo “apel\n”;
    }
    else
    {
    echo “$fruit[$k] bukan sebuah apel\n”;
    }
    $k++;
}
```

Pernyataan ini menghasilkan output berikut:

```
Jeruk bukanlah apel
apel
```

Script mengeksekusi pernyataan sebagai berikut:

1. Variabel diatur sebelum memulai loop. $fruit adalah larik dengan tiga nilai, $testvar adalah variabel uji yang disetel ke "tidak", dan $k adalah variabel penghitung yang disetel ke 0.
2. Perulangan dimulai dengan menguji apakah $testvar != “ya” benar. Karena $testvar disetel ke “tidak”, pernyataan tersebut benar, maka perulangan berlanjut.
3. Kondisi dalam pernyataan if diuji. Apakah $fruit[$k] == “apel” benar? Pada titik ini, $k adalah 0, jadi skrip memeriksa $fruit[0]. Karena $fruit[0] adalah “jeruk”, pernyataan tersebut tidak benar. Pernyataan di blok if tidak dieksekusi, jadi skrip melompat ke pernyataan else.
4. Pernyataan di blok else dieksekusi. Blok else mengeluarkan baris “jeruk bukanlah apel”. Ini adalah baris pertama dari output.
5. $k bertambah satu. Sekarang $k menjadi sama dengan 1.
6. Bagian bawah loop tercapai. Aliran kembali ke bagian atas while loop.
7. Kondisi $testvar != “ya” diuji kembali. Apakah $testvar != “ya” benar? Karena $testvar belum diubah dan masih disetel ke "tidak", itu benar, jadi loop berlanjut.
8. Kondisi pada pernyataan if diuji kembali. Apakah $fruit[$k] == “apel” benar? Pada titik ini, $k adalah 1, jadi skrip memeriksa $fruit[1]. Karena $fruit[1] adalah "apel", pernyataan itu benar. Jadi loop memasuki blok if.
9. Pernyataan di blok if dieksekusi. Pernyataan ini mengatur $testvar ke "ya" dan menghasilkan "apel". Ini adalah baris kedua dari output.
10. $k bertambah lagi. Sekarang $k sama dengan 2.
11. Bagian bawah loop tercapai lagi. Sekali lagi, aliran kembali ke puncak perulangan while.
12. Kondisi $testvar != “ya” diuji untuk terakhir kalinya. Apakah $testvar != “ya” benar? Karena $testvar telah diubah dan sekarang disetel ke "ya", itu tidak benar. Loop berhenti.

Untuk menulis loop while yang tidak terbatas disini sangat dimungkinkan — yaitu, loop yang berulang selamanya. Kita dapat dengan mudah, tanpa bermaksud, menulis loop di mana kondisinya selalu benar. Jika kondisi tidak pernah menjadi salah, perulangan tidak pernah berakhir. Untuk diskusi tentang infinite loop, lihat bagian “Menghindari infinite loop.”

Catatan: Seperti halnya pernyataan if, kita akan melihat bahwa kurung kurawal diperlukan untuk menahan pernyataan di dalam pernyataan while, kecuali jika hanya ada satu.

do … while loop

Sedikit variasi pada while loop adalah do ... while loop, digunakan ketika kita ingin blok kode dieksekusi setidaknya sekali dan dibuat kondisional hanya setelah itu. Contoh do ... while ini menunjukkan versi modifikasi dari kode untuk tabel perkalian 12 yang menggunakan perulangan seperti itu.

Contoh* do ... while loop *untuk mencetak tabel perkalian untuk 12

```
<?php
    $count = 1;
    do
        echo "$count times 12 is " . $count * 12 . "<br>";
        while (++$count <= 12);
?>
```

Perhatikan bagaimana kembali menginisialisasi $count ke 1 (bukan 0) karena kode dieksekusi segera, tanpa kesempatan untuk menambah variabel. Selain itu, kodenya terlihat sangat mirip. Tentu saja, jika kita memiliki lebih dari satu pernyataan di dalam do ... while loop, selalu ingat untuk menggunakan kurung kurawal, perhatikan contoh dibawah ini.

Contoh Memperluas contoh sebelumnya untuk menggunakan kurung kurawal

```
<?php
    $count = 1;
    do {
        echo "$count times 12 is " . $count * 12;
        echo "<br>";
    } while (++$count <= 12);
?>
```

Menggunakan do.. while loop

do..while loop sangat mirip dengan while loop. Seperti while loop, do..while loop terus berulang selama kondisi tertentu benar. Tidak seperti while loop, kondisi tersebut diuji di bagian bawah setiap loop. Jika kondisinya benar, loop akan berulang. Ketika kondisi tidak benar, loop kan berhenti.

Format umum do..while loop adalah sebagai berikut:

```
do
{
block of statements
}
while ( condition );
```

Pernyataan berikut mengatur loop yang mencari apel. Skrip ini melakukan hal yang sama seperti skrip di bagian sebelumnya yang menggunakan while loop:

```
$fruit = array ( “jeruk”, “apel”, “anggur” );
$testvar = “tidak”;
$k = 0;
do
{
```

```
    if ($fruit[$k] == “apel” )
    {
        $testvar = “ya”;
        echo “apel\n”;
    }
    else
    {
        echo “$fruit[$k] bukanlah sebuah apel\n”;
    }
    $k++;
} while ( $testvar != “ya” );
```

Output dari pernyataan ini di browser adalah sebagai berikut:

```
Jeruk bukanlah sebuah apel
apel
```

Ini adalah output yang sama yang ditunjukkan untuk contoh while loop. Perbedaan antara while loop dan do.. while loop adalah kondisi yang diperiksa. Dalam while loop, kondisi diperiksa di bagian atas loop. Oleh karena itu, loop tidak akan pernah dieksekusi jika kondisinya tidak pernah benar. Dalam do..while loop, kondisi diperiksa di bagian bawah loop. Oleh karena itu, loop selalu dieksekusi setidaknya sekali, bahkan jika kondisinya tidak pernah benar. Misalnya, dalam loop sebelumnya yang memeriksa apel, anggaplah kondisi asli disetel ke ya, alih-alih tidak, dengan menggunakan pernyataan ini:

```
$testvar = “ya”;
```

Kondisi tes salah dari awal. Hal ini tidak pernah benar. Dalam whileloop, tidak ada output. Blok pernyataan tidak pernah berjalan. Namun, dalam do..while loop, blok pernyataan dijalankan satu kali sebelum kondisi diuji. Jadi, loop while tidak menghasilkan output, tetapi do..while loop menghasilkan output berikut:

```
Jeruk bukanlah apel
```

do..while loop menghasilkan satu baris output sebelum kondisi diuji. Itu tidak menghasilkan baris kedua dari output karena kondisi tes salah.

Menghindari infinite loop

Kita dapat dengan mudah mengatur loop sehingga tidak pernah berhenti. Ini disebut infinite loop. Mereka mengulangi selamanya. Namun, jarang ada orang yang membuat infinite loop dengan sengaja. Biasanya kesalahan dalam pemrograman. Misalnya, sedikit perubahan pada skrip yang mengatur while loop dapat membuatnya menjadi infinite loop. Berikut adalah skrip yang ditampilkan dengan sedikit perubahan:

```
$fruit = array ( “jeruk”, “apel”, “anggur” );
$testvar = “tidak”;
while ( $testvar != “ya” )
{
    $k = 0;
    if ($fruit[$k] == “apel” )
```

```
    { $
        testvar = “ya”;
        echo “apel\n”;
    }
    else
    {
        echo “$fruit[$k] bukan sebuah apel\n”;
    }
    $k++;
}
```

Perubahan kecil adalah memindahkan pernyataan $k = 0; dari luar loop ke dalam loop. Perubahan kecil ini membuatnya menjadi lingkaran tanpa akhir. Skrip yang diubah ini memiliki output berikut:

```
Jeruk bukanlah sebuah apel
Jeruk bukanlah sebuah apel
Jeruk bukanlah sebuah apel
Jeruk bukanlah sebuah apel
...
```

Ini akan berulang selamanya. Setiap kali loop berjalan, ia me-reset $k ke 0. Kemudian ia mendapatkan $fruit[0] dan menggemakannya. Pada akhir loop, $k bertambah menjadi 1. Namun, ketika loop dimulai lagi, $k diatur kembali ke 0. Akibatnya, hanya nilai pertama dalam array, oranye, yang pernah dibaca. Loop tidak pernah sampai ke apel, dan $testvar tidak pernah disetel ke "ya". Lingkaran tidak ada habisnya. Jika kita menguji skrip dan mendapatkan output yang berulang tanpa henti, tidak perlu panik. Lakukan salah satu dari berikut ini:

- Jika kita menggunakan PHP di halaman web: Tunggu. Itu akan berhenti dengan sendirinya dalam waktu singkat. Waktu default adalah 30 detik, tetapi periode batas waktu mungkin telah diubah oleh administrator PHP. Kita juga dapat mengklik tombol Stop pada browser kita untuk menghentikan tampilan di browser Anda.
- Jika kita menggunakan PHP CLI: Tekan Ctrl+C (atau Cmd+C di Mac). Ini menghentikan skrip agar tidak berjalan. Terkadang output akan terus ditampilkan sedikit lebih lama, tetapi akan segera berhenti.

Kemudian cari tahu mengapa loop berulang tanpa henti dan perbaiki. Kesalahan umum yang dapat mengakibatkan loop tak terbatas adalah menggunakan tanda sama dengan tunggal (=) ketika kita bermaksud menggunakan tanda sama dengan ganda (==). Tanda sama dengan tunggal menyimpan nilai dalam variabel; tanda sama ganda menguji apakah dua nilai sama. Kondisi berikut menggunakan tanda sama dengan tunggal selalu benar:

```
while ($testvar = “ya”)
```

Kondisi hanya menetapkan $testvar sama dengan "ya". Ini bukan pertanyaan yang bisa salah. Apa yang mungkin ingin kita tulis adalah ini:

```
while ($testvar == “ya”)
```

Ini adalah pertanyaan yang menanyakan apakah $testvar sama dengan “ya”, yang dapat dijawab benar atau salah. Kesalahan umum lainnya adalah mengabaikan pernyataan yang

menambah penghitung. Misalnya, pada skrip sebelumnya di bagian ini, jika kita mengabaikan pernyataan $k++;, $k selalu 0, dan hasilnya adalah infinite loop.

Keluar dari loop

Terkadang kita ingin skrip kita keluar dari lingkaran. PHP menyediakan dua pernyataan untuk tujuan ini:

- break: Benar-benar keluar dari loop dan melanjutkan dengan pernyataan skrip setelah loop.
- continue: Melompat ke akhir loop tempat kondisi diuji. Jika kondisi tes positif, skrip berlanjut dari atas loop.

Pernyataan break dan continue biasanya digunakan dalam pernyataan kondisional. Secara khusus, break paling sering digunakan dalam pernyataan switch.

Break Loop

Sama seperti kita melihat cara keluar dari pernyataan switch, kita juga bisa keluar dari loop for menggunakan perintah break yang sama. Langkah ini mungkin diperlukan ketika, misalnya, salah satu pernyataan kita mengembalikan kesalahan dan loop tidak dapat melanjutkan eksekusi dengan aman. Satu kasus di mana hal ini mungkin terjadi adalah ketika menulis file mengembalikan kesalahan, mungkin karena disk penuh.

***Contoh Menulis file menggunakan* for loop *dengan* error trapping**

```
<?php
    $fp = fopen("text.txt", 'wb');
    for ($j = 0 ; $j < 100 ; ++$j)
    {
        $written = fwrite($fp, "data");
        if ($written == FALSE) break;
    }
    fclose($fp);
?>
```

Ini adalah bagian paling rumit dari kode yang telah kita lihat sejauh ini, tetapi kita siap untuk itu. Kita akan melihat perintah penanganan file, tetapi untuk saat ini yang perlu kita ketahui adalah bahwa baris pertama membuka file *text.txt* untuk menulis dalam mode biner, dan kemudian mengembalikan pointer ke file dalam variabel $fp, yang kemudian digunakan untuk merujuk ke file yang terbuka.

Loop kemudian mengulangi 100 kali (dari 0 hingga 99) menulis data string ke file. Setelah setiap penulisan, variabel $write diberi nilai oleh fungsi fwrite yang mewakili jumlah karakter yang ditulis dengan benar. Tetapi jika ada kesalahan, fungsi fwrite memberikan nilai FALSE. Perilaku fwrite memudahkan kode untuk memeriksa variabel $written untuk melihat apakah variabel tersebut disetel ke FALSE dan, jika demikian, untuk keluar dari loop ke pernyataan berikut yang menutup file. Jika kita ingin meningkatkan kode, baris:

```
if ($written == FALSE) break;
```

dapat disederhanakan menggunakan operator NOT, seperti ini:

```
if (!$written) break;
```

Faktanya, pasangan pernyataan loop dalam dapat disingkat menjadi pernyataan tunggal berikut:

```
break 2;
```

Perintah break bahkan lebih kuat dari yang kita kira karena jika kita memiliki kode *nesting* lebih dari satu lapisan yang perlu kita keluarkan, kita dapat mengikuti perintah break dengan angka untuk menunjukkan berapa banyak level yang harus ditembus. Pernyataan berikut ini menunjukkan perbedaan antara continue dan break. Potongan kode pertama ini menunjukkan contoh pernyataan break:

```
$counter = 0;
while ( $counter < 5 )
{
    $counter++;
    If ( $counter == 3 )
    {
        echo “break\n”;
        break;
    }
    echo “Last line in loop: counter=$counter\n”;
}
echo “First line after loop\n\n”;
```

Output dari pernyataan ini adalah sebagai berikut:

```
Last line in loop: counter=1
Last line in loop: counter=2
break
First line after loop
```

Perhatikan bahwa loop pertama berakhir pada pernyataan break. Itu berhenti perulangan dan langsung melompat ke pernyataan setelah perulangan. Itu tidak benar dari pernyataan continue.

Pernyataan continue

pernyataan continue sedikit mirip dengan statemen break, kecuali jika di instruksikan PHP untuk berhenti memproses loop saat ini dan pindah ke kanan ke iterasi berikutnya. Jadi, alih-alih keluar dari seluruh loop, PHP hanya keluar dari iterasi saat ini. Pendekatan ini dapat berguna dalam kasus di mana kita tahu tidak ada gunanya melanjutkan eksekusi dalam loop saat ini dan kita ingin menyimpan siklus prosesor atau mencegah kesalahan terjadi dengan bergerak ke kanan ke iterasi loop berikutnya. Dalam contoh yang disajikan dibawah ini, pernyataan continue digunakan untuk mencegah kesalahan pembagian dengan nol dikeluarkan ketika variabel $j memiliki nilai 0.

***Contoh Menjebak kesalahan pembagian dengan nol menggunakan* continue**

```
<?php
  $j = 10;
  while ($j > −10)
  {
      $j--;
      if ($j == 0) continue;
```

```
        echo (10 / $j) . "<br>";
    }
?>
```

Untuk semua nilai $j antara 10 dan 10, dengan pengecualian 0, hasil penghitungan 10 dibagi $j ditampilkan. Tetapi untuk kasus tertentu $j menjadi 0, pernyataan continue dikeluarkan dan eksekusi langsung melompat ke iterasi loop berikutnya.
Kode berikut memberi kita contoh pernyataan continue:

```
$counter = 0;
while ( $counter < 5 )
{
    $counter++;
    If ( $counter == 3 )
    {
        echo “continue\n”;
        continue;
    }
    echo “Last line in loop: counter=$counter\n”;
}
echo “First line after loop\n”;
```

Output dari pernyataan ini adalah sebagai berikut:

```
Last line in loop: counter=1
Last line in loop: counter=2
continue
Last line in loop: counter=4
Last line in loop: counter=5
First line after loop
```

Tidak seperti perulangan pernyataan break, perulangan ini tidak berakhir pada pernyataan continue. Itu hanya menghentikan pengulangan ketiga dari loop dan melompat kembali ke atas loop. Kemudian menyelesaikan loop, dengan pengulangan keempat dan kelima, sebelum pergi ke pernyataan setelah loop. Salah satu kegunaan pernyataan break adalah asuransi terhadap infinite loop. Pernyataan berikut di dalam loop dapat menghentikannya pada titik yang masuk akal:

```
$test4infinity++;
if ($test4infinity > 100 )
{
    break;
}
```

Jika kita yakin bahwa loop kita tidak boleh berulang lebih dari 100 kali, gunakan pernyataan ini untuk menghentikan loop jika tidak ada habisnya. Gunakan nomor apa pun yang tampaknya masuk akal untuk loop yang kita buat.

Memeriksa konten variabel

Terkadang kita hanya perlu mengetahui apakah suatu variabel ada atau jenis data apa yang ada dalam variabel tersebut. Berikut adalah beberapa cara umum untuk menguji variabel:

isset($varname) #

3.8 MENGGUNAKAN FUNGSI

Aplikasi sering melakukan tugas yang sama pada titik yang berbeda dalam skrip atau dalam skrip yang berbeda. Fungsi dirancang untuk memungkinkan kita menggunakan kembali kode yang sama di lokasi yang berbeda. Fungsi adalah sekelompok pernyataan PHP yang melakukan tugas tertentu. Kita dapat menggunakan fungsi di mana pun kita perlu melakukan tugas. Misalnya, kita sering menampilkan logo perusahaan di seluruh situs web kita dengan pernyataan berikut:

```
echo “<p><img src=’Images/logo.jpg’ width=’50’ height=’50’
    hspace=’10’ align=’left’ /></p>”;
echo “<p style=’font-size: x-large’>My Fine Company</p>”;
echo “<p style=’font-style: italic’>quality products</p>”;
```

Daripada mengetik kode ini di setiap tempat di skrip kita di mana kita ingin menampilkan logo Anda, kita dapat membuat fungsi yang berisi pernyataan dan beri nama display_logo. Kemudian, kita bisa menggunakan fungsi tersebut kapan pun kita ingin menampilkan logo Anda. Menggunakan fungsi terlihat seperti ini:

```
display_logo();
```

Kita dapat melihat bahwa menggunakan satu baris ini menghemat banyak pengetikan dan lebih mudah dibaca dan dipahami daripada mengetik pernyataan echo di mana pun logo diperlukan. Di bagian selanjutnya, kita memberi tahu kita cara membuat dan memanggil fungsi, menggunakan variabel di dalam fungsi, meneruskan dan mengembalikan nilai ke dan dari fungsi, dan menyederhanakan pekerjaan kita dengan fungsi bawaan PHP

Membuat fungsi

Kita dapat membuat fungsi dengan memasukkan kode ke dalam blok fungsi. Format umumnya adalah sebagai berikut:

```
function functionname()
{
block of statements;
return;
}
```

Misalnya, kita dapat membuat fungsi display_logo() yang kita bahas di bagian sebelumnya dengan pernyataan berikut:

```
function display_logo()
  {
   echo “<p><img src=’Images/logo.jpg’ width=’50’ height=’50’
      hspace=’10’ align=’left’ /></p>”;
   echo “<p style=’font-size: x-large’>My Fine Company</p>”;
```

```
 echo “<p style=’font-style: italic’>quality products</p>”;
 return;
}
```

Kita kemudian dapat memanggil fungsi di mana saja kita ingin menampilkan logo, sebagai berikut:

```
display_logo();
```

Pernyataan return di akhir fungsi sebelumnya menghentikan fungsi dan mengembalikan kontrol ke skrip utama. Pernyataan return tidak diperlukan di akhir fungsi, karena fungsi tetap berhenti di akhir dan mengembalikan kontrol ke skrip panggilan. Namun, pernyataan pengembalian membuat fungsi lebih mudah dipahami. Kita dapat membuat fungsi dengan pernyataan definisi fungsi di mana saja dalam skrip, tetapi praktik yang biasa dilakukan adalah menyatukan semua fungsi di awal atau akhir skrip. Fungsi yang kita rencanakan untuk digunakan di lebih dari satu skrip dapat ditentukan dalam file terpisah yang kita sertakan dalam skrip apa pun yang perlu menggunakan fungsi tersebut. Menyertakan file dalam skrip dibahas di bagian, “Mengatur Skrip”.

Menggunakan variabel dalam fungsi

Kita dapat membuat dan menggunakan variabel di dalam fungsi Anda. Variabel seperti itu disebut lokal ke fungsi. Namun, variabel tidak tersedia di luar fungsi; itu tidak tersedia untuk skrip utama. Jika kita ingin menggunakan variabel di luar fungsi, kita harus membuat variabel global, bukan lokal, dengan menggunakan pernyataan global. Misalnya, variabel $name dibuat dalam fungsi berikut:

```
function format_name()
{
    $first_name = “Agus”;
    $last_name = “Wibowo”;
    $name = $last_name, “.$first_name;
}
format_name();
echo “$name”;
```

Pernyataan ini tidak menghasilkan output apa pun. Dalam pernyataan echo, $name tidak mengandung nilai apapun. Variabel $name dibuat di dalam fungsi, jadi tidak ada di luar fungsi. Kita dapat membuat variabel di dalam fungsi yang memang ada di luar fungsi dengan menggunakan pernyataan global. Pernyataan berikut berisi fungsi yang sama dengan pernyataan global yang ditambahkan:

```
function format_name()
{
    Global $name
    $first_name = “Agus”;
    $last_name = “Wibowo”;
    $name = $last_name . “, “.$first_name;
}
format_name();
echo “$name”;
```

Script sekarang menggemakan ini:

Wibowo, Agus

Kita harus membuat variabel global sebelum kita dapat menggunakannya. Jika pernyataan global mengikuti pernyataan penugasan $name, skrip tidak menghasilkan output apa pun. Artinya, pada fungsi sebelumnya, jika pernyataan global mengikuti pernyataan $name =, fungsi tersebut tidak akan bekerja dengan benar. Demikian pula, jika variabel dibuat di luar fungsi, kita tidak dapat menggunakannya di dalam fungsi kecuali jika bersifat global. Dalam pernyataan berikut, satu-satunya pernyataan global ada di dalam fungsi:

```
$first_name = “Agus”;
$last_name = “Wibowo”;
function format_name()
{
    global $first_name, $last_name;
    $name = $last_name.”, “.$first_name;
    echo “$name”;
}
format_name();
```

Karena kode tidak menyertakan pernyataan global di luar fungsi, $last_name dan $first_name di dalam fungsi adalah variabel yang berbeda dari $last_name dan $first_name yang dibuat dalam skrip di luar fungsi. Variabel $last_name dan $first_name di dalam fungsi dibuat saat kita menamainya dan tidak memiliki nilai. Oleh karena itu, $name hanya menggemakan koma, sebagai berikut:

```
,
```

Kita memerlukan pernyataan global agar fungsi berfungsi dengan benar.

Melewati nilai ke fungsi

Kita meneruskan nilai ke suatu fungsi dengan meletakkan nilai di antara tanda kurung saat kita memanggil fungsi, sebagai berikut:

functionname(value,value,...);

Tentu saja, variabel tidak bisa muncul begitu saja. Fungsinya harus menunggu mereka. Pernyataan fungsi menyertakan nama variabel untuk nilai yang diharapkannya, sebagai berikut:

```
function functionname($varname1,$varname2,...)
{
    statements
    return;
    }
```

Misalnya, fungsi berikut menghitung pajak penjualan:

```
function compute_salestax($amount,$custState)
{
switch ( $custState )
```

```
{
      case “KR” :
            $salestaxrate = 0;
            break;
      case “ID” :
      $salestaxrate = 1.0;
      break;
      default:
            $salestaxrate = .5;
            break;
      }
      $salestax = $amount * $salestaxrate;
      echo “$salestax</br>”;
}
```

Baris pertama menunjukkan bahwa fungsi mengharapkan dua nilai — $amount dan $custState. Saat kita memanggil fungsi, kita memberikannya dua nilai, sebagai berikut:

```
$amount = 2000.00;
$custState = “ID”;
compute_salestax($amount,$custState);
```

Dalam hal ini, jumlah yang diteruskan adalah 2000,00 dan statusnya adalah ID. Hasil adalah 2000, karena salestaxrate untuk ID adalah 1,0.

Melewati jenis nilai yang tepat

Kita dapat meneruskan nilai secara langsung, termasuk nilai yang dihitung, atau kita dapat meneruskan variabel yang berisi nilai. Panggilan berikut ini valid:

```
compute_salestax(2000,”ID”);
compute_salestax(2*1000,””);
compute_salestax(2000,”I”.”D”);
```

Kita dapat melewatkan nilai dari tipe data apa pun. Umumnya, kita ingin menguji nilai yang diteruskan untuk memeriksa apakah nilai adalah tipe data yang diharapkan. Misalnya, fungsi berikut mengharapkan array:

```
function add_numbers($numbers)
{
    if(is_array($numbers))
    {
        for($i=0;$i <sizeof($numbers);$i++)
        {
            @$sum = $sum + $numbers[$i];
        }
        echo $sum;
    }
    else
    {
 echo “value passed is not an array”;
```

```
 return;
    }
}
```

Kita dapat menggunakan pernyataan berikut untuk memanggil fungsi add_numbers:

```
$arrayofnumbers = array(100,200);
add_numbers($arrayofnumbers);
```

Fungsi menampilkan 300, yang merupakan jumlah dari 100 ditambah 200. Jika nilai yang dilewatkan bukan array, sebagai berikut:

```
add_numbers(100);
```

fungsi menampilkan pesan:

```
value passed is not an array
```

Melewati nilai dalam urutan yang benar

Fungsi menerima nilai-nilai dalam urutan yang mereka lewati. Artinya, misalkan kita memiliki fungsi berikut:

```
function functionx($x,$y,$z)
{
 do stuff
}
```

Kita memanggil fungsi, sebagai berikut:

```
functionx($var1,$var2,$var3);
functionx sets $x=$var1, $y=$var2, and $z=$var3
```

Jika nilai yang kita berikan tidak dalam urutan yang diharapkan, fungsi menggunakan nilai yang salah saat melakukan tugas. Misalnya, mungkin definisi kita untuk suatu fungsi untuk menghitung pajak penjualan terlihat seperti berikut:

```
function compute_salestax($orderCost,$custState)
{
      compute tax
}
```

Di sini, $orderCost adalah biaya pesanan, dan $custState adalah status tempat pelanggan tinggal. Tetapi misalkan kita menggunakan panggilan berikut:

```
compute_salestax($custState,$orderCost);
```

Fungsi menggunakan nilai variabel $custState sebagai biaya pesanan, yang disetel ke 0, karena merupakan string. Ini menetapkan $custStatevariable ke nomor di $orderCost, yang tidak akan cocok dengan salah satu kategorinya. Outputnya akan menjadi 0.

Melewati jumlah nilai yang tepat

Suatu fungsi dirancang untuk mengharapkan sejumlah nilai tertentu untuk diteruskan ke sana. Jika kita tidak mengirim nilai yang cukup, fungsi akan menyetel nilai yang hilang ke NULL. Jika kita mengaktifkan level pesan peringatan, pesan peringatan akan ditampilkan. Misalnya, kita memiliki fungsi berikut yang memformat nama:

```
function format_name($first_name,$last_name)
{
  $name = "$last_name, ".$first_name;
  echo $name;
}
```

Fungsi mengharapkan dua nilai untuk diteruskan ke sana. Misalkan kita menyebutnya dengan pernyataan berikut:

```
format_name("Agus");
```

Kita melihat pesan yang mirip dengan berikut ini:

Warning: Missing argument 2 for format_name() in **testing.php** on line **9**

Namun, peringatan tidak menghentikan skrip; itu terus berjalan. Jadi, skrip menghasilkan yang berikut:

```
, Agus
```

Jika kita mengirim terlalu banyak nilai, fungsi akan mengabaikan nilai tambahan. Dalam kebanyakan kasus, kita tidak ingin memberikan jumlah nilai yang salah, meskipun ini dapat berguna dalam beberapa kasus yang jarang terjadi. Kita dapat mengatur nilai default untuk digunakan saat nilai tidak diteruskan. Default ditetapkan saat kita menulis fungsi, sebagai berikut:

```
function add_2_numbers($num1=1,$num2=1)
{
 $total = $num1 + $num2;
 echo "total = $total";
}
```

Jika salah satu atau kedua nilai tidak diteruskan ke fungsi, fungsi menggunakan default yang ditetapkan, tetapi jika nilai diteruskan, digunakan sebagai ganti default. Misalnya, kita dapat menggunakan salah satu panggilan berikut:

```
add_2_numbers(2,2);
add_2_numbers(2);
add_2_numbers();
```

Hasilnya adalah, secara berurutan:

```
$total = 4
$total = 3
$total = 2
```

Melewati nilai dengan referensi

Saat kita memasukkan nilai ke dalam variabel dalam definisi fungsi seperti yang ditunjukkan sejauh ini, kita melewati nilai. Melewati nilai adalah cara paling umum untuk meneruskan nilai ke suatu fungsi, sebagai berikut:

```
function add_1($num1)
{
 $num1 = $num1 + 1;
}
```

Saat melewati nilai, salinan dibuat dari $num1 dan diteruskan ke fungsi. Sementara $num1 diubah di dalam fungsi, dengan menambahkan 1 ke dalamnya, variabel $num1 di luar fungsi tidak berubah. Jadi, jika kita memanggil fungsi dengan pernyataan berikut:

```
 $num1 = 3;
 add_1($num1);
 echo $num1;
```

Outputnya adalah:

3

$num1 masih berisi nilai yang sama seperti sebelum kita memanggil fungsi. Kita dapat mengubahnya dengan membuat variabel global di dalam fungsi atau dengan mengembalikan $num1 dari fungsi setelah diubah dan memanggil fungsi, sebagai berikut:

```
$num1 = add_1($num1);
```

Nilai baru $num1 dikembalikan dari fungsi dan disimpan di $num1di luar fungsi. Dalam beberapa kasus, kita ingin mengubah nilai variabel secara langsung, mengubah nilainya di luar fungsi. Passing by reference digunakan untuk tugas ini. Untuk melewatkan variabel dengan referensi, tambahkan & sebelum nama variabel, sebagai berikut:

```
function add_1(&$num1)
   {
    $num1 = $num1 + 1;
   }
```

Saat kita memanggil fungsi ini, penunjuk ke lokasi variabel akan diteruskan, bukan salinan variabel. Artinya, pemanggilan fungsi meneruskan pointer ke wadah yang disebut $num tempat nilai 3 disimpan. Saat kita mengubah variabel dengan pernyataan di dalam fungsi, nilai di lokasi asli akan berubah. Jadi, jika kita memanggil fungsi dengan pernyataan berikut:

```
   $num1 = 3;
   add_1($num1);
```

```
echo $num1;
```

Outputnya adalah:

```
4
```

Karena kita meneruskan pointer ke variabel, berikut ini tidak masuk akal:

```
add_1(&7);
```

Passing by reference digunakan terutama ketika melewatkan nilai yang sangat besar, seperti objek atau array besar. Lebih efisien untuk melewatkan pointer daripada melewatkan salinan nilai yang sangat besar.

Mengembalikan nilai dari suatu fungsi

Jika kita ingin suatu fungsi mengirim nilai kembali ke skrip utama, gunakan pernyataan return. Skrip utama dapat menempatkan nilai dalam variabel atau menggunakannya dengan cara apa pun yang akan menggunakan nilai apa pun. Untuk mengembalikan nilai dari fungsi, masukkan pernyataan pengembalian ke dalam fungsi. Bentuk umumnya adalah:

```
return value;
```

Misalnya, fungsi yang menambahkan dua angka mungkin terlihat seperti ini:

```
function add_2_numbers($num1,$num2)
{
 $total = $num1 + $num2;
 return $total;
}
```

Total dari dua angka dikembalikan. Kita memanggil fungsi, sebagai berikut:

```
$sum = add_2_numbers(5,6);
```

$sum kemudian sama dengan nilai dalam $total yang dikembalikan dari fungsi — 11. Sebenarnya, kita dapat menggunakan pintasan dan mengirim total kembali ke skrip utama dengan satu pernyataan:

```
return $num1 + $num2;
```

Skrip utama dapat menggunakan nilai dengan cara apa pun yang biasa. Pernyataan berikut menggunakan pemanggilan fungsi dengan cara yang valid:

```
$total_height = add_2_numbers($height1,$height2);
$totalSize = $current_size + add_2_numbers($size1,$size2);
if (add_2_numbers($costSocks,$costShoes)  > 200.00 )
 $echo “No sale”;
```

Pernyataan kembali hanya dapat mengembalikan satu nilai. Namun, nilai yang dikembalikan dapat berupa array, sehingga kita sebenarnya dapat mengembalikan banyak nilai dari suatu

fungsi. Kita dapat menggunakan pernyataan kembali dalam pernyataan bersyarat untuk mengakhiri fungsi, sebagai berikut:

```
function find_value($array,$value)
{
 for($i=1;$i<sizeof($array);$i++)
 {
 if($array[$i] = $value)
 {
 echo "$i. $array[$i]<br />";
 return;
 }
 }
}
```

Fungsi memeriksa array untuk melihat apakah itu berisi nilai. Misalnya, kita dapat memanggil fungsi dengan pernyataan berikut:

```
$names = array("Joe","Sam","Juan");
find_value($names,"Sam");
```

Fungsi mencari melalui nilai-nilai dalam array yang mencari Sam. Jika menemukan Sam, ia berhenti mencari. Output menunjukkan item array tempat Sam ditemukan, sebagai berikut:

```
1. Sam
```

Seringkali fungsi dirancang untuk mengembalikan nilai Boolean (benar atau salah), seperti pada fungsi berikut:

```
function is_over_100($number)
{
 if($number > 100)
 {
 return true;
 }
 else
 {
 return false;
 }
}
```

Angka yang sama dengan atau kurang dari 100 mengembalikan salah; angka lebih dari 100 kembali benar. Desain fungsi umum lainnya mengembalikan nilai jika fungsi berhasil tetapi mengembalikan false jika fungsi tidak berhasil. Misalnya, kita dapat mendesain fungsi find_value sebagai berikut:

```
function find_value($array,$value)
{
 for($i=1;$i<sizeof($array);$i++)
```

```
{
if($array[$i] == $value)
{
return i$;
}
}
return false;
}
```

Jika fungsi menemukan nilai dalam array, ia mengembalikan jumlah elemen array tempat ditemukannya $value. Namun, jika tidak menemukan nilai di mana pun dalam array, ia mengembalikan false.

Menggunakan fungsi bawaan

Banyaknya fungsi bawaan PHP adalah salah satu alasan mengapa PHP begitu kuat dan berguna. Fungsi yang disertakan dengan PHP adalah fungsi normal. Mereka tidak berbeda dengan fungsi yang kita buat sendiri. Hanya saja PHP sudah selesai semua bekerja untuk Anda. Kita dapat memanggil fungsi bawaan PHP dengan cara yang sama seperti kita memanggil fungsi yang kita buat sendiri. Kita menggunakan nama fungsi dan memberikan nilai apa pun yang dibutuhkan fungsi. Kita membahas fungsi PHP tertentu di seluruh buku ini. Misalnya, kita telah membahas beberapa fungsi yang dapat kita gunakan untuk memeriksa apakah suatu variabel ada atau kosong. Berikut adalah beberapa fungsi tersebut:

```
isset($varname) empty($varname)
```

Dokumentasi online PHP menjelaskan semua fungsi bawaan di www.php.net/manual/en/funcref.php. Selain itu, dokumentasi PHP menyediakan fungsi pencarian yang sangat berguna ketika kita mengingat nama fungsi tetapi tidak dapat mengingat sintaks yang tepat. Ketik nama fungsi di kotak teks Cari Untuk di bagian atas halaman web dan pilih Daftar Fungsi dari daftar turun bawah.

3.9 MENGATUR SKRIP

Script adalah serangkaian pernyataan PHP, dan setiap pernyataan melakukan tindakan. PHP dimulai pada awal skrip dan mengeksekusi setiap pernyataan secara bergantian. Beberapa pernyataan adalah pernyataan kompleks yang mengeksekusi pernyataan sederhana secara kondisional atau berulang. Sebuah aplikasi seringkali terdiri dari lebih dari satu script PHP. Secara umum, satu skrip melakukan satu tugas utama. Misalnya, aplikasi mungkin menyertakan skrip untuk menampilkan formulir dan skrip yang menyimpan data dalam database.

Namun, ini adalah pedoman, bukan aturan. Beberapa skrip menampilkan formulir dan memproses data formulir. Setiap skrip harus diatur menjadi beberapa bagian untuk setiap tugas tertentu. Mulailah setiap bagian dengan komentar yang menjelaskan apa yang dilakukan bagian tersebut. Pisahkan bagian satu sama lain dengan baris kosong. Misalnya, skrip login mungkin memiliki bagian sebagai berikut:

```
#display the login form
 statements that display the login form
#check for valid user name and password
 statements that check for valid user name and password
#display first page of website or error message
```

statements that display the site if user had valid login
or error message if login invalid

Tujuannya adalah untuk membuat naskah sejelas dan sejelas mungkin. Skrip perlu dipelihara dan diperbarui selama periode waktu tertentu, seringkali bukan oleh orang yang membuatnya. Semakin jelas dan mudah dipahami, semakin mudah untuk memelihara dan memperbaruinya. Bagian berikut memberi kita beberapa tip dan trik untuk mengatur skrip PHP kita dengan cara yang menyederhanakan tugas pemrograman Anda.

Memisahkan kode tampilan dari kode logika

Salah satu prinsip praktik yang baik untuk menulis aplikasi adalah memisahkan logika pemrograman PHP dari HTML yang menampilkan halaman web. Untuk melakukan ini, HTML yang menampilkan halaman ditempatkan dalam file terpisah. File ini kemudian dapat digunakan dalam skrip di mana pun halaman web perlu ditampilkan. Kita dapat menyimpan kode HTML yang menampilkan formulir dalam file terpisah dan kemudian menggunakan kode tersebut setiap kali formulir perlu ditampilkan. Tidak hanya membuat skrip PHP kita lebih mudah dibaca, tetapi juga membuat perubahan bentuk menjadi lebih sederhana. Kita dapat membuat perubahan hanya di file yang berisi kode HTML daripada harus mencari di mana-mana aplikasi menampilkan formulir dan membuat perubahan di setiap lokasi.

Misalnya, pelanggan kita menambahkan item ke keranjang belanja. Di halaman web keranjang belanja, kita menyertakan dua tombol — satu bertuliskan Lanjutkan Belanja dan satu lagi bertuliskan Log Out. Ketika pengguna mengklik salah satu tombol, skrip PHP berikut dijalankan:

```
<?php
if($button == “Continue Shopping”)
{
 include(“catalog.inc”);
}
else
{
 include(“logout.inc”);
}
?>
```

Jika pengguna mengklik Continue Shopping, file yang berisi kode HTML yang menampilkan katalog akan digunakan. Jika pengguna mengklik tombol Log Out, file yang berisi kode HTML untuk pesan logout akan digunakan.

Menggunakan kembali kode

Praktik lain yang membuat skrip mudah dirawat adalah menggunakan kembali kode. Adalah umum untuk menemukan diri kita mengetik sepuluh baris pernyataan PHP yang sama di beberapa tempat dalam skrip. Kita dapat menyimpan blok kode itu dan menggunakannya kembali di mana pun dibutuhkan. Menyimpan kode yang dapat digunakan kembali secara terpisah membuat skrip lebih mudah dibaca dan dipahami. Selain itu, ketika kode perlu diubah, kita hanya mengubahnya di satu tempat, daripada mengubahnya di selusin tempat berbeda dalam skrip. Kita dapat menggunakan kembali kode dengan menyimpan kode dalam suatu fungsi dan memanggil fungsi tersebut di mana pun kita perlu melakukan tugas tersebut. Cara lain kita dapat menggunakan kembali kode adalah dengan menyimpan kode dalam file terpisah dan memasukkan file tersebut ke dalam skrip jika diperlukan

Pengorganisasian dengan fungsi

Sering-seringlah menggunakan fungsi untuk mengatur skrip Anda. Fungsi berguna ketika skrip kita perlu melakukan tugas yang sama di lokasi yang berulang dalam skrip, di skrip yang berbeda di aplikasi, dan bahkan di aplikasi yang berbeda. Setelah kita menulis fungsi yang melakukan tugas dan kita tahu itu berfungsi, kita dapat menggunakannya di mana pun kita membutuhkannya. Cari peluang untuk menggunakan fungsi. Skrip kita jauh lebih mudah dibaca dan dipahami dengan baris seperti ini:

```
getCustomerName();
```

dibandingkan dengan 20 baris pernyataan yang benar-benar mendapatkan nama pelanggan. Faktanya, setelah kita menulis skrip PHP untuk sementara waktu, kita akan memiliki banyak fungsi yang telah kita tulis untuk berbagai skrip. Sangat sering skrip yang kita tulis dapat menggunakan fungsi yang kita tulis untuk aplikasi lain dua pekerjaan yang lalu. Misalnya, kita bisa membuat fungsi yang disebut getStateNames() yang mengembalikan array yang menyimpan 50 nama negara dalam urutan abjad dan fungsi yang disebut getStateCodes() yang mengembalikan array dengan semua 50 kode singkatan negara dua huruf dalam urutan yang sama. Selalu gunakan nama fungsi deskriptif. Panggilan fungsi dalam skrip kita harus memberi tahu kita apa fungsinya. Nama panjang tidak apa-apa. Kita tidak ingin melihat baris dalam skrip kita yang berbunyi.

```
function1();
```

Bahkan baris seperti berikut ini kurang informatif daripada yang seharusnya:

```
getData();
```

Kita ingin melihat garis seperti ini:

```
getAllCustomerNames();
```

Pengorganisasian dengan menyertakan file

Include pernyataan membawa konten file ke dalam skrip Anda. Dengan demikian, kita dapat memasukkan pernyataan ke dalam file eksternal — file yang terpisah dari file skrip kita — dan menyisipkan file di mana pun kita inginkan dalam skrip dengan pernyataan sertakan. include statement berguna untuk menyimpan statement yang berulang. Berikut adalah beberapa cara untuk menggunakan sertakan file untuk mengatur skrip Anda:

- Masukkan semua atau sebagian besar HTML kita ke dalam file yang disertakan. Misalnya, jika script kita mengirimkan formulir ke browser, masukkan HTML untuk formulir ke dalam file eksternal. Saat kita perlu mengirim formulir, gunakan pernyataan sertakan. Menempatkan HTML ke dalam file yang disertakan adalah ide yang baik jika formulir ditampilkan beberapa kali. Bahkan merupakan ide bagus jika formulir hanya ditampilkan sekali karena itu membuat skrip kita lebih mudah dibaca.
- Masukkan fungsi kita ke dalam file include. Kita tidak memerlukan pernyataan untuk fungsi dalam skrip; kita dapat memasukkannya ke dalam file include. Jika kita memiliki banyak fungsi, atur fungsi terkait menjadi beberapa file include, seperti data_functions.inc dan form_functions.inc. Gunakan pernyataan include di bagian atas skrip Anda, baca hanya fungsi yang digunakan dalam skrip.
- Simpan pernyataan bahwa semua file di situs web kita memiliki kesamaan. Sebagian besar situs web memiliki banyak halaman web dengan banyak elemen yang sama.

Misalnya, semua halaman web dimulai dengan tag <html>, <head>, dan <body>. Jika kita menyimpan pernyataan umum dalam file sertakan, kita dapat menyertakannya di setiap halaman web, memastikan bahwa semua halaman kita terlihat sama. Misalnya, kita mungkin memiliki pernyataan berikut dalam include file:

```
<html>
<head><title><?php echo $title ?></title></head>
<body topmargin="0">
<p style="text-align: center">
 <img src="logo.gif" width="100" height="200">
<hr color="red" />
```

Jika kita menyertakan file ini di bagian atas setiap skrip di situs web Anda, kita menghemat banyak pengetikan, dan kita tahu bahwa semua halaman kita cocok. Selain itu, jika kita ingin mengubah apa pun tentang tampilan semua halaman Anda, kita harus mengubahnya hanya di satu tempat — di file sertakan.

Include dan Membutuhkan File

Seiring kemajuan kita dalam penggunaan pemrograman PHP, kemungkinan besar kita akan mulai membangun pustaka fungsi yang menurut kita akan kita perlukan lagi. Kita juga mungkin akan mulai menggunakan pustaka yang dibuat oleh pemrogram lain. Tidak perlu menyalin dan menempelkan fungsi-fungsi ini ke dalam kode Anda. Kita dapat menyimpannya dalam file terpisah dan menggunakan perintah untuk menariknya. Ada dua jenis perintah untuk melakukan tindakan ini: include dan require.

Including (.inc) file

Kita menggunakan pernyataan include untuk membawa konten file teks eksternal ke dalam skrip Anda. Format untuk pernyataan include adalah:

```
include("filename");
```

File dapat memiliki nama apa pun. Kita, penulis buku kita yang rendah hati, suka menggunakan ekstension.inc sehingga kita tahu bahwa file tersebut adalah file include segera setelah kita melihat namanya. Ini membantu dengan organisasi dan kejelasan situs web.

PHP menyediakan empat jenis pernyataan include:

- include: Termasuk dan mengevaluasi file yang ditentukan. Ini menampilkan peringatan jika tidak dapat menemukan file yang ditentukan.
- require: Melakukan hal yang sama seperti pernyataan include, kecuali bahwa itu menghasilkan, selain peringatan, kesalahan fatal ketika tidak dapat menemukan file yang ditentukan, menghentikan skrip pada saat itu.
- include_once: Melakukan hal yang sama dengan pernyataan include, kecuali menyertakan file hanya sekali. Jika file sudah disertakan, tidak akan disertakan lagi. Dalam beberapa skrip, file mungkin disertakan lebih dari sekali, menyebabkan pendefinisian ulang fungsi, penetapan ulang variabel, dan masalah lain yang mungkin terjadi.
- require_once: Melakukan hal yang sama dengan pernyataan require, kecuali ia hanya menyertakan file satu kali. Jika file sudah disertakan, tidak akan disertakan lagi. Pernyataan ini mencegah masalah yang mungkin terjadi saat file disertakan lebih dari sekali.

File eksternal disertakan dalam skrip kita di lokasi pernyataan include. Isi file dibaca sebagai kode HTML, bukan PHP. Oleh karena itu, jika kita ingin menggunakan pernyataan PHP di file

include Anda, kita harus menyertakan tag PHP di file include. Melupakan tag PHP dalam file include adalah kesalahan umum. Ini juga merupakan masalah keamanan karena tanpa tag PHP, kode dalam file yang disertakan akan ditampilkan kepada pengguna sebagai HTML.

Pernyataan Include

Dengan menggunakan include, kita dapat memberi tahu PHP untuk mengambil file tertentu dan memuat semua kontennya. Seolah-olah kita menempelkan file yang disertakan ke file saat ini di titik penyisipan. Contoh berikut menunjukkan bagaimana kita akan memasukkan file bernama library.php.

Contoh Termasuk file PHP.

```
<?php
    include "library.php";
    // Your code goes here
?>
```

Menggunakan include_once

Setiap kali kita mengeluarkan direktif sertakan, itu menyertakan file yang diminta lagi, bahkan jika kita sudah memasukkannya. Misalnya, misalkan library.php berisi banyak fungsi yang berguna, jadi kita memasukkannya ke dalam file Anda, tetapi juga menyertakan library lain yang menyertakan library.php. Melalui nesting, kita secara tidak sengaja memasukkan library.php dua kali. Ini akan menghasilkan pesan kesalahan, karena kita mencoba mendefinisikan konstanta atau fungsi yang sama beberapa kali. Jadi kita harus menggunakan include_once sebagai gantinya.

Contoh Menyertakan file PHP hanya sekali

```
<?php
    include_once "library.php";
    // Your code goes here
?>
```

Kemudian, setiap kali include lain atau include_once ditemukan, jika sudah dieksekusi, itu akan diabaikan sepenuhnya. Untuk menentukan apakah file telah dieksekusi, jalur file absolut dicocokkan setelah semua jalur relatif diselesaikan dan file ditemukan di jalur sertakan Anda. Catatan: Secara umum, mungkin lebih baik tetap menggunakan include_once dan mengabaikan pernyataan dasar include. Dengan begitu, kita tidak akan pernah mengalami masalah file yang disertakan beberapa kali.

Menggunakan require dan require_once

Masalah potensial dengan include dan include_once adalah PHP hanya akan mencoba memasukkan file yang diminta. Eksekusi program berlanjut meskipun file tidak ditemukan. Ketika sangat penting untuk menyertakan file, mintalah itu. Untuk alasan yang sama yang saya berikan untuk menggunakan include_once, saya sarankan kita biasanya tetap menggunakan require_once setiap kali kita membutuhkan file.

Contoh Membutuhkan file PHP hanya sekali

```
<?php
    require_once "library.php";
    // Your code goes here
?>
```

Menggunakan variabel dalam pernyataan include
Kita dapat menggunakan nama variabel untuk nama file, sebagai berikut:

```
include("$filename");
```

Misalnya, kita mungkin ingin menampilkan pesan yang berbeda pada hari yang berbeda. Kita dapat menyimpan pesan-pesan ini dalam file yang diberi nama untuk hari di mana pesan tersebut akan muncul. Misalnya, kita dapat memiliki file bernama Sun.inc dengan konten berikut:

```
<p>Go ahead. Sleep in. No work today.</p>
```

dan file serupa untuk semua hari dalam seminggu. Pernyataan berikut dapat digunakan untuk menampilkan pesan yang benar untuk hari ini:

```
$today = date("D");
include("$today".".inc");
```

Setelah pernyataan pertama, $today berisi hari dalam seminggu, dalam bentuk singkatan. Pernyataan kedua menyertakan file yang benar, menggunakan hari yang disimpan di $today. Jika $today berisi Sun, pernyataan tersebut menyertakan file bernama Sun.inc.

3.10 MENYIMPAN TERMASUK FILE DENGAN AMAN

Tempat kita menyimpan file yang disertakan dapat menjadi masalah keamanan untuk situs web. File yang disimpan di situs web dapat diunduh oleh pengguna mana pun, kecuali dilindungi. Secara teoritis, pengguna dapat terhubung ke situs web kita dengan menggunakan URL berikut:
http://example.com/secretpasswords.inc
Jika server web dikonfigurasi untuk memproses bagian PHP hanya dalam file dengan ekstensi .php dan secretpasswords.inc berisi pernyataan berikut:

```
<?php
 $mysecretaccount="account48756";
 $mypassword="secret";
?>
```

server web wajib menampilkan isi secretpasswords.inc kepada pengguna. Kita dapat melindungi dari ini dengan salah satu cara berikut:

- Nama termasuk file dengan ekstensi .php. Ini perlu dilakukan dengan hati-hati karena memungkinkan beberapa kode PHP dijalankan secara independen, tanpa konteks apa pun. Misalnya, kita memiliki kode dalam file sertakan kita yang menghapus catatan dalam database (sangat tidak mungkin). Menjalankan kode di luar skrip mungkin memiliki konsekuensi negatif. Selain itu, kita mungkin merasa nyaman untuk memberi nama file dengan ekstensi .inc, sehingga kita dapat melihat sekilas bahwa itu adalah fragmen, bukan skrip yang dimaksudkan untuk berjalan dengan sendirinya.
- Konfigurasikan server web untuk memindai bagian PHP dalam file dengan ekstensi .inc, serta ekstensi .php. Ini memungkinkan kita untuk mengenali file yang disertakan dengan namanya, tetapi masih memiliki masalah kemungkinan konsekuensi yang tidak diinginkan dari menjalankan file secara independen, seperti yang dibahas sebelumnya.

- Simpan file di lokasi yang tidak dapat diakses oleh pengguna luar. Ini adalah solusi yang lebih disukai, tetapi mungkin tidak dapat dilakukan di beberapa lingkungan, seperti saat menggunakan perusahaan hosting web.

Tempat terbaik untuk menyimpan file yang disertakan adalah direktori di mana pengguna luar tidak dapat mengaksesnya. Misalnya, untuk situs web Anda, siapkan direktori include yang berada di luar ruang web Anda: yaitu, direktori di lokasi yang tidak dapat diakses oleh pengguna luar menggunakan browser mereka. Misalnya, ruang web default untuk Apache, kecuali jika telah diubah dalam file konfigurasi (biasanya httpd.conf), adalah htdocs di direktori tempat Apache diinstal. Jika kita menyimpan file sertakan dalam direktori yang tidak ada di ruang web Anda, seperti d:\include, kita melindungi file dari pengguna luar. Untuk menyertakan file dari direktori tersembunyi (seperti direktori di luar ruang web Anda), kita dapat menggunakan nama path lengkap ke file tersebut, sebagai berikut:

include("d:/hidden/secretpasswords.inc");

Namun, PHP memungkinkan kita untuk mengatur direktori include. Kita dapat memasukkan file dari direktori include hanya dengan menggunakan nama file.

Menyiapkan termasuk direktori

PHP mencari include file di direktori saat ini, tempat file halaman web kita disimpan, dan dalam satu atau lebih direktori yang ditentukan oleh pengaturan di file php.ini Anda. Kita dapat menyertakan file dari direktori include tanpa menentukan jalur ke file. Kita dapat melihat lokasi direktori include saat ini dengan menggunakan pernyataan phpinfo(). Di output, di bagian inti PHP, kita dapat menemukan pengaturan untuk include_path yang menunjukkan di mana direktori include kita saat ini berada. Misalnya, di PHP 5, lokasi default mungkin adalah c:\php5\pear. Kita dapat mengubah pengaturan untuk direktori include kita di file php.ini. Temukan pengaturan untuk include_path dan ubah ke jalur ke direktori pilihan Anda, sebagai berikut:

```
include_path=".;c:\php\include"; # for Windows
include_path=".:/user/local/include"; # for Unix/Linux
```

Kedua pernyataan tersebut menentukan dua direktori tempat PHP mencari file include. Direktori pertama adalah dot (artinya direktori saat ini), diikuti oleh direktori kedua, path. Kita dapat menentukan direktori include sebanyak yang kita inginkan dan PHP akan mencarinya, sesuai urutan daftarnya, untuk menemukan file include. Jalur direktori dipisahkan oleh titik koma untuk Windows atau titik dua untuk Unix dan Linux. Jika kita tidak dapat mengatur jalur sendiri di php.ini, kita dapat mengatur jalur di setiap skrip individu dengan menggunakan pernyataan berikut:

```
ini_set("include_path","d:\hidden");
```

Pernyataan menetapkan include_path ke direktori yang ditentukan hanya saat skrip sedang berjalan. Itu tidak mengatur direktori untuk seluruh situs web Anda. Untuk mengakses file dari direktori include, cukup gunakan nama file, sebagai berikut. Kita tidak perlu menggunakan nama path lengkap.

include("secretpasswords.inc");

Jika file include kita tidak ada dalam direktori penyertaan, kita mungkin perlu menggunakan seluruh nama path dalam pernyataan penyertaan. Jika file berada di direktori yang sama dengan skrip, nama file saja sudah cukup. Namun, jika file berada di direktori lain, seperti subdirektori dari direktori yang skripnya berada di dalam atau di direktori tersembunyi di luar ruang web, kita perlu menggunakan nama path lengkap ke file, sebagai berikut:

```
include("d:\hidden\secretpasswords.inc");
```

BAB 4
VERSI PHP DAN SISTEM OPERASI

4.1 MENGELOLA FILE

Informasi yang kita simpan di hard drive kita diatur ke dalam file. Daripada menyimpan file dalam satu laci file besar, membuatnya sulit ditemukan, file disimpan di banyak laci, yang disebut direktori atau folder. Sistem file dan direktori disebut sistem file. Sistem file diatur dalam struktur hierarkis, dengan tingkat teratas yang merupakan direktori tunggal yang disebut root, seperti c:\ di Windows atau / di Linux atau Mac. Direktori root berisi direktori lain, dan setiap direktori dapat berisi direktori lain, dan seterusnya. Struktur sistem file dapat turun ke banyak level.

Direktori adalah jenis file yang kita gunakan untuk mengatur file lain. Ini berisi daftar file dan informasi yang diperlukan untuk sistem operasi untuk menemukan file-file itu. Direktori dapat berisi file dan direktori lain. File dapat diperiksa (untuk melihat apakah ada, misalnya), disalin, dihapus, dan diganti namanya, antara lain. Fungsi untuk melakukan tugas manajemen file ini dijelaskan di bagian berikut. Kita juga mengetahui tentang fungsi yang memungkinkan kita mengelola direktori dan menemukan apa yang ada di dalamnya.

Dalam bab ini, kita membahas fungsi yang paling berguna untuk mengelola file, tetapi lebih banyak fungsi yang tersedia. Saat kita perlu melakukan tindakan pada file atau direktori, periksa dulu dokumentasi PHP online di www.php.net/manual untuk melihat apakah fungsi yang ada melakukan apa yang perlu kita lakukan. Menggunakan fungsi lebih disukai, jika ada fungsi yang sesuai. Jika fungsi tersebut tidak ada, kita dapat menggunakan perintah sistem operasi atau program dalam bahasa lain. Mendapatkan informasi tentang file Seringkali kita ingin mengetahui informasi tentang file. PHP memiliki fungsi yang memungkinkan kita untuk mengetahui informasi file dari dalam skrip. Kita dapat mengetahui apakah ada file dengan pernyataan file_exists, sebagai berikut:

```
$result = file_exists(“stuff.txt”);
```

Setelah pernyataan ini, $result berisi true atau false. Fungsinya sering digunakan dalam pernyataan bersyarat, seperti berikut:

```
if(!file_exists(“stuff.txt”))
{
 echo “File not found!\n”;
}
```

Ketika kita mengetahui file tersebut ada, kita dapat mengetahui informasi tentangnya.

Tabel 4.1 Pernyataan $result yang berisi output true atau false

Fungsi	Apa yang dilakukan	Output
is_file(“stuff.txt”	Menguji apakah file merupakan file reguler, daripada direktori atau jenis file spesifik lainnya	True atau false
is_dir(“stuff.txt”)	Menguji apakah file merupakan sebuahdirektori	True atau false

is_executable(“do.txt”)	Menguji apakah filedapat di eksekusi	True atau false
is_writable(“stuff.txt”)	Menguji apakah kita dapat menulis pada file	True atau false
is_readable(“stuff.txt”)	Menguji apakah kita dapat membaca file	True atau false
fileatime(“stuff.txt”)	Mengembalikan waktu pada file yang di akses terakhir kali	Timestamp unik (seperti 1057196122) atau false
filectime(“stuff.txt”)	Mengembalikan waktu ketika file yang dibuat	Timestamp unik atau false
filemtime(“stuff.txt”)	Mengembalikan waktu ketika file telah dimodifikasi	Timestamp unik atau false
filegroup(“stuff.txt”)	Mengembalikan group ID pada file	Bilangan bulat yang merupakan group user ID atau False
fileowner(“stuff.txt”)	Mengembalikan user ID pada file	Bilangan bulat yang merupakan user ID atau False
filesize(“stuff.txt”)	Mengembalikan ukuran file dalam byte	Bilangan bulat atau false
filetype(“stuff.txt”)	Mengembalikan jenis file	Jenis file (termasuk file, dir) atau false jika error atau tidak dapat di identifikasi
basename(“/t1/do.txt”)	Mengembalikan nama file dari path	do.txt
dirname(“/t1/do.txt”)	Mengembalikan nama direktori dari path	/t1

Fungsi yang mengembalikan informasi berguna tentang jalur/nama file adalah pathinfo(). Kita dapat menggunakan pernyataan berikut:

```
$pinfo = pathinfo(“/topdir/nextdir/stuff.txt”);
```

Setelah pernyataan, $pinfo adalah array yang berisi tiga elemen berikut:

```
$pinfo[dirname] = /topdir/nextdir
$pinfo[basename] = stuff.txt
$pinfo[extension] = txt
```

Saat kita menguji file dengan salah satu fungsi is_something, kesalahan pengetikan apa pun, seperti salah mengeja nama file, memberikan hasil yang salah. Misalnya, is_dir("tyme") mengembalikan false jika "tyme" adalah file, bukan direktori. Tapi, itu juga mengembalikan false jika "tyme" tidak ada karena kita bermaksud mengetik "type". Timestamp Unix dikembalikan oleh beberapa fungsi yang diberikan pada tabel. Kita dapat mengonversi timestamp ini menjadi tanggal dengan fungsi tanggal.

Menyalin, mengganti nama, dan menghapus file

Kita dapat menyalin file yang ada ke file baru. Setelah menyalin, kita memiliki dua salinan file dengan dua nama berbeda. Menyalin file seringkali berguna untuk membuat cadangan file penting. Untuk menyalin file, gunakan pernyataan salin, sebagai berikut:

```
copy(“fileold.txt”,”filenew.txt”);
```

Pernyataan ini menyalin fileold.txt, file yang sudah ada, ke filenew.txt. Jika file dengan nama filenew.txt sudah ada, file tersebut akan ditimpa. Jika kita tidak ingin menimpa file yang sudah ada, kita dapat mencegahnya dengan menggunakan pernyataan berikut:

```
If(!file_exists(“filenew.txt”))
{
 copy(“fileold.txt”,”filenew.txt”);
}
else
{
 echo “File already exists!\n”;
}
```

Kita dapat menyalin file ke direktori yang berbeda dengan menggunakan nama path sebagai tujuan, sebagai berikut:

```
copy(“fileold.txt”,”newdir/filenew.txt”);
```

Kita dapat mengganti nama file dengan menggunakan pernyataan rename, sebagai berikut:

```
rename(“oldname.txt”,”newname.txt”);
```

Jika kita mencoba mengganti nama file dengan nama file yang sudah ada, peringatan ditampilkan, sebagai berikut, dan file tidak diganti namanya:

Warning: rename(fileold.txt,filenew.txt): File exists in **c:test.php** on line **17**

Untuk menghapus file yang tidak diinginkan, gunakan pernyataan unlink, sebagai berikut:

```
unlink(“badfile.txt”);
```

Setelah pernyataan ini, file akan dihapus. Jika file tidak ada untuk memulai, batalkan tautan tidak mengeluh. Ini bertindak sama seperti jika itu telah menghapus file. PHP tidak memberi tahu kita jika file tidak ada. Jadi, hati-hati dengan typo.

4.2 MENGATUR FILE

File diatur ke dalam direktori, juga disebut folder. Bagian ini menjelaskan cara membuat dan menghapus direktori dan cara mendapatkan daftar file dalam direktori.

Membuat direktori

Untuk membuat direktori, gunakan fungsi mkdir, sebagai berikut:

```
mkdir(“testdir”);
```

Pernyataan ini membuat direktori baru bernama testdir di direktori yang sama tempat skrip berada. Artinya, jika skripnya adalah /test/test.php, direktori barunya adalah /test/testdir. Jika direktori sudah ada dengan nama yang sama, peringatan akan ditampilkan, sebagai berikut, dan direktori baru tidak dibuat:

Warning: mkdir(): File exists in **d:/test/test.php** on line **5**

Kita dapat memeriksa terlebih dahulu untuk melihat apakah direktori tersebut sudah ada dengan menggunakan pernyataan berikut:

```
If(!is_dir("mynewdir"))
{
 mkdir("mynewdir");
}
else
{
 echo "Directory already exists!";
}
```

Setelah direktori dibuat, kita dapat mengatur isinya dengan menyalin file ke dalam dan ke luar direktori. Untuk membuat direktori di direktori lain, gunakan seluruh nama path, sebagai berikut:

```
mkdir("/topdir/nextdir/mynewdir");
```

Kita dapat menggunakan jalur relatif untuk membuat direktori baru, sebagai berikut

```
mkdir("../mynewdir");
```

Dengan pernyataan ini, jika skrip kita adalah /topdir/test/makedir.php, direktori baru adalah /topdir/mynewdir. Untuk mengubah ke direktori lain, gunakan pernyataan berikut:

```
chdir("../anotherdir");
```

Membangun daftar semua file dalam direktori

Mendapatkan daftar file dalam direktori seringkali berguna. Misalnya, kita mungkin ingin memberikan daftar file untuk diunduh pengguna atau ingin menampilkan gambar dari file di direktori tertentu. PHP menyediakan fungsi untuk membuka dan membaca direktori. Untuk membuka direktori, gunakan pernyataan opendir, sebagai berikut:

```
$dh = opendir("/topdir/testdir");
```

Jika kita mencoba membuka direktori yang tidak ada, peringatan akan ditampilkan, sebagai berikut:

Warning: opendir(testdir): failed to open dir: Invalid argument in **test13.php** on line **5**

Dalam pernyataan sebelumnya, variabel $dh adalah pegangan direktori, penunjuk ke direktori terbuka yang dapat kita gunakan nanti untuk membaca dari direktori. Untuk membaca nama file dari direktori, gunakan fungsi readdir, sebagai berikut:

```
$filename = readdir($dh);
```

Setelah pernyataan ini, $filename berisi nama file. Hanya nama file yang disimpan dalam $filename, bukan seluruh jalur ke file. Untuk membaca semua nama file dalam sebuah direktori, kita dapat menggunakan while loop, sebagai berikut:

```
while($filename = readdir($dh))
{
 echo $filename."\n";
}
```

Fungsi readdir tidak memberikan kontrol apa pun atas urutan nama file yang dibaca, jadi kita tidak selalu mendapatkan nama file dalam urutan yang kita harapkan. Misalkan kita ingin membuat galeri gambar yang menampilkan semua gambar dalam direktori tertentu di halaman web. Kita dapat menggunakan fungsi opendir dan readdir untuk melakukan ini. Daftar dibawah inimenunjukkan skrip yang membuat galeri gambar.

Skrip yang Membuat Galeri Gambar

```
<?php
    /* Script name: displayGallery
    * Description: Displays all the image files that are
    * stored in a specified directory.
    */
 echo "<html><head><title>Image Gallery</title></head>
      <body>";
 $dir = "../test1/testdir/"; →8
 $dh = opendir($dir); →9
 while($filename = readdir($dh)) →10
 {
      $filepath = $dir.$filename; →12
      if(is_file($filepath) and ereg("\.jpg$",$filename)) →13
      {
      $gallery[] = $filepath;
      }
 }
 sort($gallery); →16
 foreach($gallery as $image) →17
 {
    echo "<hr />";
    echo "<img src='$image' /><br />";
 }
?>
</body></html>
```

Perhatikan nomor baris di akhir beberapa baris dalam daftar skrip diatas. Berikut pembahasan cara kerja skrip mengacu pada nomor baris pada daftar script diatas:

- →8 Baris ini menyimpan nama direktori dalam $dir untuk digunakan nanti dalam program. Perhatikan bahwa / disertakan di akhir nama direktori. Jangan gunakan \, bahkan dengan Windows.
- →9 Baris ini membuka direktori.

- →10 Baris ini memulai while loop yang membaca setiap nama file dalam direktori.
- →12 Baris ini membuat variabel $filepath, yang merupakan path lengkap ke file. Jika / tidak disertakan di akhir nama direktori pada Baris 8, $filepath tidak akan menjadi jalur yang valid.
- →13 Baris ini memeriksa apakah file tersebut adalah file grafik dengan mencari ekstensi .jpg. Jika file memiliki ekstensi .jpg, path file lengkap ditambahkan ke array yang disebut $gallery.
- →16 Baris ini mengurutkan array sehingga gambar ditampilkan dalam urutan abjad.
- →17 Baris ini memulai loop foreach yang menampilkan gambar di halaman web.

4.3 MENGGUNAKAN PERINTAH SISTEM OPERASI

Saat kita perlu berinteraksi dengan sistem operasi Anda, sebaiknya gunakan fungsi PHP yang disediakan untuk tujuan ini. Menggunakan fungsi PHP lebih cepat dan biasanya lebih aman daripada menjalankan perintah sistem operasi secara langsung. Namun, terkadang PHP tidak menyediakan fungsi untuk melakukan tugas yang kita butuhkan. Dalam kasus seperti itu, kita dapat menggunakan fitur PHP yang memungkinkan kita untuk menjalankan perintah sistem operasi.

Di bagian ini, kita berasumsi bahwa kita mengetahui format dan penggunaan perintah sistem untuk sistem operasi Anda. Menjelaskan perintah sistem operasi berada di luar cakupan buku ini. Jika kita perlu menjalankan perintah sistem operasi dari skrip PHP Anda, bagian ini menunjukkan caranya. PHP memungkinkan kita untuk menggunakan perintah sistem atau menjalankan program dalam bahasa lain dengan menggunakan salah satu metode berikut:

- backticks: PHP mengeksekusi perintah sistem yang berada di antara dua backticks (`) dan menampilkan hasilnya.
- fungsi sistem: Fungsi ini mengeksekusi perintah sistem, menampilkan output, dan mengembalikan baris terakhir dari output.
- fungsi exec: Fungsi ini mengeksekusi perintah sistem, menyimpan output dalam array, dan mengembalikan baris terakhir dari output.
- fungsi passthru: Fungsi ini menjalankan perintah sistem dan menampilkan output.

Kita dapat menjalankan perintah apa pun yang dapat kita ketikkan ke prompt sistem. Perintah dijalankan persis seperti apa adanya. Kita dapat menjalankan perintah sederhana: ls atau dir, rename atau mv, rm atau del, meskipun lebih efisien menggunakan fungsi PHP bawaan untuk itu, seperti yang telah dibahas.

Jika sistem operasi kita memungkinkan kita untuk menyalurkan atau mengalihkan output, kita dapat menyalurkan atau mengarahkan ulang dalam perintah sistem yang kita jalankan di PHP. Jika sistem operasi kita memungkinkan kita memasukkan dua perintah dalam satu baris, kita dapat memasukkan dua perintah ke dalam satu perintah yang kita jalankan dari PHP. Contoh perintah berikut ini valid untuk dijalankan dari PHP, tergantung pada sistem operasinya:

```
dir
rm badfile.txt
dir | sort
cd c:\php ; dir (Not valid in Windows)
“cd c:\php && dir” (Windows)
dir > dirfile
sort < unsortedfile.txt
```

Pada beberapa kesempatan, kita ingin menjalankan perintah sistem yang membutuhkan waktu lama untuk diselesaikan. Kita dapat menjalankan perintah sistem di latar belakang (jika sistem operasi kita mendukung hal-hal seperti itu) sementara PHP melanjutkan skrip. Jika kita melakukan ini, kita perlu mengarahkan output ke file, daripada mengembalikannya ke skrip, sehingga PHP dapat melanjutkan sebelum perintah sistem selesai. Bagian berikut menjelaskan metode sebelumnya secara lebih rinci.

Menggunakan backtick

Cara sederhana untuk menjalankan perintah sistem adalah dengan meletakkan perintah di antara dua tanda centang belakang (`), sebagai berikut:

```
$result = `dir c:\php`;
```

Variabel $result berisi output pernyataan — dalam hal ini, daftar file di direktori c:\php. Jika kita echo $result, output berikut akan ditampilkan:

```
Volume in drive C has no label.
Volume Serial Number is 58B2-DBD6
     Directory of c:\php
10/10/2013 05:43 PM <DIR>          .
10/10/2013 05:43 PM <DIR>          ..
10/10/2013 04:53 PM <DIR>          dev
10/10/2013 04:53 PM <DIR>          ext
10/10/2013 04:53 PM <DIR>          extras
08/30/2013 07:11 AM           417,792 fdftk.dll
08/30/2013 07:11 AM            90,112 fribidi.dll
08/30/2013 07:11 AM           346,624 gds32.dll
08/30/2013 07:11 AM                90 go-pear.bat
08/30/2013 07:11 AM            96,317 install.txt
08/30/2013 07:11 AM         1,097,728 libeay32.dll
08/30/2013 07:11 AM           166,912 libmcrypt.dll
08/30/2013 07:11 AM           165,643 libmhash.dll
08/30/2013 07:11 AM         2,035,712 libmysql.dll
08/30/2013 07:11 AM           385,024 libswish-e.dll
08/30/2013 07:11 AM             3,286 license.txt
08/30/2013 07:11 AM            57,344 msql.dll
08/30/2013 07:11 AM           168,858 news.txt
08/30/2013 07:11 AM           278,800 ntwdblib.dll
10/10/2013 04:53 PM <DIR>          PEAR
08/30/2013 07:11 AM            41,017 php-cgi.exe
08/30/2013 07:11 AM            32,825 php-win.exe
08/30/2013 07:11 AM            32,821 php.exe
08/30/2013 07:11 AM             2,523 php.gif
08/30/2013 07:11 AM            46,311 php.ini-dist
08/30/2013 07:11 AM            49,953 php.ini-recommended
08/30/2013 07:11 AM            36,924 php5apache.dll
08/30/2013 07:11 AM            36,925 php5apache2.dll
08/30/2013 07:11 AM            36,927 php5apache2_2.dll
08/30/2013 07:11 AM            36,932 php5apache2_filter.dll
08/30/2013 07:11 AM            57,410 php5apache_hooks.dll
```

```
08/30/2013 07:11 AM           669,318 php5embed.lib
08/30/2013 07:11 AM            28,731 php5isapi.dll
08/30/2013 07:11 AM            28,731 php5nsapi.dll
08/30/2013 07:11 AM         4,796,472 php5ts.dll
08/30/2013 07:11 AM            86,076 php_mysqli.dll
08/30/2013 07:11 AM               135 pws-php5cgi.reg
08/30/2013 07:11 AM               139 pws-php5isapi.reg
08/30/2013 07:11 AM             1,830 snapshot.txt
08/30/2013 07:11 AM           200,704 ssleay32.dll
              35 File(s) 11,569,880 bytes
              6 Dir(s) 180,664,549,376 bytes free
```

Operator backtick dinonaktifkan saat safe_mode diaktifkan. Pada beberapa sistem, safe_mode diatur ke Off secara default saat PHP diinstal. Pada sistem lain, safe_mode diatur ke On. Administrator sistem dapat mengubah nilai ini.

Menggunakan fungsi sistem

Fungsi sistem mengeksekusi perintah sistem, menampilkan output, dan mengembalikan baris terakhir output dari perintah sistem. Untuk menjalankan perintah sistem, gunakan pernyataan berikut:

```
$result = system("dir c:\php");
```

Ketika pernyataan ini dijalankan, daftar direktori ditampilkan, dan $result berisi baris terakhir yang dihasilkan dari perintah. Jika echo kita $result, kita melihat sesuatu seperti berikut:

```
11 Dir(s) 566,263,808 bytes free
```

Isi $result dengan fungsi sistem adalah baris terakhir dari output perintah dir.

Menggunakan fungsi exec

Fungsi exec menjalankan perintah sistem tetapi tidak menampilkan output. Sebaliknya, output dapat disimpan dalam array, dengan setiap baris output menjadi elemen dalam array. Baris terakhir dari output dikembalikan. Mungkin kita hanya ingin tahu berapa banyak file dan byte gratis dalam sebuah direktori. Dengan pernyataan berikut, kita menjalankan perintah tanpa menyimpan output dalam array:

```
$result = exec("dir c:\php");
```

Perintah dijalankan, tetapi outputnya tidak ditampilkan. Variabel $result berisi baris terakhir dari output. Jika kita echo $result, tampilannya akan terlihat seperti ini:

```
11 Dir(s) 566,263,808 bytes free
```

Outputnya adalah baris terakhir dari output perintah dir. Jika kita ingin menyimpan seluruh output dari perintah dir dalam sebuah array, gunakan perintah berikut:

```
$result = exec("dir c:\php",$dirout);
```

Setelah pernyataan ini, array $dirout berisi daftar direktori, dengan satu baris per item. Kita dapat menampilkan daftar direktori sebagai berikut:

```
foreach($dirout as $line)
{
    echo “$line\n”;
}
```

Loop menampilkan berikut ini:

```
Volume in drive C has no label.
  Volume Serial Number is 394E-15E5
  Directory of c:\php
10/10/2013 05:43 PM <DIR> .
10/10/2013 05:43 PM <DIR> ..
10/10/2013 04:53 PM <DIR> dev
10/10/2013 04:53 PM <DIR> ext
10/10/2013 04:53 PM <DIR> extras
08/30/2013 07:11 AM 417,792 fdftk.dl
```

Kita juga dapat menggunakan pernyataan berikut untuk mendapatkan elemen tertentu dari output array:

```
echo $dirout[3]; echo $dirout[7];
```

Outputnya adalah sebagai berikut:

```
Directory of C:\PHP
10/10/2013 04:53 PM <DIR> dev
```

Menggunakan fungsi passthru

Fungsi passthru mengeksekusi perintah sistem dan menampilkan output persis seperti yang dikembalikan. Untuk menjalankan perintah sistem, gunakan pernyataan berikut:

```
passthru(“dir c:\php”);
```

Pernyataan tersebut menampilkan daftar direktori tetapi tidak mengembalikan apa pun. Oleh karena itu, kita tidak menggunakan variabel untuk menyimpan data yang dikembalikan. Output ditampilkan dalam bentuk mentah; itu tidak diproses. Oleh karena itu, fungsi ini dapat digunakan ketika output biner diharapkan.

Mengakses pesan kesalahan dari perintah sistem

Metode untuk menjalankan perintah sistem tidak menampilkan atau mengembalikan pesan kesalahan informasi saat perintah sistem gagal. Kita tahu perintah sistem tidak berfungsi karena kita tidak mendapatkan hasil yang kita harapkan. Tetapi karena fungsinya tidak mengembalikan pesan kesalahan, kita tidak tahu apa yang salah.

Kita dapat mengembalikan atau menampilkan pesan kesalahan sistem operasi dengan menambahkan beberapa karakter tambahan ke perintah sistem yang kita jalankan. Pada sebagian besar sistem operasi, jika kita menambahkan karakter 2>&1 setelah perintah sistem, pesan kesalahan akan dikirim ke mana pun output diarahkan. Misalnya, kita dapat menggunakan pernyataan berikut:

```
$result = system(“di c:\php”);
```

Fungsi sistem menampilkan direktori ketika perintah sistem dijalankan. Namun, perhatikan bahwa dir salah ketik. Ini adalah di daripada dir. Tidak ada perintah sistem yang disebut di, sehingga perintah sistem tidak dapat dijalankan, dan tidak ada yang ditampilkan. Misalkan kita menggunakan pernyataan berikut sebagai gantinya:

```
$result = system("di c:\php 2>&1");
```

Dalam hal ini, pesan kesalahan ditampilkan. Di Windows, pesan kesalahan yang ditampilkan adalah sebagai berikut:

```
'di' is not recognized as an internal or external command,
operable program or batch file.
```

Pastikan kita tidak menyertakan spasi di 2>&1. Formatnya membutuhkan karakter bersama-sama, tanpa spasi.

4.4 MEMAHAMI MASALAH KEAMANAN

Saat kita menjalankan perintah sistem, kita mengizinkan pengguna untuk melakukan tindakan di komputer Anda. Jika perintah sistem adalah dir c:\php, tidak apa-apa. Namun, jika perintah sistem adalah rm /bin/* atau del c:*.*, kita tidak akan puas dengan hasilnya. Kita harus berhati-hati saat menggunakan fungsi yang menjalankan perintah sistem di luar skrip Anda.

Selama kita hanya menjalankan perintah yang kita tulis sendiri, seperti dir atau ls, kita baik-baik saja. Tetapi ketika kita mulai menjalankan perintah yang menyertakan data yang dikirim oleh pengguna, kita harus sangat berhati-hati. Misalnya, kita memiliki aplikasi di mana pengguna mengetikkan nama ke dalam formulir dan aplikasi kita kemudian membuat direktori dengan nama yang dikirim oleh pengguna. Pengguna mengetik Wibowo ke dalam bidang formulir bernama directoryName. Skrip kita yang memproses formulir memiliki perintah, sebagai berikut:

```
$directoryName = $_POST['directoryName'];
exec("mkdir $directoryName");
```

Karena $directoryName = Wibowo, mkdir Wibowo adalah perintah sistem yang dijalankan. Direktori dibuat, dan semua orang senang. Namun, misalkan pengguna mengetik Wiboo; rm * ke dalam formulir. Dalam hal ini, $directoryName =Wibowo;rm *. Perintah sistem yang dijalankan sekarang adalah mkdir Wibowo;rm *. Pada banyak sistem operasi, seperti Unix dan Linux, karakter titik koma memisahkan dua perintah sehingga dua perintah dapat dimasukkan dalam satu baris. Perintah dijalankan sebagai berikut:

```
mkdir Wibowo rm *
```

Sekarang kita punya masalah. Direktori Wibowo dibuat, dan semua file di direktori saat ini dihapus. Jika kita menggunakan variabel dalam perintah sistem, kita harus menggunakannya dengan hati-hati. Kita harus tahu dari mana asalnya. Jika berasal dari luar skrip, kita perlu memeriksa nilai dalam variabel sebelum menggunakannya. Pada contoh sebelumnya, kita dapat menambahkan kode sehingga skrip memeriksa variabel untuk memastikan variabel tersebut hanya berisi huruf dan angka sebelum menggunakannya dalam perintah mkdir.

4.5 MENGGUNAKAN FTP

Mentransfer file dari satu komputer ke komputer lain terjadi trilyun kali sehari di Internet. Ketika rekan-rekan di seberang negara perlu berbagi file, itu tidak masalah. Transfer cepat hanya membutuhkan beberapa detik, dan semua pihak memiliki file yang mereka butuhkan. File Transfer Protocol (FTP) adalah cara umum untuk mentransfer file dari satu komputer ke komputer lain. FTP memungkinkan kita mendapatkan daftar direktori dari komputer lain atau mengunduh atau mengunggah satu file atau beberapa file sekaligus. FTP adalah perangkat lunak klien/server. Untuk menggunakan FTP untuk mentransfer file antara komputer kita dan komputer jarak jauh, kita terhubung ke server FTP di komputer jarak jauh dan mengirimkannya permintaan.

Perlu dicatat bahwa FTP secara inheren tidak aman, dan tidak dengan cara terapi akan membantu. Saat kita menggunakan FTP, nama pengguna, kata sandi, dan file itu sendiri akan dilewatkan melalui jaringan tanpa enkripsi. Ini berarti bahwa seseorang dengan pengetahuan dan akses yang cukup ke jaringan kita dapat "mengendus" nama pengguna dan kata sandi. Jika kita mencari metode yang lebih aman untuk mentransfer file, lihat perintah SCP atau SFTP. Konon, FTP masih digunakan secara luas, terutama untuk penyedia hosting.

Untuk menggunakan FTP dalam skrip Anda, dukungan FTP harus diaktifkan saat PHP diinstal. Jika kita menginstal PHP untuk Windows, kita tidak perlu melakukan apa pun ekstra untuk mengaktifkan dukungan FTP. Jika kita mengkompilasi PHP di Unix, Linux, atau Mac dan kita ingin mengaktifkan dukungan FTP, kita dapat menggunakan opsi instalasi dukungan FTP, sebagai berikut:

--enable-ftp

Di bagian ini, kita memberi tahu kita apa yang perlu kita ketahui tentang masuk ke server FTP Anda, mengakses daftar direktori, mentransfer file ke dan dari server FTP, dan menggunakan berbagai fungsi untuk menyelesaikan tugas terkait FTP.

Masuk ke server FTP

Untuk terhubung ke server FTP di komputer yang ingin kita gunakan untuk bertukar file, gunakan fungsi ftp_connect, sebagai berikut:

$connect = ftp_connect*(“janet.valade.com”);*

Atau, kita dapat terhubung dengan menggunakan alamat IP, sebagai berikut:

$connect = ftp_connect*(“172.17.204.2”);*

Setelah kita terhubung, kita harus masuk ke server FTP. Kita memerlukan ID pengguna dan kata sandi untuk masuk. Kita mungkin memiliki ID dan kata sandi pribadi kita sendiri, atau kita mungkin menggunakan ID umum dan kata sandi yang dapat digunakan siapa saja. Beberapa situs publik di Internet mengizinkan siapa pun masuk dengan menggunakan ID pengguna anonim dan alamat email pengguna sebagai kata sandi. Untuk keamanan, sebaiknya masukkan ID pengguna dan kata sandi ke dalam file terpisah dan sertakan file tersebut bila diperlukan.

Fungsi ftp_login memungkinkan kita untuk masuk ke server FTP setelah kita membuat koneksi. Pernyataan ini mengasumsikan kita memiliki ID akun dan kata sandi yang disimpan dalam variabel, sebagai berikut:

$login_result = ftp_login($connect,$userid,$passwd);

Jika kita mencoba masuk tanpa membuat koneksi ke server FTP terlebih dahulu, kita akan melihat peringatan berikut:

Warning: ftp_login() expects parameter 1 to be resource, boolean given in **d:\test1\test13.php** on line **9**

Peringatan tidak menghentikan program. Login gagal, tetapi skrip berlanjut, yang mungkin bukan yang kita inginkan. Karena sisa skrip kita mungkin bergantung pada koneksi FTP yang berhasil, kita mungkin ingin menghentikan skrip jika fungsinya gagal. Pernyataan berikut menghentikan skrip jika fungsi gagal:

```
$connect = ftp_connect("janet.valade.com")
 or die("Can't connect to server");
$login_result = ftp_login($connect,$userid,$passwd)
 or die("Can't login to server");
```

Setelah kita masuk ke server FTP, kita dapat mengirimkannya permintaan untuk menyelesaikan tugas, seperti mendapatkan daftar direktori atau mengunggah dan mengunduh file, seperti yang dijelaskan di bagian berikut

Mendapatkan daftar direktori

Salah satu tugas umum adalah mendapatkan daftar direktori. Pernyataan ftp_nlist mendapatkan daftar direktori dari komputer jarak jauh dan menyimpannya dalam array, sebagai berikut:

```
$filesArr = ftp_nlist($connect,"data");
```

Parameter kedua dalam tanda kurung adalah nama direktori. Jika kita tidak mengetahui nama direktori, kita dapat meminta server FTP untuk mengirimkan nama direktori saat ini kepada Kita, sebagai berikut:

```
$directory_name = ftp_pwd($connect);
$filesArr = ftp_nlist($connect,$directory_name);
```

Daftar direktori yang dikirim FTP setelah pernyataan ftp_nlist berjalan disimpan dalam array, satu nama file di setiap elemen array. Kita kemudian dapat menampilkan daftar direktori dari array, sebagai berikut:

```
foreach($filesArr as $value)
{
 echo "$value\n";
}
```

Mengunduh dan mengunggah file dengan FTP

Kita dapat mengunduh file dari komputer jarak jauh dengan fungsi ftp_get. Pernyataan berikut mengunduh file dari komputer jarak jauh setelah kita masuk ke server FTP:

```
ftp_get($connect,"newfile.txt","data.txt",FTP_ASCII);
```

Nama file pertama, *newfile.txt,* adalah nama file yang akan ada di komputer kita setelah diunduh. Nama file kedua, data.txt, adalah nama file yang ingin kita unduh. Istilah FTP_ASCII dalam pernyataan memberi tahu FTP jenis file apa yang sedang diunduh. Berikut adalah pilihan untuk mode file:

- FTP_ASCII: Ini adalah file teks.
- FTP_BINARY: File bahasa mesin, pada dasarnya apa pun yang bukan teks biasa.

Kita dapat menentukan mode file mana yang kita butuhkan dengan memeriksa isi file. Jika isinya adalah karakter yang dapat kita baca dan pahami, file tersebut adalah ASCII. Jika isinya tampak seperti sampah, file tersebut adalah biner. File grafis, misalnya, adalah biner.
Kita dapat mengunggah file dengan fungsi serupa yang disebut ftp_put. Pernyataan berikut mengunggah file:

```
ftp_put($connect,”newfile.txt”,”data.txt”,FTP_ASCII);
```

Nama file pertama, newfile.txt, adalah nama file yang akan ada di komputer jarak jauh setelah diunggah. Nama file kedua, data.txt, adalah nama file yang ingin kita unggah. Ketika kita selesai mentransfer file melalui koneksi FTP Kita, kita dapat menutup koneksi dengan pernyataan berikut:

```
ftp_close($connect);
```

Script di bawah ini mendownload semua file dalam direktori yang memiliki ekstensi .txt. File diunduh dari komputer jarak jauh melalui koneksi FTP.
Skrip untuk Mengunduh File melalui FT:

```
<?php
 /* Script name: downloadFiles
 * Description: Downloads all the files with a .txt
 * extension in a directory via FTP.
 */
include(“ftpstuff.inc”);
 $dir_name = “data/”;
 $connect = ftp_connect($servername)
 or die(“Can’t connect to FTP server”);
 $login_result = ftp_login($connect,$userID,$passwd)
 or die(“Can’t log in”);
 $filesArr = ftp_nlist($connect,$dir_name);
 foreach($filesArr as $value)
 {
 if(preg_match(“#\.txt$#”,$value))
 {
 if(!file_exists($value))
 {
 ftp_get($connect,$value,$dir_name.$value,FTP_ASCII);
 }
 else
 {
 echo “File $value already exists!\n”;
 }
```

```
 }
 }
 ftp_close($connect);
?>
```

Script mendapatkan daftar direktori dari komputer jarak jauh dan menyimpannya di $filesArr. Pernyataan foreach loop melalui nama file di $filesArr dan memeriksa untuk melihat apakah setiap file memiliki ekstensi .txt. Saat file memiliki ekstensi .txt, skrip akan menguji untuk melihat apakah file dengan nama yang sama sudah ada di komputer lokal. Jika file dengan nama itu belum ada, file tersebut diunduh; jika file seperti itu memang ada, pesan akan diprint, dan file tidak diunduh.

Script di atas menyertakan file bernama ftpstuff.inc. File ini berisi informasi yang diperlukan untuk terhubung ke server dengan FTP. File ftpstuff.inc berisi kode yang mirip dengan berikut ini:

```
<?php
 $servername = “yourserver”;
 $userID = “youruserid”;
 $passwd = “yourpassword”;
?>
```

Melihat fungsi FTP lainnya

Fungsi FTP tambahan melakukan tindakan lain, seperti mengubah ke direktori lain di komputer jarak jauh atau membuat direktori baru di komputer jarak jauh. Tabel - berisi sebagian besar fungsi FTP yang tersedia.

Tabel 4.2 Fungsi FTP

Fungsi	Apa yang dilakukan
ftp_cdup($connect)	Ubah direktori langsung dibawah direktori saat ini.
ftp_chdir($connect, ”*directoryname*”)	Ubah direktori pada komputer jarak jauh
ftp_close($connect)	Tutup koneksi FTP
ftp_connect(“servername”)	Bukan koneksi komputer. *Servername* dapat menjadi ama domain atau sebuah IP adress
ftp_delete($connect,”path/ filename”)	Hapus file pada komputer jarak jauh
ftp_exec ($connect,”command”)	Ekseskusi perintah sistem pada komputer jarak jauh
ftp_ fget($connect,$fh,”data. txt”,FTP_ASCII)	Download sebuah file yang dibuka untuk mengontrol komputer. $fh adalah file yang dihandle oleh file yang terbuka
ftp_fput($connect,”new. txt”,$fh,FTP_ASCII)	Upload sebuah file yang dibuka untuk mengontrol komputer. $fh adalah file yang dihandle oleh file yang terbuka
ftp_get($connect,”d. txt”,”sr.txt”,FTP_ASCII)	Mendownload file melalui komputer jarak jauh. Sr.txt adalah nama file yang akan didownload, dan d.txt adalah nama file yang sudah di download
ftp_login($connect,$userID ,$password)	Log in ke server FTP

ftp_mdtm($connect, ”filename.txt”)	Dapatkan waktu pada file terakhir kali di modifikasi
ftp_mkdir($connect, ”directoryname”)	Membuat direktori baru pada komputer jarak jauh
ftp_nlist($connect, ”directoryname”)	Mendapatkan daftar file yang terkontrol pada direktori. File akan di kembalikan pada array
ftp_put($connect,”d. txt”,”sr.txt”,FTP_ASCII)	Upload sebuah file untuk mengontrol komputer. Sr.txt adalah nama file yang akan didownload, dan d.txt adalah nama file yang sudah di download
ftp_pwd($connect)	Dapatkan nama direktori saat ini pada kontrol komputer
ftp_rename($connect,”oldna me”,”newname”)	Ubah nama file pada komputer jarak jauh
ftp_rmdir($connect, ”directoryname”)	Hapus direktori pada komputer jarak jauh
ftp_ size($connect,”filename. txt”)	Mengembalikan ukuran file pada komputer jarak jauh
ftp_systype($connect)	Mengembalikan jenis sistem pada file server komputer jarak jauh (misalnya, Unix)

Membaca dan Menulis File

Buku ini berisi informasi tentang penggunaan PHP dan MySQL secara bersamaan. Di sebagian besar aplikasi, kita menyimpan data yang dibutuhkan oleh aplikasi dalam database MySQL. Namun, terkadang kita perlu membaca atau menulis informasi dalam file teks yang bukan merupakan database. Bagian ini menjelaskan cara membaca dan menulis data dalam file teks, juga disebut file datar.

Kita menggunakan pernyataan PHP untuk membaca dari atau menulis ke file datar.Menggunakan file datar memerlukan tiga langkah:

1. Buka filenya.
2. Menulis data ke dalam file atau mengambil data dari file tersebut.
3. Tutup file.

Langkah-langkah ini dibahas secara rinci di bagian berikut:

Mengakses file

Langkah pertama, sebelum kita dapat menulis informasi ke dalam atau membaca informasi dari sebuah file, adalah membuka file tersebut. Berikut ini adalah format umum untuk pernyataan yang membuka file:

$fh = fopen(“filename”,”mode”)

Variabel, $fh, disebut sebagai pegangan file, digunakan dalam pernyataan yang menulis data ke atau membaca data dari file yang terbuka sehingga PHP mengetahui file mana yang akan ditulis atau dibaca. Variabel $fh berisi informasi yang mengidentifikasi lokasi file yang terbuka. Kita menggunakan mode saat membuka file untuk memberi tahu PHP apa yang ingin kita lakukan dengan file tersebut. Tabel dibawah ini menunjukkan mode yang dapat kita gunakan.

Tabel 4.3 Mode yang dapat digunakan dalam PHP

Mode	Apa yang dilakukan	Apa yang terjadi jika file tidak ada
r	*Read only*	Pesan peringatan tidka di tampilkan

r+	*Reading and writing*	Pesan peringatan tidka di tampilkan
w	*Write only*	PHP mencoba untuk membuat ini.(jika file ada, maka PHP akan menimpanya)
w+	*Reading and writing*	PHP mencoba untuk membuat ini.(jika file ada, maka PHP akan menimpanya)
a	*Append data at the end of gile*	PHP mencooba membuat ini
a+	*Reading and appending*	PHP mencooba membuat ini

Nama file dapat berupa nama file sederhana (namafile.txt), jalur ke file (c:/data/namafile.txt), atau URL (http://situsKita.com/namafile.txt).

Membuka file dalam mode baca

Kita dapat membuka file file1.txt untuk membaca informasi dalam file dengan pernyataan berikut:

```
$fh = fopen("file1.txt","r");
```

Berdasarkan pernyataan ini, PHP mencari file1.txt di direktori saat ini, yang merupakan direktori tempat skrip PHP kita berada. Jika file tidak dapat ditemukan, pesan peringatan, mirip dengan berikut ini, mungkin atau mungkin tidak ditampilkan, tergantung pada set tingkat kesalahan:

Warning: fopen(file1.txt): failed to open stream: No such file or directory in **d:\test2.php** on line **15**

Ingat, kondisi peringatan tidak menghentikan skrip. Skrip terus berjalan, tetapi file tidak terbuka, sehingga pernyataan selanjutnya yang membaca atau menulis ke file tidak akan dieksekusi. Kita mungkin ingin skrip berhenti jika file tidak dapat dibuka. Kita perlu melakukan ini sendiri dengan pernyataan mati, sebagai berikut:

```
$fh = fopen("file1.txt","r")
                or die("Can't open file");
```

Pernyataan die menghentikan skrip dan menampilkan pesan yang ditentukan.

Membuka file dalam mode write

Kita dapat membuka file dalam direktori tertentu untuk menyimpan informasi dengan menggunakan jenis pernyataan berikut:

```
$fh = fopen("c:/testdir/file1.txt","w");
```

Jika file tidak ada, itu dibuat di direktori yang ditunjukkan. Namun, jika direktori tidak ada, direktori tidak dibuat, dan peringatan akan ditampilkan. (Kita harus membuat direktori sebelum mencoba menulis file ke dalam direktori.) kita dapat memeriksa apakah ada direktori sebelum mencoba menulis file ke dalamnya dengan menggunakan pernyataan berikut:

```
If(is_dir("c:/tester"))
{
 $fh = fopen("c:/testdir/file1.txt","w");
}
```

Dengan pernyataan ini, pernyataan fopen dijalankan hanya jika path/nama file ada dan merupakan direktori.

Membuka file di situs web lain

Kita juga dapat membuka file di situs web lain dengan menggunakan pernyataan seperti berikut:

```
$fh = fopen("http://janet.valade.com/index.html","r");
```

Kita dapat menggunakan URL hanya dengan mode baca, bukan dengan mode tulis, dan ada cara yang lebih baik untuk melakukannya — yaitu, fungsi cURL. Lihat manual PHP di http://php.net/manual/en/book.curl.php untuk informasi lebih lanjut tentang fungsi cURL.

Menutup file

Untuk menutup file setelah kita selesai membaca atau menulisnya, gunakan pernyataan berikut:

```
fclose($fh);
```

Dalam pernyataan ini, $fh adalah variabel penanganan file yang kita buat saat kita membuka file.

Menulis ke file

Setelah kita membuka file, kita dapat menulis ke dalamnya dengan menggunakan pernyataan fwrite, yang memiliki format umum berikut:

```
fwrite($fh,datatosave);
```

Dalam pernyataan ini, $fh adalah pegangan file yang kita buat ketika kita membuka file yang berisi pointer ke file yang terbuka, dan datatosave adalah informasi yang akan disimpan dalam file. Informasi dapat berupa string atau variabel. Misalnya, kita dapat menggunakan pernyataan berikut:

```
$today = date("Y-m-d");
$fh = fopen("file2.txt","a");
fwrite($fh,"$today\n");
fclose($fh);
```

Pernyataan ini menyimpan tanggal saat ini dalam file bernama file2.txt. Perhatikan bahwa file dibuka dalam mode append (a). Jika file tidak ada, itu dibuat, dan tanggal ditulis sebagai baris pertama. Jika file ada, tanggal ditambahkan ke akhir file. Dengan cara ini, kita membuat file log yang menyimpan daftar tanggal skrip dijalankan. Pernyataan fwrite menyimpan persis apa yang kita kirim. Setelah pernyataan fwrite dijalankan dua kali, file2.txt berisi:

```
2013-10-22
2013-10-22
```

Tanggal muncul pada baris terpisah karena karakter baris baru (\n) ditulis ke file. Pastikan untuk membuka file dengan mode a jika kita ingin menambahkan informasi ke file. Jika kita menggunakan mode tulis, file akan ditimpa setiap kali dibuka.

Membaca dari file

Kita dapat membaca dari file dengan menggunakan pernyataan fgets, yang memiliki format umum berikut:

```
$line = fgets($fh)
```

Dalam pernyataan ini, $fh menahan pointer ke file yang terbuka. Pernyataan ini membaca string hingga menemukan akhir baris atau akhir file, mana saja yang lebih dulu, dan menyimpan string dalam $line. Untuk membaca seluruh file, kita terus membaca baris sampai kita mencapai akhir file. PHP mengenali akhir file dan menyediakan fungsi feof untuk memberi tahu kita saat kita mencapai akhir file. Pernyataan berikut membaca dan menampilkan semua baris dalam file:

```
while(!feof($fh))
{
 $line = fgets($fh);
 echo “$line”;
}
```

Di baris pertama, feof($fh) mengembalikan nilai true ketika akhir file tercapai. Tanda seru meniadakan kondisi yang sedang diuji, sehingga pernyataan while terus berjalan selama akhir file tidak tercapai. Ketika akhir file tercapai, while berhenti. Jika kita menggunakan pernyataan ini untuk membaca file log yang dibuat di bagian sebelumnya, kita mendapatkan output berikut:

```
2013-10-22 2013-10-22
```

Seperti yang kita lihat, karakter baris baru disertakan saat baris dibaca. Dalam beberapa kasus, kita tidak ingin akhir baris disertakan. Jika demikian, kita perlu menghapusnya dengan menggunakan pernyataan berikut:

```
while(!feof($fh))
{
 $line = rtrim(fgets($fh));
 echo “$line”;
}
```

Fungsi rtrim menghapus spasi kosong yang tertinggal dan karakter baris baru. Output dari pernyataan-pernyataan tersebut adalah sebagai berikut:

```
2013-10-222013-10-22
```

Membaca file sepotong demi sepotong

Terkadang kita ingin membaca string dengan ukuran tertentu dari file. Kita dapat memberitahu fgets untuk membaca sejumlah karakter dengan menggunakan format berikut:

```
$line = fgets($fh,n)
```

Pernyataan ini memberitahu PHP untuk membaca string yang panjangnya n-1 karakter hingga mencapai akhir baris atau akhir file. Misalnya, kita dapat menggunakan pernyataan berikut:

```
while(!feof($fh))
{
 $char4 = fgets($fh,5);
 echo "$char4\n";
}
```

Pernyataan ini membaca setiap string empat karakter hingga akhir file. Outputnya adalah sebagai berikut:

```
2013
-10-
22
2013
-10-
22
```

Perhatikan bahwa ada baris baru di akhir setiap baris file.

Membaca file ke dalam array

Seringkali berguna untuk memiliki seluruh file dalam sebuah array. Kita dapat melakukannya dengan pernyataan berikut:

```
$fh = fopen("file2.txt","r");
while(!feof($fh))
{
 $content[] = fgets($fh);
}
fclose($fh);
```

Hasilnya adalah array $content dengan setiap baris file sebagai elemen array. Kunci array adalah angka. PHP menyediakan fungsi shortcut untuk membuka file dan membaca seluruh isi ke dalam array, satu baris di setiap elemen array. Pernyataan berikut menghasilkan hasil yang sama seperti lima baris sebelumnya:

```
$content = file("file2.txt");
```

Pernyataan membuka file2.txt, menempatkan setiap baris ke dalam elemen array $content, dan kemudian menutup file. Fungsi file dapat memperlambat skrip kita jika file yang kita buka sangat besar. Seberapa besar tergantung pada jumlah memori komputer yang tersedia. Jika skrip kita tampak lambat, coba baca file dengan fgets daripada file dan lihat apakah itu mempercepat skrip. Kita dapat mengarahkan fungsi file untuk membuka file secara otomatis di direktori include kita dengan menggunakan pernyataan berikut:

```
Content = file("file2.txt",1);
```

1 memberitahu PHP untuk mencari file2.txt di direktori include daripada di direktori saat ini.

Membaca file menjadi string

Terkadang menempatkan seluruh isi file ke dalam satu string panjang bisa bermanfaat. Misalnya, kita mungkin ingin mengirim konten file dalam pesan email. PHP menyediakan fungsi untuk membaca file menjadi string, sebagai berikut:

$content = file_get_contents("file2.txt",1);

Fungsi file_get_contents bekerja sama dengan fungsi file, kecuali bahwa ia menempatkan seluruh isi file ke dalam string daripada array. Setelah pernyataan ini, kita dapat menggemakan $content sebagai berikut:

echo $content;

Outputnya adalah sebagai berikut:

2013-10-22
2013-10-22

Output muncul pada baris terpisah karena karakter akhir baris dibaca dan disimpan sebagai bagian dari string. Jadi, saat kita menggemakan string, kita juga menggemakan karakter akhir baris, yang memulai baris baru. Fungsi file_get_contents diperkenalkan di versi 4.3.0. Ini tidak tersedia di versi PHP yang lebih lama.

Bertukar Data dengan Program Lain

Kita mungkin terkadang perlu memberikan informasi ke program lain atau membaca informasi ke dalam PHP dari program lain. File datar sangat berguna untuk tugas seperti itu, dan kita menjelaskan cara melakukan tugas semacam itu di sini.

Bertukar data dalam file datar

Hampir semua perangkat lunak memiliki kemampuan untuk membaca informasi dari file datar atau menulis informasi ke dalam file datar. Misalnya, secara default, pengolah kata kita menyimpan dokumen kita dalam formatnya sendiri, yang hanya dapat dipahami oleh pengolah kata. Namun, kita dapat memilih untuk menyimpan dokumen dalam format teks. Dokumen teks adalah file datar berisi teks yang dapat dibaca oleh perangkat lunak lain. Pengolah kata kita juga dapat membaca file teks, bahkan yang ditulis oleh perangkat lunak lain.

Saat skrip PHP kita menyimpan informasi ke dalam file teks, informasi tersebut dapat dibaca oleh perangkat lunak apa pun yang memiliki kemampuan untuk membaca file teks. Misalnya, file teks dapat dibaca oleh sebagian besar perangkat lunak pengolah kata. Namun, beberapa perangkat lunak memerlukan format tertentu dalam file teks. Misalnya, aplikasi perangkat lunak buku alamat mungkin membaca data dari file datar tetapi memerlukan informasi berada di lokasi tertentu — misalnya, 20 karakter pertama dalam satu baris dibaca sebagai nama, 20 karakter berikutnya dibaca sebagai jalan. alamat, dan sebagainya. Kita perlu tahu format apa yang dibutuhkan perangkat lunak dalam file datar. Kemudian tulis file datar dalam format yang benar dalam skrip PHP kita dengan menggunakan pernyataan fwrite.

Bertukar data dalam format yang dipisahkan koma

File nilai yang dipisahkan koma (CSV) — juga disebut file yang dipisahkan koma — adalah format umum yang digunakan untuk mentransfer informasi antar program perangkat lunak.

Memahami format yang dipisahkan koma

File CSV digunakan untuk mentransfer informasi yang dapat disusun sebagai tabel, diatur sebagai baris dan kolom. Misalnya, program spreadsheet mengatur data sebagai baris dan kolom dan dapat membaca dan menulis file CSV. File CSV juga sering digunakan untuk mentransfer data antara perangkat lunak database yang berbeda, seperti antara MySQL dan Microsoft Access. Banyak program perangkat lunak lain dapat membaca dan menulis data dalam file CSV. File CSV diatur dengan setiap baris tabel pada baris terpisah dalam file, dan

kolom dalam baris dipisahkan dengan koma. Misalnya, buku alamat dapat diatur sebagai file CSV, sebagai berikut:

```
Joseph Teguh,1234 Jl. Pedurungan,Kota Semarang,ID,99999
Agus Wibowo,5678 Jl. Ungaran,Kota Semarang,ID,11111
Mars Caroline,1234 Jalan Majapahit,Kota Semarang,ID,88888
```

Excel dapat membaca file ini menjadi tabel dengan lima kolom. Tanda koma menKitakan akhir satu kolom dan awal berikutnya.

Membuat file yang dipisahkan koma

Pernyataan PHP berikut membuat file CSV:

```
$address[] = “Joseph Teguh,1234 Jl. Pedurungan,Kota Semarang,ID,99999”;
$address[] = “Agus Wibowo,5678 Jl. Ungaran,Kota Semarang,ID,11111”;
$address[] = “Mars Caroline,1234 Jalan Majapahit,Kota Semarang,ID,88888”;
$fh = fopen(“addressbook.txt”,”a”);
for ($i=0;$i<3;$i++)
{
 fwrite($fh,$address[$i].”\n”);
}
fclose($fh);
```

Membaca file yang dipisahkan koma

PHP dapat membaca file CSV dengan menggunakan file atau fungsi fgets. Namun, PHP menyediakan fungsi yang disebut fgetcsv yang dirancang khusus untuk membaca file CSV. Saat kita menggunakan fungsi ini untuk membaca baris dalam file CSV, baris disimpan dalam larik, dengan setiap entri kolom dalam elemen larik. Misalnya, kita dapat menggunakan fungsi untuk membaca baris pertama file CSV buku alamat, seperti yang ditunjukkan di sini:

```
$address = fgetcsv($fh,1000);
```

Dalam pernyataan ini, $fh adalah pegangan file, dan 1000 adalah jumlah karakter yang harus dibaca. Untuk membaca seluruh baris, gunakan sejumlah karakter yang lebih panjang dari baris terpanjang. Hasil dari pernyataan ini adalah array, sebagai berikut:

```
$address[0] = Joseph Teguh
$address[1] = 1234 Jl. Pedurungan
$address[2] = Kota Semarang
$address[3] = ID
$address[4] = 99999
```

Menggunakan pembatas lainnya

File CSV berfungsi dengan baik untuk mentransfer data dalam banyak kasus. Namun, jika koma adalah bagian dari data, koma tidak dapat digunakan untuk memisahkan kolom. Misalnya, salah satu baris data adalah ini:

```
Universitas STEKOM, 12234 Pedurungan,Kota Semarang,ID,99999
```

Koma dalam nama perusahaan akan membagi data menjadi dua membuat enam kolom, bukan lima.

Saat data berisi koma, kita dapat menggunakan karakter yang berbeda untuk memisahkan kolom. Misalnya, tab biasanya digunakan untuk memisahkan kolom. File ini disebut file tab-separated values (TSV), atau file tab-delimited. Kita dapat menulis file tab-delimited dengan menyimpan "\t" daripada koma di file output.
Kita dapat membaca file yang berisi tab dengan menentukan pemisah kolom dalam pernyataan, sebagai berikut:

```
$address = fgetcsv($fh,1000,"\t");
```

Kita dapat menggunakan karakter apa pun untuk memisahkan kolom. Script di dibawah ini erisi fungsi yang mengubah file CSV menjadi file tab-delimited. Skrip yang Mengonversi File CSV menjadi File Berbatas Tab:

```
<?php
    /* Script name: Convert
    * Description: Reads in a CSV file and outputs a
    * tab-delimited file. The CSV file must have a
    * .CSV extension.
    */
    $myfile = "testing";                                    →7
    function convert($filename)                             →8
    {
        if( @$fh_in = fopen("{$filename}.csv","r"))         →10
        {
        $fh_out = fopen("{$filename}.tsv","a");             →12
        while( !feof($fh_in))                               →13
        {
        $line = fgetcsv($fh_in,1024);                       →15
        if( $line[0] == "")                                 →16
        {
        fwrite($fh_out,"\n");
        }
        else {                                              →20
        fwrite($fh_out,implode($line,"\t")."\n");           →21
        }
    }
    fclose($fh_in);
    fclose($fh_out);
}
else {                                                      →27
      echo "File doesn't exist\n";
      return false;
}
echo "Conversion completed!\n";
return true;                                                →32
}
```

```
convert($myfile);                                    →34
?>
```

→7 Baris ini mendefinisikan nama file sebagai pengujian.
→8 Baris ini mendefinisikan fungsi bernama convert() dengan satu parameter, $filenama.
→10 Baris ini membuka file yang memiliki nama file yang diteruskan ke fungsi dengan ekstensi .csv. File dibuka dalam mode baca. Jika file berhasil dibuka, pernyataan konversi di blok if akan dieksekusi. Jika file tidak ditemukan, blok else yang dimulai pada Baris 27 akan dieksekusi.
→12 Baris ini membuka file yang memiliki nama file yang diteruskan ke fungsi dengan ekstensi .tsv. File dibuka dalam mode tambahkan. File ada di direktori saat ini dalam skrip ini. Jika file berada di direktori lain di mana menurut kita ada kemungkinan file tersebut mungkin tidak terbuka dalam mode tulis, gunakan pernyataan if di sini untuk menguji di mana file dibuka dan lakukan beberapa tindakan jika tidak.
→13 Baris ini memulai while loop yang berlanjut ke akhir file.
→15 Pernyataan ini membaca satu baris dari file input ke dalam array $line. Setiap entri kolom disimpan dalam elemen array.
→16 Pernyataan ini menguji apakah baris dari file input memiliki teks di dalamnya. Jika baris tidak memiliki teks apa pun, karakter baris baru disimpan dalam file output. Dengan demikian, setiap baris kosong dalam file input disimpan dalam file output.
→20 Jika baris dari file input tidak kosong, maka akan dikonversi ke format tabdelimited dan ditulis ke file output.
→21 Pernyataan ini mengubah baris dan menulisnya ke file output dalam satu pernyataan. Fungsi implode mengubah array $line menjadi string, dengan elemen yang dipisahkan oleh tab.
→27 Blok lain ini dijalankan ketika file input tidak dapat ditemukan. Pesan kesalahan digaungkan, dan fungsi mengembalikan false.
→32 Fungsi telah berhasil diselesaikan, sehingga mengembalikan nilai true.
→34 Baris ini memanggil fungsi, meneruskan nama file ke fungsi dalam variabel $myfile.

4.6 KOMPATIBILITAS VERSI PHP

PHP sedang dalam proses pengembangan yang berkelanjutan, dan ada beberapa versi. Jika kita perlu memeriksa apakah fungsi tertentu tersedia untuk kode Kita, kita dapat menggunakan fungsi function_exists, yang memeriksa semua fungsi yang telah ditentukan dan dibuat pengguna. Contoh dibawah ini memeriksa fungsi array_combine, yang khusus untuk PHP versi 5.

Contoh Memeriksa keberadaan suatu fungsi

```
<?php
    if (function_exists("array_combine"))
    {
        echo "Function exists";
    }
    else
    {
        echo "Function does not exist - better write our own";
    }
?>
```

Dengan menggunakan kode seperti ini, kita dapat memanfaatkan fitur-fitur di versi PHP yang lebih baru namun kode kita tetap berjalan di versi sebelumnya, selama kita mereplikasi fitur apa pun yang hilang. Fungsi kita mungkin lebih lambat daripada yang ada di dalamnya, tetapi setidaknya kode kita akan jauh lebih portabel. Kita juga dapat menggunakan fungsi phpversion untuk menentukan versi PHP mana yang menjalankan kode Kita. Hasil yang dikembalikan akan mirip dengan yang berikut, tergantung pada versi:

5.4.21

Terminologi

Saat membuat program untuk menggunakan objek, kita perlu mendesain gabungan data dan kode yang disebut kelas. Setiap objek baru berdasarkan kelas ini disebut instance (atau kejadian) dari kelas itu. Data yang terkait dengan suatu objek disebut propertinya; fungsi yang digunakannya disebut metode. Dalam mendefinisikan kelas, kita memberikan nama propertinya dan kode untuk metodenya. Ada juga slot untuk memasukkan koin (metode yang digunakan untuk mengaktifkan objek), dan pembaca cakram laser (metode yang digunakan untuk mengambil musik, atau properti, dari CD). Saat kita membuat objek, yang terbaik adalah menggunakan enkapsulasi, atau menulis kelas sedemikian rupa sehingga hanya metodenya yang dapat digunakan untuk memanipulasi propertinya. Dengan kata lain, kita menolak akses langsung kode luar ke datanya. Metode yang kita berikan dikenal sebagai antarmuka objek.

Pendekatan ini membuat proses debug menjadi mudah: kita harus memperbaiki kode yang salah hanya di dalam kelas. Selain itu, ketika kita ingin memutakhirkan program, jika kita telah menggunakan enkapsulasi yang tepat dan memelihara antarmuka yang sama, kita cukup mengembangkan kelas pengganti baru, men-debug mereka sepenuhnya, dan kemudian menukarnya dengan yang lama. Jika tidak berfungsi, kita dapat menukar yang lama kembali untuk segera memperbaiki masalah sebelum men-debug kelas baru lebih lanjut. Setelah kita membuat kelas, kita mungkin menemukan bahwa kita memerlukan kelas lain yang mirip dengannya tetapi tidak persis sama. Hal yang cepat dan mudah dilakukan adalah mendefinisikan kelas baru menggunakan pewarisan. Saat kita melakukan ini, kelas baru kita memiliki semua properti yang diwarisinya. Kelas asli sekarang disebut superclass, dan yang baru adalah subclass (atau kelas turunan).

Gambar 4.1 Jukebox: contoh yang bagus dari objek mandiri

Dalam contoh jukebox kita, jika kita menemukan jukebox baru yang dapat memutar video bersama musik, kita dapat mewarisi semua properti dan metode dari superclass jukebox asli dan menambahkan beberapa properti baru (video) dan metode baru (pemutar film) . Manfaat luar biasa dari sistem ini adalah jika kita meningkatkan kecepatan atau aspek lain dari superclass, subclass-nya akan menerima manfaat yang sama.

4.7 PHP 5 DESTRUKTOR

Juga baru di PHP 5 adalah kemampuan untuk membuat metode destruktor. Kemampuan ini berguna ketika kode telah membuat referensi terakhir ke suatu objek atau ketika skrip mencapai akhir. Contoh dibawah ini menunjukkan cara membuat metode destruktor.

Contoh Membuat metode destruktor di PHP 5

```
<?php
    class User
    {
        function __destruct()
        {
            // Destructor code goes here
        }
    }
?>
```

Metode Penulisan

Seperti yang telah kita lihat, mendeklarasikan metode mirip dengan mendeklarasikan fungsi, tetapi ada beberapa perbedaan. Misalnya, nama metode yang dimulai dengan garis bawah ganda (__) dicadangkan dan kita tidak boleh membuat formulir ini. Kita juga memiliki akses ke variabel khusus yang disebut $this, yang dapat digunakan untuk mengakses properti objek saat ini. Untuk melihat cara kerjanya, lihat Contoh dibawah, yang berisi metode berbeda dari definisi kelas Pengguna yang disebut get_password.

Contoh Menggunakan variabel $this dalam sebuah metode

```
<?php
    class User
    {
    public $name, $password;
        function get_password()
        {
              return $this->password;
        }
    }
?>
```

get_password menggunakan variabel $this untuk mengakses objek saat ini dan kemudian mengembalikan nilai properti kata sandi objek itu. Perhatikan bagaimana $ sebelumnya dari properti $password dihilangkan ketika kita menggunakan operator ->. Membiarkan $ di tempat adalah kesalahan umum yang mungkin kita alami, terutama ketika kita pertama kali menggunakan fitur ini. Inilah cara kita menggunakan kelas yang didefinisikan dalam Contoh diatas:

```
$object = new User;
$object->password = "secret";
echo $object->get_password();
```

Kode ini mencetak rahasia kata sandi.

Metode Statis di PHP 5

Jika kita menggunakan PHP 5, kita juga dapat mendefinisikan metode sebagai statis, yang berarti metode tersebut dipanggil pada kelas, bukan pada objek. Metode statis tidak memiliki akses ke properti objek apa pun dan dibuat serta diakses seperti pada contoh yang disajikan dibawah ini.

Contoh Membuat dan mengakses metode statis

```
<?php
    User::pwd_string();
    class User
    {
        static function pwd_string()
        {
            echo "Please enter your password";
        }
    }
?>
```

Perhatikan bagaimana kita memanggil kelas itu sendiri, bersama dengan metode statis, menggunakan titik dua ganda (juga dikenal sebagai operator resolusi lingkup), bukan ->. Fungsi statis berguna untuk melakukan tindakan yang berkaitan dengan kelas itu sendiri, tetapi tidak untuk instance kelas tertentu.. Catatan: Jika kita mencoba mengakses properti $this->, atau properti objek lainnya dari dalam fungsi statis, kita akan menerima pesan kesalahan.

Mendeklarasikan Properti

Tidak perlu mendeklarasikan properti secara eksplisit di dalam kelas, karena properti dapat didefinisikan secara implisit saat pertama kali digunakan. Untuk mengilustrasikan hal ini, dalam Contoh kelas Pengguna tidak memiliki properti dan metode tetapi merupakan kode legal.

Contoh Mendefinisikan properti secara implisit

```
<?php
    $object1 = new User();
    $object1->name = "Alice";
    echo $object1->name;
    class User {}
?>
```

Kode ini dengan benar menampilkan string Alice tanpa masalah, karena PHP secara implisit mendeklarasikan variabel $object1->name. Tetapi pemrograman semacam ini dapat menyebabkan bug yang sangat sulit ditemukan, karena nama dideklarasikan dari luar kelas. Untuk membantu diri kita sendiri dan siapa pun yang akan memelihara kode Kita, saya sarankan kita membiasakan diri untuk selalu mendeklarasikan properti kita secara eksplisit di dalam kelas. Kita akan senang melakukannya. Juga, ketika kita mendeklarasikan properti di dalam kelas, kita dapat menetapkan nilai default untuk itu. Nilai yang kita gunakan harus berupa konstanta dan bukan hasil dari fungsi atau ekspresi. Contoh dibawah ini menunjukkan beberapa tugas yang valid dan tidak valid.

Contoh Deklarasi properti yang valid dan tidak valid

```
<?php
   class Test
   {
      public $name = "Josh"; // Valid
      public $age = 42; // Valid
      public $time = time(); // Invalid - calls a function
      public $score = $level * 2; // Invalid - uses an expression
   }
?>
```

Mendeklarasikan Konstanta

Dengan cara yang sama kita dapat membuat konstanta global dengan fungsi define, kita dapat mendefinisikan konstanta di dalam kelas. Praktek yang diterima secara umum adalah menggunakan huruf besar untuk membuatnya menonjol, seperti pada Contoh dibawah ini.

Contoh Mendefinisikan konstanta dalam kelas

```
<?php
   Translate::lookup();
   class Translate
   {
      const ENGLISH = 0;
      const SPANISH = 1;
      const FRENCH = 2;
      const GERMAN = 3;
      // ...
      static function lookup()
      {
         echo self::SPANISH;
      }
   }
?>
```

Kita dapat mereferensikan konstanta secara langsung, menggunakan kata kunci self dan operator titik dua ganda. Perhatikan bahwa kode ini memanggil kelas secara langsung, menggunakan operator titik dua ganda pada baris 1, tanpa membuat turunannya terlebih dahulu. Seperti yang kita harapkan, nilai yang diprint saat kita menjalankan kode ini adalah 1. Ingatlah bahwa setelah kita menentukan konstanta, kita tidak dapat mengubahnya.

4.8 RUANG LINGKUP PROPERTI DAN METODE DI PHP 5

PHP 5 menyediakan tiga kata kunci untuk mengontrol ruang lingkup properti dan metode:

public

Properti ini adalah default saat kita mendeklarasikan variabel menggunakan var atau kata kunci public, atau saat variabel dideklarasikan secara implisit saat pertama kali digunakan. Kata kunci var dan public dapat dipertukarkan karena, meskipun tidak digunakan lagi, var dipertahankan untuk kompatibilitas dengan versi PHP sebelumnya. Metode diasumsikan public secara default.

protected

Properti dan metode (anggota) ini hanya dapat direferensikan oleh metode kelas objek dan subkelas mana pun.

private

Anggota ini hanya dapat direferensikan dengan metode dalam kelas yang sama — bukan oleh subkelas.

Inilah cara memutuskan mana yang perlu kita gunakan:

- Gunakan public ketika kode luar harus mengakses anggota ini dan memperluas kelas juga harus mewarisinya.
- Gunakan protected ketika kode luar seharusnya tidak mengakses anggota ini tetapi memperluas kelas harus mewarisinya.
- Gunakan private ketika kode luar tidak boleh mengakses anggota ini dan memperluas kelas juga tidak boleh mewarisinya.

Contoh dibawah ini mengilustrasikan penggunaan kata kunci ini.

Contoh Mengubah properti dan cakupan metode

```
<?php
    class Example
    {
        var $name = "Michael"; // Same as public but deprecated
        public $age = 23; // Public property
        protected $usercount; // Protected property
        private function admin() // Private method
        {
            // Admin code goes here
        }
    }
?>
```

Properti dan Metode Statis

Sebagian besar data dan metode berlaku untuk instance kelas. Misalnya, di kelas Pengguna, kita ingin melakukan hal-hal seperti mengatur kata sandi pengguna tertentu atau memeriksa kapan pengguna telah terdaftar. Fakta dan operasi ini berlaku secara terpisah untuk setiap pengguna dan oleh karena itu menggunakan properti dan metode khusus instance. Tetapi terkadang kita ingin mempertahankan data tentang seluruh kelas. Misalnya, untuk melaporkan berapa banyak pengguna yang terdaftar, kita akan menyimpan variabel yang berlaku untuk seluruh kelas Pengguna. PHP menyediakan properti dan metode statis untuk data tersebut.

Seperti ditunjukkan secara singkat dalam Contoh metode statis, mendeklarasikan anggota kelas statis membuat mereka dapat diakses tanpa instantiasi kelas. Properti yang dideklarasikan statis tidak dapat diakses secara langsung dalam instance kelas, tetapi metode statis bisa. Contoh dibawah ini mendefinisikan kelas yang disebut Uji dengan properti statis dan metode publik.

Contoh Mendefinisikan kelas dengan properti statis

```
<?php
    $temp = new Test();
    echo "Test A: " . Test::$static_property . "<br>";
    echo "Test B: " . $temp->get_sp() . "<br>";
    echo "Test C: " . $temp->static_property . "<br>";
```

```
class Test
{
    static $static_property = "I'm static";
    function get_sp()
    {
        return self::$static_property;
    }
}
?>
```

Ketika kita menjalankan kode ini, ia mengembalikan output berikut:

```
Test A: I'm static
Test B: I'm static
Notice: Undefined property: Test::$static_property
Test C:
```

Contoh ini menunjukkan bahwa properti $static_property dapat direferensikan secara langsung dari kelas itu sendiri melalui operator titik dua ganda dalam Uji A. Selain itu, Uji B dapat memperoleh nilainya dengan memanggil metode get_sp dari objek $temp, yang dibuat dari kelas Uji. Tetapi Uji C gagal, karena properti static $static_property tidak dapat diakses oleh objek $temp. Perhatikan bagaimana metode get_sp mengakses $static_property menggunakan kata kunci self. Ini adalah cara di mana properti statis atau konstanta dapat langsung diakses di dalam kelas.

Warisan

Setelah kita menulis sebuah kelas, kita dapat memperoleh subkelas darinya. Ini dapat menghemat banyak penulisan ulang kode yang melelahkan: kita dapat mengambil kelas yang mirip dengan kelas yang perlu kita tulis, memperluasnya ke subkelas, dan hanya memodifikasi bagian-bagian yang berbeda. Kita mencapai ini menggunakan operator extends. Dalam Contoh dibawah, kelas Pelanggan dinyatakan sebagai subkelas Pengguna melalui operator ekstensi.

Contoh Mewarisi dan memperluas kelas

```
<?php
    $object = new Subscriber;
    $object->name = "Guswi";
    $object->password = "pword";
    $object->phone = "081 2345 6789";
    $object->email = "guswi@bloggs.com";
    $object->display();
    class User
    {
        public $name, $password;
        function save_user()
        {
        echo "Save User code goes here";
        }
        }
        class Subscriber extends User
```

```
        {
        public $phone, $email;
        function display()
        {
            echo "Name: " . $this->name . "<br>";
            echo "Pass: " . $this->password . "<br>";
            echo "Phone: " . $this->phone . "<br>";
            echo "Email: " . $this->email;
        }
    }
?>
```

Kelas User asli memiliki dua properti, $name dan $password, dan metode untuk menyimpan pengguna saat ini ke database. Subscriber memperluas kelas ini dengan menambahkan dua properti tambahan, $phone dan $email, dan menyertakan metode untuk menampilkan properti objek saat ini menggunakan variabel $this, yang mengacu pada nilai objek yang sedang diakses saat ini. Output dari kode ini adalah:

```
Name: Guswi
Pass: pword
Phone: 081 2345 6789
Email: guswi@bloggs.com
```

Operator parent

Jika kita menulis metode di subkelas dengan nama yang sama dengan yang ada di kelas induknya, pernyataannya akan menggantikan pernyataan kelas induk. Terkadang ini bukan perilaku yang kita inginkan dan kita perlu mengakses metode orang tua. Untuk melakukan ini, kita dapat menggunakan operator induk, seperti pada Contoh dibawah ini.

Contoh Mengganti metode dan menggunakan operator induk

```
<?php
    $object = new Anak;
    $object->test();
    $object->test2();
    class Ayah
    {
        function test()
        {
            echo "[Class ayah] I am your Father<br>";
        }
        }
        class anak extends ayah
        {
        function test()
        {
            echo "[Class anak] I am Agus<br>";
        }
        function test2()
        {
```

```
            parent::test();
        }
    }
?>
```

Kode ini membuat kelas bernama Ayah dan kemudian subkelas bernama Anak yang mewarisi properti dan metodenya, lalu menimpa pengujian metode. Oleh karena itu, ketika baris 2 memanggil pengujian metode, metode baru dijalankan. Satu-satunya cara untuk mengeksekusi metode pengujian yang diganti di kelas Ayah adalah dengan menggunakan operator induk, seperti yang ditunjukkan pada fungsi test2 dari kelas Anak. Kode menghasilkan yang berikut:

```
[Class Son] I am Agus
[Class Dad] I am your Father
```

Jika kita ingin memastikan bahwa kode kita memanggil metode dari kelas saat ini, kita dapat menggunakan kata kunci self, seperti ini:

```
self::method();
```

Subkelas constructor

Ketika kita memperluas kelas dan mendeklarasikan konstruktor kita sendiri, kita harus menyadari bahwa PHP tidak akan secara otomatis memanggil metode konstruktor dari kelas induk. Jika kita ingin memastikan bahwa semua kode inisialisasi dijalankan, subkelas harus selalu memanggil *konstruktor parent*, lihat contoh dibawah ini.

Contoh Memanggil konstruktor kelas parent

```
<?php
    $object = new Tiger();
    echo "Tigers have..."<br>”;
    echo "Fur: " . $object->fur . "<br>";
    echo "Stripes: " . $object->stripes;
    class Wildcat
    {
        public $fur; // Wildcats have fur
        function __construct()
        {
            $this->fur = "TRUE";
        }
    }
    class Tiger extends Wildcat
    {
        public $stripes; // Tigers have stripes
        function __construct()
        {
            parent::__construct(); // Call parent constructor first $this->stripes = "TRUE";
        }
    }
?>
```

Contoh ini mengambil keuntungan dari warisan dengan cara yang khas. Kelas Wildcat telah membuat properti $fur, yang ingin kita gunakan kembali, jadi kita membuat kelas Tiger untuk mewarisi $fur dan sebagai tambahan membuat properti lain, $stripes. Untuk memverifikasi bahwa kedua konstruktor telah dipanggil, program menampilkan yang berikut:

```
Tigers have...
Fur: TRUE
Stripes: TRUE
```

Metode akhir

Ketika kita ingin mencegah subclass dari override metode superclass, kita dapat menggunakan kata kunci final. Contoh dibawah ini menunjukkan caranya.

Contoh Membuat metode akhir

```
<?php
    class User
    {
        final function copyright()
        {
            echo "Kelas ini ditulis oleh Agus Wibowo";
        }
    }
?>
```

Setelah kita mencerna isi bab ini, kita harus memiliki perasaan yang kuat tentang apa yang dapat dilakukan PHP untuk Kita. Kita harus dapat menggunakan fungsi dengan mudah dan, jika diinginkan, menulis kode berorientasi objek.

BAB 5
MEMBANGUN WEBSITE KHUSUS MEMBER

Banyak situs web rahasia — terbatas hanya untuk pengguna yang berwenang — atau memiliki bagian rahasia. Situs web semacam itu mengharuskan pengguna untuk masuk sebelum mereka dapat melihat informasi rahasia. Berikut adalah beberapa contoh situasi di mana situs web mungkin membatasi akses:

- Administrasi e-commerce: Banyak pedagang online mengharuskan pelanggan untuk masuk sehingga informasi mereka dapat disimpan untuk transaksi di masa mendatang. Informasi pelanggan, khususnya informasi keuangan, perlu dilindungi dari pandangan publik.
- Kerahasiaan: Banyak situs web perlu membatasi informasi untuk orang-orang tertentu. Misalnya, informasi perusahaan mungkin dibatasi untuk staf perusahaan atau anggota departemen tertentu.
- Akses berbayar: Beberapa situs web menyediakan akses ke informasi yang tersedia untuk dijual, sehingga informasi tersebut perlu dibatasi untuk orang yang telah membayarnya.

Login pengguna adalah salah satu aplikasi paling umum di web, dengan banyak kegunaan. Kita yakin kita telah melihat dan masuk ke banyak aplikasi masuk. Jika kita perlu membuat aplikasi masuk yang rumit. Di sini, kita memberi tahu kita tentang beberapa fitur penting dari jenis aplikasi ini dan kemudian memandu kita dalam membuat semua elemen yang diperlukan: basis data pengguna, formulir web untuk mengumpulkan informasi dan memasukkan pengguna, dan semua detail backend yang memungkinkan jenis ini aplikasi berjalan dengan lancar.

5.1 MEMAHAMI SITUS KHUSUS ANGGOTA

Aplikasi login pengguna bisa sangat sederhana, seperti aplikasi di mana administrator menyiapkan daftar pengguna yang valid. Siapa pun yang mencoba mengakses file yang dilindungi akan diminta untuk memasukkan nama pengguna dan kata sandi, yang dicentang dengan daftar pengguna yang valid. Di sisi lain, aplikasi login bisa jauh lebih rumit. Itu dapat memungkinkan pengunjung situs web untuk mendaftar untuk mengakses, mengatur akunnya sendiri. Aplikasi mungkin mengumpulkan informasi dari pelanggan saat mereka mendaftar. Aplikasi mungkin memberikan kemampuan bagi pengguna untuk mengelola akun mereka sendiri. Fitur-fitur yang dapat disediakan oleh aplikasi login beragam. Fungsi dasar dari aplikasi login dalam bab ini adalah untuk memungkinkan pengguna yang terdaftar untuk masuk ke situs web dan untuk mencegah pengguna yang belum mendaftar. Fungsi utama kedua adalah untuk memungkinkan pengguna untuk mendaftar, menyimpan informasi mereka dalam database. Untuk memenuhi fungsionalitas dasarnya, aplikasi login pengguna harus melakukan hal berikut:

- Beri pelanggan pilihan apakah akan mendaftar untuk akses situs web atau masuk ke situs web jika mereka sudah terdaftar.
- Menampilkan formulir pendaftaran yang memungkinkan pelanggan baru untuk mengetik informasi pendaftaran mereka. Informasi yang akan dikumpulkan dalam formulir dibahas di bagian berikut, “Membuat Basis Data Pengguna.”

- Validasi informasi yang disampaikan dalam formulir. Pastikan bidang yang wajib diisi tidak kosong dan informasi yang dikirimkan dalam format yang benar.
- Menyimpan informasi yang telah divalidasi dalam database.
- Menampilkan form login yang menanyakan username dan password pelanggan terdaftar.
- Bandingkan nama pengguna dan kata sandi yang dimasukkan dengan nama pengguna dan kata sandi di database. Jika kecocokan ditemukan, kirim halaman web dari situs ke pelanggan. Jika tidak ditemukan kecocokan, berikan kesempatan kepada pelanggan untuk mencoba login lain.

Selain kemampuan untuk mendaftar dan masuk, aplikasi masuk bisa menjadi jauh lebih kompleks, memberikan kemampuan bagi administrator untuk menetapkan peran ke akun tertentu. Misalnya, pengguna mungkin adalah administrator yang dapat melihat dan mengubah detail akun pengguna lain. Meskipun fungsionalitas itu berada di luar cakupan bab ini, ini adalah fungsi lain untuk sistem otentikasi.

5.2 MEMBUAT DATABASE PENGGUNA

Desain aplikasi membutuhkan database yang menyimpan informasi pengguna. Basis data adalah inti dari aplikasi ini. Basis data diperlukan untuk menyimpan nama pengguna dan kata sandi semua pengguna yang diizinkan mengakses situs web. Seringkali, database digunakan untuk menyimpan lebih banyak informasi tentang pelanggan. Informasi ini dapat digunakan untuk tujuan pemasaran.

Aplikasi login pada bab ini mengasumsikan bahwa pengguna adalah pelanggan yang bersedia memberikan nama, alamat, dan informasi lainnya. Jenis aplikasi ini paling sesuai untuk situs yang menjual produk ke pelanggan. Basis data pengguna bernama Pelanggan.

Merancang database Pelanggan

Tugas desain pertama kita adalah memilih informasi yang ingin kita simpan di database Pelanggan. Paling tidak, kita perlu menyimpan nama pengguna dan kata sandi yang dapat digunakan pengguna untuk masuk. Ini juga berguna untuk mengetahui kapan akun pengguna dibuat. Dalam memutuskan informasi mana yang akan dikumpulkan selama pendaftaran pengguna, kita perlu menyeimbangkan keinginan kita untuk mengumpulkan semua informasi yang berpotensi berguna yang dapat kita pikirkan dengan keinginan pengguna kita untuk menghindari formulir yang terlihat terlalu memakan waktu dan keengganan mereka untuk memberikan keluar informasi pribadi. Salah satu kompromi adalah meminta beberapa informasi opsional. Pengguna yang tidak keberatan akan memasukkannya, dan mereka yang keberatan dapat membiarkannya kosong. Beberapa informasi diperlukan agar situs web kita dapat menjalankan fungsinya. Misalnya, pengguna dapat dengan mudah melihat bahwa situs yang akan mengirimi mereka sesuatu perlu mengumpulkan nama dan alamat. Namun, mereka mungkin tidak mengerti mengapa kita memerlukan nomor telepon. Bahkan jika kita memerlukannya, pengguna terkadang memasukkan nomor telepon palsu. Jadi, kecuali kita memiliki audiens yang terikat, seperti karyawan Kita, yang harus memberikan semua yang kita minta, pikirkan baik-baik informasi apa yang harus dikumpulkan. Sangat mudah bagi pengguna untuk meninggalkan situs web kita saat merasa kesal. Bukannya mereka berkendara berkilo-kilo meter ke toko kita dan mencari tempat parkir selama berjam-jam. Mereka dapat pergi hanya dengan sekali klik.

Untuk contoh aplikasi dalam bab ini, asumsikan situs web adalah toko online yang menjual produk. Dengan demikian, kita perlu mengumpulkan informasi kontak pelanggan. Kita yakin kita memerlukan nomor teleponnya jika kita perlu menghubunginya tentang pesanannya. Sebagian besar pelanggan bersedia memberikan nomor telepon ke pengecer online terkemuka, menyadari bahwa pesanan dapat memiliki masalah yang perlu didiskusikan. Sisa dari bagian ini membahas rincian informasi dan penyimpanannya dalam database MySQL. Database hanya berisi satu tabel. Informasi pelanggan disimpan dalam tabel, satu record (baris) untuk setiap pelanggan.

Mengakses database Pelanggan

PHP menyediakan fungsi MySQL untuk mengakses database kita dari skrip PHP Kita. Fungsi MySQL meneruskan informasi yang diperlukan untuk mengakses database, seperti nama akun MySQL dan kata sandi. Nama akun dan kata sandi MySQL tidak terkait dengan nama akun atau kata sandi lain yang kita miliki, seperti kata sandi untuk masuk ke sistem. Dalam aplikasi ini, informasi yang dibutuhkan oleh fungsi mysqli PHP disimpan dalam file terpisah bernama dbstuff.inc. File ini disimpan dalam direktori di luar ruang web, untuk alasan keamanan. File berisi informasi yang mirip dengan berikut ini:

```
<?php
define(“DBHOST”, “YOURHOST”);
define(“DBUSER”, “YOURUSER”);
define(“DBPASS”, “YOURPASSWORD”);
define(“DB”,”CustomerDirectory”);
?>
```

Perhatikan tag PHP di awal dan akhir file. Jika tag ini tidak disertakan, informasi tersebut mungkin ditampilkan di halaman web untuk dilihat oleh seluruh dunia. Tidak apa yang kita inginkan sama sekali. Untuk alasan keamanan, file ini disimpan dalam direktori di luar ruang web. Kita dapat mengatur direktori include di file php.ini Kita. Database ini dimaksudkan untuk menyimpan data yang dimasukkan oleh pelanggan — bukan oleh Kita. Ini akan kosong ketika aplikasi pertama kali tersedia untuk pelanggan sampai pelanggan menambahkan data. Saat kita menguji skrip aplikasi Kita, skrip akan menambahkan baris ke database. Kita perlu menambahkan baris dengan nama pengguna dan kata sandi untuk penggunaan kita sendiri saat menguji skrip

5.3 MEMBUAT FUNGSI DASAR

Langkah pertama dalam membuat aplikasi besar apa pun adalah membuat beberapa file dasar yang akan digunakan untuk menampung fungsi generik. Untuk aplikasi ini, gunakan file validasi tersebut bersama dengan file fungsi utama yang kemudian akan membutuhkan file lain.

Ini mewakili perubahan konseptual penting dari formulir. File fungsi akan bertanggung jawab untuk memulai sesi, menyiapkan konstanta apa pun yang mungkin kita perlukan, dan termasuk file lain yang diperlukan. Ini menyelamatkan kita dari keharusan mengingat apa yang harus disertakan di mana dan dari keharusan mengingat untuk memulai sesi di mana saja. File fungsi dasar kita akan disebut functions.inc dan akan ditempatkan di root dokumen, ini ditunjukkan pada skrip dibawah ini.

Skrip File Fungsi Dasar

```
<?php
//generic file for generic functions and other includes
session_start();
require_once("../dbstuff.inc");
require_once("validation.inc");
?>
```

Seperti yang kita lihat dari daftar skrip diatas, sesi dimulai dan dua file diperlukan: file dbstuff.inc yang kita lihat di bagian sebelumnya dan file validasi.inc, yang ditampilkan pada skrip dibawah ini.
Skrip File validation.inc

```
<?php
function is_valid_state($state) {
 $validStates = array("AL","CA","CO","FL","IL","NJ","N
                Y","WI");
 if (in_array($state,$validStates)) {
 return true;
 } else {
 return false;
 }
} //end function is_valid_state
function is_valid_zip($zip) {
            if (preg_match('/^[\d]+$/',$zip)) {
            return true;
            } else if (strlen($zip) == 5 || strlen($zip) == 9) {
            return true;
            } else {
            return false;
            }
} //end function is_valid_zip
?>
```

Perubahan utama validasi.inc adalah pada larik status yang valid. Saat kita menelusuri bab ini, file lain akan ditambahkan ke file functions.inc dan fungsi lain dapat ditambahkan sesuai kebutuhan Kita.

5.4 MEMBUAT HALAMAN PENDAFTARAN

Ada beberapa bidang tambahan, berdasarkan persyaratan data untuk aplikasi ini, dan ada require_once di bagian atas file untuk menyertakan file fungsi generik Kita. Daftar skrip dibawah ini menunjukkan kode untuk halaman registrasi, yang disebut register.php

Skrip Halaman Pendaftaran

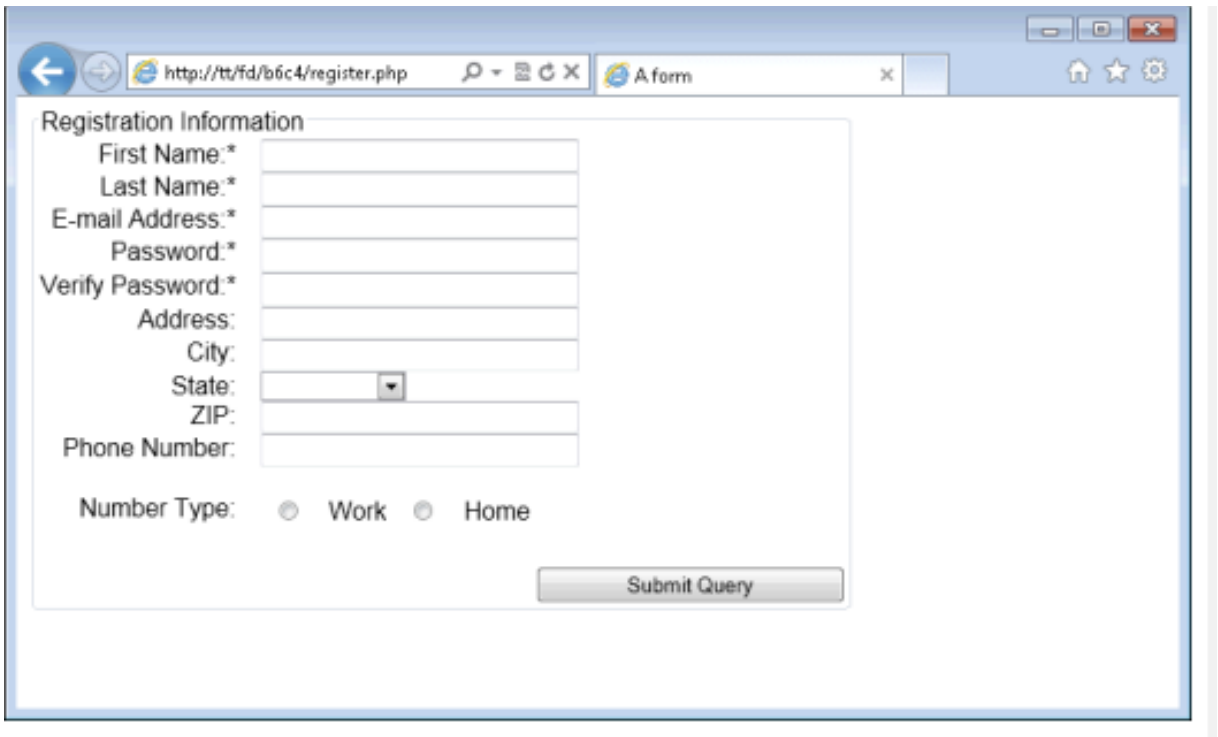
```
<?php require_once("functions.inc"); ?>
<!doctype html>
```

```
<html>
<head>
<script type=”text/javascript” src=”https://ajax.googleapis.
com/ajax/libs/jquery/1.8.3/jquery.min.js”></script>
<script type=”text/javascript” src=”register.js”></script>
<link rel=”stylesheet” type=”text/css” href=”form.css”>
<title>A form</title>
</head>
<body>
<form id=”userForm” method=”POST” action=”register-process.
php”>
<div>
 <fieldset>
 <legend>Registration Information</legend>
 <div id=”errorDiv”>
<?php
 if (isset($_SESSION[‘error’]) && isset($_
SESSION[‘formAttempt’])) {
 unset($_SESSION[‘formAttempt’]);
 print “Errors encountered<br />\n”;
foreach ($_SESSION[‘error’] as $error) {
 print $error . “<br />\n”;
 } //end foreach
 } //end if
?>
          </div>
           <label for=”fname”>First Name:* </label>
           <input type=”text” id=”fname” name=”fname”>
           <span class=”errorFeedback errorSpan”
id=”fnameError”>First Name is required</span>
           <br />
           <label for=”lname”>Last Name:* </label>
           <input type=”text” id=”lname” name=”lname”>
           <span class=”errorFeedback errorSpan”
id=”lnameError”>Last Name is required</span>
           <br />
           <label for=”email”>E-mail Address:* </label>
           <input type=”text” id=”email” name=”email”>
           <span class=”errorFeedback errorSpan”
id=”emailError”>E-mail is required</span>
           <br />
           <label for=”password1”>Password:* </label>
           <input type=”password” id=”password1”
name=”password1”>
           <span class=”errorFeedback errorSpan”
id=”password1Error”>Password required</span>
```

```
        <br />
        <label for=”password2”>Verify Password:* </label>
        <input type=”password” id=”password2”
name=”password2”>
        <span class=”errorFeedback errorSpan”
id=”password2Error”>Passwords don’t match</span>
        <br />
        <label for=”addr”>Address: </label>
        <input type=”text” id=”addr” name=”addr”>
        <br />
        <label for=”city”>City: </label>
        <input type=”text” id=”city” name=”city”>
        <br />
        <label for=”state”>State: </label>
        <select name=”state” id=”state”>
        <option></option>
        <option value=”AL”>Alabama</option>
        <option value=”CA”>California</option>
        <option value=”CO”>Colorado</option>
        <option value=”FL”>Florida</option>
        <option value=”IL”>Illinois</option>
        <option value=”NJ”>New Jersey</option>
        <option value=”NY”>New York</option>
       <option value=”WI”>Wisconsin</option>
        </select>
        <br />
        <label for=”zip”>ZIP: </label>
        <input type=”text” id=”zip” name=”zip”>
        <br />
        <label for=”phone”>Phone Number: </label>
        <input type=”text” id=”phone” name=”phone”>
        <span class=”errorFeedback errorSpan”
id=”phoneError”>Format: xxx-xxx-xxxx</span>
        <br />
        <br />
        <label for=”work”>Number Type:</label>
        <input class=”radioButton” type=”radio”
name=”phonetype” id=”work” value=”work”>
        <label class=”radioButton” for=”work”>Work</label>
        <input class=”radioButton” type=”radio”
name=”phonetype” id=”home” value=”home”>
        <label class=”radioButton” for=”home”>Home</label>
        <span class=”errorFeedback errorSpan phoneTypeError”
id=”phonetypeError”>Please choose an option</span>
        <br />
        <input type=”submit” id=”submit” name=”submit”>
```

```
        </fieldset>
</div>
</form>
</body>
</html>
```

Jika dilihat di browser, tampilan halamannya seperti di gambar dibawah ini



Gambar 5.1 Tata letak halaman pendaftaran.

Halaman PHP proses pendaftaran, yang disebut sebagai tindakan formulir dari halaman register.php mencakup banyak penanganan kesalahan yang sama. Selain itu, fungsi registrasi juga disertakan pada halaman tersebut. Daftar skrip dibawah ini menunjukkan halaman registerprocess.php.

Skrip Halaman proses register

```
<?php
require_once('functions.inc');
//prevent access if they haven't submitted the form.
if (!isset($_POST['submit'])) {
 die(header("Location: register.php"));
}
$_SESSION['formAttempt'] = true;
if (isset($_SESSION['error'])) {
 unset($_SESSION['error']);
}
      $_SESSION['error'] = array();
$required = array("lname","fname","email","password1","passw
ord2");
```

```
//Check required fields
foreach ($required as $requiredField) {
if (!isset($_POST[$requiredField]) || $_POST[$requiredField]
== "") {
                $_SESSION['error'][] = $requiredField . " is
      required.";
 }
}
      if (!preg_match('/^[\w .]+$/',$_POST['fname'])) {
       $_SESSION['error'][] = "First Name must be letters and
numbers only.";
}
      if (!preg_match('/^[\w .]+$/',$_POST['lname'])) {
                $_SESSION['error'][] = "Last Name must be letters and
          numbers only.";
}
if (isset($_POST['state']) && $_POST['state'] != "") {
 if (!is_valid_state($_POST['state'])) {
 $_SESSION['error'][] = "Please choose a valid state";
 }
}
if (isset($_POST['zip']) && $_POST['zip'] != "") {
 if (!is_valid_zip($_POST['zip'])) {
 $_SESSION['error'][] = "ZIP code error.";
 }
}
if (isset($_POST['phone']) && $_POST['phone'] != "") {
 if (!preg_match('/^[\d]+$/',$_POST['phone'])) {
$_SESSION['error'][] = "Phone number should be digits
only";
 } else if (strlen($_POST['phone']) < 10) {
 $_SESSION['error'][] = "Phone number must be at least
10 digits";
 }
 if (!isset($_POST['phonetype']) || $_POST['phonetype'] ==
"") {
 $_SESSION['error'][] = "Please choose a phone number
type";
 } else {
 $validPhoneTypes = array("work","home");
 if (!in_array($_POST['phonetype'],$validPhoneTypes))
{
 $_SESSION['error'][] = "Please choose a valid
phone number type.";
 }
 }
```

```
}
if (!filter_var($_POST['email'],FILTER_VALIDATE_EMAIL)) {
 $_SESSION['error'][] = "Invalid e-mail address";
}
if ($_POST['password1'] != $_POST['password2']) {
 $_SESSION['error'][] = "Passwords don't match";
}
//final disposition
if (count($_SESSION['error']) > 0) {
 die(header("Location: register.php"));
} else {
 if(registerUser($_POST)) {
 unset($_SESSION['formAttempt']);
 die(header("Location: success.php"));
 } else {
error_log("Problem registering user: {$_POST['email']}");
 $_SESSION['error'][] = "Problem registering account";
 die(header("Location: register.php"));
 }
}
function registerUser($userData) {
 $mysqli = new mysqli(DBHOST,DBUSER,DBPASS,DB);
 if ($mysqli->connect_errno) {
error_log("Cannot connect to MySQL: " . $mysqli->connect_error);
 return false;
 }
 $email = $mysqli->real_escape_string($_POST['email']);
//check for an existing user
 $findUser = "SELECT id from Customer where email =
'{$email}'";
 $findResult = $mysqli->query($findUser);
 $findRow = $findResult->fetch_assoc();
 if (isset($findRow['id']) && $findRow['id'] != "") {
 $_SESSION['error'][] = "A user with that e-mail
address already exists";
 return false;
 }
      $lastName = $mysqli->real_escape_string($_POST['lname']);
      $firstName = $mysqli->real_escape_string($_
POST['fname']);
      $cryptedPassword = crypt($_POST['password1']);
      $password = $mysqli->real_escape_
string($cryptedPassword);
      if (isset($_POST['addr'])) {
      $street = $mysqli->real_escape_string($_
POST['addr']);
```

```
        } else {
        $street = "";
        }
        if (isset($_POST['city'])) {
        $city = $mysqli->real_escape_string($_POST['city']);
        } else {
        $city = "";
        }
        if (isset($_POST['state'])) {
        $state = $mysqli->real_escape_string($_
POST['state']);
 } else {
 $state = "";
 }
 if (isset($_POST['zip'])) {
 $zip = $mysqli->real_escape_string($_POST['zip']);
 } else {
 $zip = "";
 }
 if (isset($_POST['phone'])) {
 $phone = $mysqli->real_escape_string($_
POST['phone']);
        } else {
        $phone = "";
        }
        if (isset($_POST['phonetype'])) {
        $phoneType = $mysqli->real_escape_string($_
POST['phonetype']);
 } else {
$phoneType = "";
 }
 $query = "INSERT INTO Customer (email,create_
date,password,last_name,first_name,street,city,state,zip,p
hone,phone_type) " .
 " VALUES ('{$email}',NOW(),'{$password}','{$lastName}
','{$firstName}'" .
 ",'{$street}','{$city}','{$state}','{$zip}','{$phone}
','{$phoneType}')";
 if ($mysqli->query($query)) {
 $id = $mysqli->insert_id;
 error_log("Inserted {$email} as ID {$id}");
 return true;
 } else {
 error_log("Problem inserting {$query}");
 return false;
 }
```

```
} //end function registerUser
?>
```

Fungsi registerUser dipanggil jika tidak ada kesalahan lain yang ditemukan. Oleh karena itu, pada saat kita masuk ke fungsi registerUser, kita sudah tahu bahwa ada alamat email yang valid, bahwa kata sandinya cocok, dan semua bidang yang diperlukan telah diisi. Ini berarti bahwa fungsi registerUser dapat berkonsentrasi pada tugasnya: Dapatkan informasi pengguna yang dimasukkan ke dalam database. Fungsi registerUser pertama terhubung ke database MySQL dengan menggunakan konstanta yang didefinisikan dalam file dbstuff.inc. Dengan asumsi koneksi ada, alamat email diloloskan agar aman digunakan dalam pernyataan SQL. Alamat email kemudian digunakan untuk memeriksa apakah pengguna sudah ada dengan alamat email tersebut. Jika salah satu ditemukan, maka kesalahan diatur dan Boolean false dikembalikan, yang akan memicu tampilan kesalahan.

Tabel MySQL diberi nama customer, dengan huruf besar C. Jika kita mencoba mengaksesnya dengan huruf kecil c, seperti pada pelanggan, kueri akan gagal. Dengan asumsi bahwa pengguna yang ada tidak ditemukan, setiap nilai yang akan dimasukkan ke dalam database kemudian di-escape menggunakan fungsi mysqli_real_escape_string()PHP. Kata sandi juga dienkripsi menggunakan fungsi PHP crypt() bawaan. Pernyataan INSERT dibangun dan dieksekusi terhadap database. Jika pernyataan dijalankan dengan benar, maka ID diambil; jika tidak, kesalahan dihasilkan.

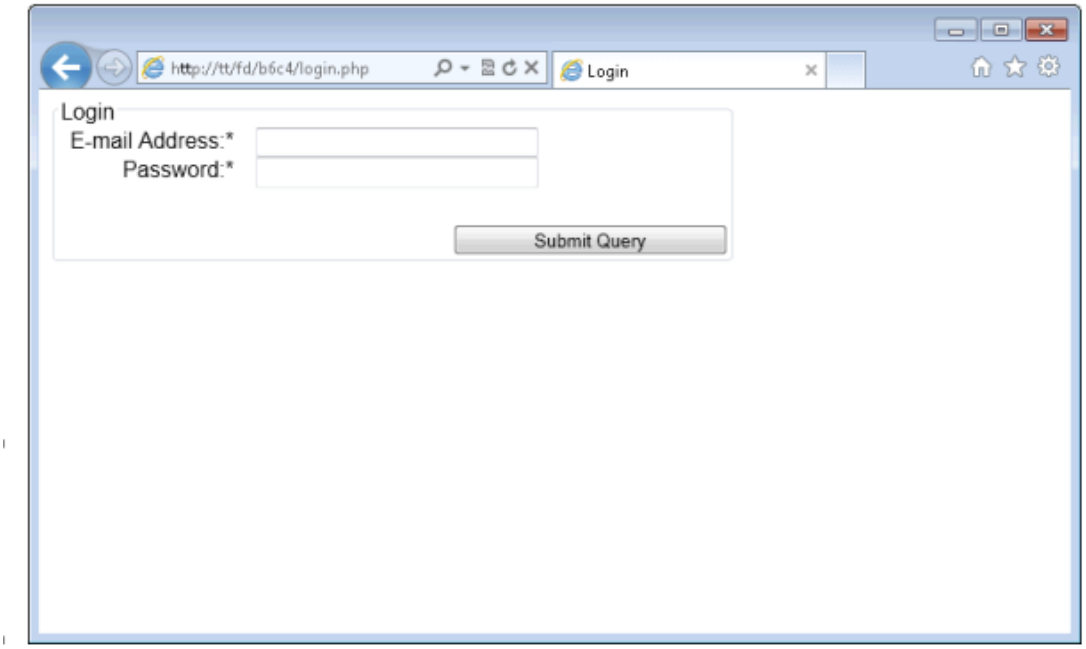
5.5 MEMBANGUN HALAMAN SUKSES

Jika pendaftaran berhasil, pengguna diarahkan ke success.php. Dalam contoh, success.php akan menjadi halaman yang sangat sederhana, tetapi kita dapat membuat halaman serumit yang kita inginkan. Daftar skrip dibawah ini menunjukkan kode untuk halaman sukses.

```
<!doctype html>
<html>
<head>
<title>Registration Success</title>
</head>
<body>
<div>
        Thank you for registering
</div>
<div>
            <a href=”login.php”>Click here to login</a>
</div>
</body>
</html>
```

5.6 MEMBUAT HALAMAN LOGIN

Sekarang kita memiliki kemampuan untuk mendaftarkan pengguna, saatnya membuat halaman terkait dengan masuk ke aplikasi. Halaman login akan terlihat seperti gambar dibawah ini.



Gambar 5.2 Halaman login

Daftar skrip ini menunjukkan kode untuk membuat halaman login.

```
<?php require_once("functions.inc"); ?>
<!doctype html>
<html>
<head>
<script type="text/javascript" src="https://ajax.googleapis.
        com/ajax/libs/jquery/1.8.3/jquery.min.js"></script>
<script type="text/javascript" src="login.js"></script>
<link rel="stylesheet" type="text/css" href="form.css">
<title>Login</title>
</head>
<body>
<form id="loginForm" method="POST" action="login-process.
        php">
<div>
         <fieldset>
         <legend>Login</legend>
         <div id="errorDiv">
<?php
                  if (isset($_SESSION['error']) && isset($_
        SESSION['formAttempt'])) {
                           unset($_SESSION['formAttempt']);
                           print "Errors encountered<br />\n";
                           foreach ($_SESSION['error'] as $error) {
                                    print $error . "<br />\n";
                                     } //end foreach
         } //end if
```

```
?>
        </div>
                <label for=”email”>E-mail Address:* </label>
                <input type=”text” id=”email” name=”email”>
                <span class=”errorFeedback errorSpan”
        id=”emailError”>E-mail is required</span>
                <br />
                <label for=”password”>Password:* </label>
                <input type=”password” id=”password” name=”password”>
                <span class=”errorFeedback errorSpan”
        id=”passwordError”>Password required</span>
                <br />
                <input type=”submit” id=”submit” name=”submit”>
                </fieldset>
</div>
</form>
</body>
</html>
```

5.7 OBJEK PHP

Dengan cara yang hampir sama bahwa fungsi mewakili peningkatan besar dalam kekuatan pemrograman selama hari-hari awal komputasi, di mana terkadang navigasi program terbaik yang tersedia adalah pernyataan GOTO atau GOSUB yang sangat mendasar, *object-oriented programming* (OOP) menggunakan fungsi untuk tingkat yang sama sekali baru. Setelah kita memahami memadatkan bit kode yang dapat digunakan kembali menjadi fungsi, bukanlah lompatan besar untuk mempertimbangkan menggabungkan fungsi dan datanya ke dalam objek. Mari kita ambil situs jejaring sosial yang memiliki banyak bagian. Satu bagian menangani semua fungsi pengguna; yaitu, kode untuk memungkinkan pengguna baru mendaftar dan pengguna yang sudah ada untuk mengubah detail mereka. Dalam PHP standar, kita dapat membuat beberapa fungsi untuk menangani ini dan menyematkan beberapa panggilan ke database MySQL untuk melacak semua pengguna.

Bayangkan betapa mudahnya membuat objek untuk mewakili pengguna saat ini. Untuk melakukan ini, kita bisa membuat kelas, mungkin disebut Pengguna, yang akan berisi semua kode yang diperlukan untuk menangani pengguna dan semua variabel yang diperlukan untuk memanipulasi data di dalam kelas. Kemudian, kapan pun kita perlu memanipulasi data pengguna, kita cukup membuat objek baru dengan kelas Pengguna. Kita dapat memperlakukan objek baru ini seolah-olah itu adalah pengguna sebenarnya. Misalnya, kita dapat memberikan objek nama, kata sandi, dan alamat email; tanyakan apakah pengguna seperti itu sudah ada; dan, jika tidak, buat pengguna baru dengan atribut tersebut. Kita bahkan dapat memiliki objek pesan instan, atau satu untuk mengelola apakah dua pengguna adalah teman.

Membuat Objek User

Dasar untuk bagian yang diautentikasi dari situs pelanggan kita adalah pengguna — khususnya, siapa mereka dan apakah mereka masuk atau tidak. Untuk itu, objek Pengguna akan menyediakan lapisan abstraksi yang membantu, memungkinkan kita untuk menambahkan fungsionalitas nanti saat kita membutuhkannya.

Mendeklarasikan Kelas

Sebelum kita dapat menggunakan objek, kita harus mendefinisikan kelas dengan kata kunci kelas. Definisi kelas berisi nama kelas (yang peka huruf besar/kecil), propertinya, dan metodenya. Contoh dibawah mendefinisikan kelas Pengguna dengan dua properti: $name dan $password (ditunjukkan dengan kata kunci publik—lihat Ruang Lingkup Properti dan Metode di PHP 5). Itu juga membuat instance baru (disebut $object) dari kelas ini.

Contoh mendeklarasikan kelas dan memeriksa objek.

```
<?php
    $object = new User;
    print_r($object);
    class User
    {
        public $name, $password;
        function save_user()
        {
          echo "Save User code goes here";
        }
    }
?>
```

Di sini saya juga menggunakan fungsi tak ternilai yang disebut print_r. Ini meminta PHP untuk menampilkan informasi tentang variabel dalam bentuk yang dapat dibaca manusia. _r adalah singkatan dari "dalam format yang dapat dibaca manusia." Dalam kasus objek baru $object, ia mencetak yang berikut:

```
User Object
(
    [name] =>
    [password] =>
)
```

Namun, browser mengompresi semua spasi, sehingga output di browser sedikit lebih sulit dibaca:

```
User Object ( [name] => [password] => )
```

Bagaimanapun, output mengatakan bahwa $object adalah objek yang ditentukan pengguna yang memiliki nama properti dan kata sandi.

Membuat Objek

Untuk membuat objek dengan kelas tertentu, gunakan kata kunci baru, seperti ini: object = new Class. Berikut adalah beberapa cara di mana kita bisa melakukan ini:

```
$object = new User;
$temp = new User('name', 'password');
```

Pada baris pertama, kita hanya menetapkan objek ke kelas Pengguna. Yang kedua, kita meneruskan parameter ke panggilan. Sebuah kelas mungkin memerlukan atau melarang argumen; itu mungkin juga mengizinkan argumen, tetapi tidak mengharuskannya.

Mengakses Objek

Mari tambahkan beberapa baris lagi ke Contoh sebelumnya dan periksa hasilnya. Contoh dibawah memperluas kode sebelumnya dengan mengatur properti objek dan memanggil metode.

Contoh Membuat dan berinteraksi dengan objek

```
<?php
    $object = new User;
    print_r($object); echo "<br>";
    $object->name = "Joe";
    $object->password = "mypass";
    print_r($object); echo "<br>";
    $object->save_user();
    class User
    {
        public $name, $password;
        function save_user()
        {
            echo "Save User code goes here";
        }
    }
?>
```

Seperti yang kita lihat, sintaks untuk mengakses properti objek adalah $object->property. Demikian juga, kita memanggil metode seperti ini: $object->method(). Kita harus mencatat bahwa properti contoh dan metode tidak memiliki tanda $ di depannya. Jika kita mengawalinya dengan tanda $, kode tidak akan berfungsi, karena akan mencoba merujuk nilai di dalam variabel. Misalnya, ekspresi $object->$property akan mencoba mencari nilai yang ditetapkan ke variabel bernama $property (misalkan nilai tersebut adalah string brown) dan kemudian mencoba mereferensikan properti $object->brown. Jika $property tidak terdefinisi, upaya untuk mereferensikan $object->NULL akan terjadi dan menyebabkan kesalahan. Jika dilihat menggunakan fasilitas View Source browser, output dari Contoh sebelumnya adalah:

```
User Object
(
[name] =>
[password] =>
)
User Object
(
    [name] =>
    Josh [password] => mypass
)
Save User code goes here
```

Sekali lagi, print_r menunjukkan kegunaannya dengan menyediakan konten $object sebelum dan sesudah penetapan properti. Mulai sekarang, saya akan menghilangkan pernyataan print_r, tetapi jika kita bekerja bersama dengan buku ini di server pengembangan Kita, kita dapat memasukkan beberapa untuk melihat dengan tepat apa yang terjadi. Kita juga dapat

melihat bahwa kode dalam metode save_user dieksekusi melalui panggilan ke metode tersebut. Itu mencetak string yang mengingatkan kita untuk membuat beberapa kode.

Catatan: kita dapat menempatkan fungsi dan definisi kelas di mana saja dalam kode Kita, sebelum atau sesudah pernyataan yang menggunakannya. Namun, secara umum, dianggap sebagai praktik yang baik untuk menempatkannya menjelang akhir file.

Kloning Objek

Setelah kita membuat objek, objek itu diteruskan dengan referensi saat kita meneruskannya sebagai parameter. Dalam metafora kotak korek api, ini seperti menyimpan beberapa utas yang melekat pada objek yang disimpan dalam kotak korek api, sehingga kita dapat mengikuti setiap utas yang dilampirkan untuk mengaksesnya. Dengan kata lain, membuat penugasan objek tidak menyalin objek secara keseluruhan. Kita akan melihat cara kerjanya di Contoh dibawah ini, di mana kita mendefinisikan kelas Pengguna yang sangat sederhana tanpa metode dan hanya nama properti.

Contoh Menyalin objek?

```
<?php
    $object1 = new User();
    $object1->name = "Alice";
    $object2 = $object1;
    $object2->name = "Amy";
    echo "object1 name = " . $object1->name . "<br>";
    echo "object2 name = " . $object2->name;
    class User
    {
        public $name;
    }
?>
```

Kita telah membuat objek $object1 dan menetapkan nilai Alice ke properti nama. Kemudian kita membuat $object2, memberinya nilai $object1, dan menetapkan nilai Amy hanya ke properti nama $object2—atau begitulah yang mungkin kita pikirkan. Tetapi kode ini menghasilkan yang berikut:

```
object1 name = Amy
object2 name = Amy
```

Apa yang telah terjadi? Baik $object1 dan $object2 merujuk ke objek yang sama, jadi mengubah properti name dari $object2 menjadi Amy juga menyetel properti itu untuk $object1. Untuk menghindari kebingungan ini, kita dapat menggunakan operator clone, yang membuat instance baru dari kelas dan menyalin nilai properti dari instance asli ke instance baru. Contoh mengilustrasikan penggunaan ini.

Contoh Mengkloning objek

```
<?php
  $object1 = new User();
  $object1->name = "Alice";
  $object2 = clone $object1;
  $object2->name = "Amy";
  echo "object1 name = " . $object1->name . "<br>";
```

```
echo "object2 name = " . $object2->name;
class User
{
    public $name;
}
?
```

Output dari kode ini adalah apa yang kita inginkan pada awalnya:

```
object1 name = Alice
object2 name = Amy
```

Konstruktor

Saat membuat objek baru, kita dapat meneruskan daftar argumen ke kelas yang dipanggil. Ini diteruskan ke metode khusus di dalam kelas, yang disebut konstruktor, yang menginisialisasi berbagai properti. Di masa lalu, kita biasanya memberi metode ini nama yang sama dengan kelasnya, seperti pada Contoh ini :

Contoh Membuat metode konstruktor

```
<?php
    class User
    {
    function User($param1, $param2)
    {
        // Constructor statements go here
        public $username = "Guest";
        }
    }
?>
```

Namun, PHP 5 menyediakan pendekatan yang lebih logis untuk penamaan konstruktor, yaitu dengan menggunakan nama fungsi __construct (yaitu, konstruksi yang didahului oleh dua karakter garis bawah), seperti pada Contoh dibawah ini.

Contoh Membuat metode konstruktor di PHP 5

```
<?php
    class User
    {
    function __construct($param1, $param2)
    {
            // Constructor statements go here public
            $username = "Guest";
        }
    }
?>
```

5.8 MEMBANGUN KELAS USER

Kelas User (biasanya memulai kelas dengan huruf besar di PHP) akan disimpan dalam file bernama ClassUser.php. File itu akan dimasukkan ke dalam file functions.inc dengan baris ini:

require_once(“ClassUser.php”);

Sekarang kelas Pengguna akan tersedia di mana saja yang menggunakan file functions.inc (yang cukup banyak di mana-mana di aplikasi Kita). Kelas Pengguna digunakan untuk mengotentikasi pengguna dan untuk mengatur informasi mereka ke dan dari sesi sehingga dapat digunakan di beberapa halaman aplikasi. Daftar skrip dibawah ini menunjukkan kode untuk *class User.*

```
<?php
class User {
        public $id;
        public $email;
        public $firstName;
        public $lastName;
        public $address;
        public $city;
        public $state;
word = $mysqli->real_escape_
string($pass);
        $query = “SELECT * from Customer WHERE email =
‘{$safeUser}’”;
 if (!$result = $mysqli->query($query)) {
 error_log(“Cannot retrieve account for {$user}”);
 return false;
 }
 // Will be only one row, so no while() loop needed
 $row = $result->fetch_assoc();
 $dbPassword = $row[‘password’];
 if (crypt($incomingPassword,$dbPassword) !=
$dbPassword) {
        error_log(“Passwords for {$user} don’t match”);
        return false;
        }
        $this->id = $row[‘id’];
        $this->email = $row[‘email’];
        $this->firstName = $row[‘first_name’];
        $this->lastName = $row[‘last_name’];
        $this->address = $row[‘street’];
        $this->city = $row[‘city’];
        $this->zip = $row[‘zip’];
        $this->state = $row[‘state’];
        $this->phone = $row[‘phone’];
$this->phoneType = $row[‘phone_type’];
        $this->isLoggedIn = true;
        $this->_setSession();
        return true;
```

```
        } //end function authenticate
        private function _setSession() {
        if (session_id() == '') {
        session_start();
        }
        $_SESSION['id'] = $this->id;
        $_SESSION['email'] = $this->email;
        $_SESSION['firstName'] = $this->firstName;
        $_SESSION['lastName'] = $this->lastName;
        $_SESSION['address'] = $this->address;
        $_SESSION['city'] = $this->city;
        $_SESSION['zip'] = $this->zip;
        $_SESSION['state'] = $this->state;
        $_SESSION['phone'] = $this->phone;
        $_SESSION['phoneType'] = $this->phoneType;
        $_SESSION['isLoggedIn'] = $this->isLoggedIn;
} //end function setSession
        private function _initUser() {
        if (session_id() == '') {
        session_start();
        }
        $this->id = $_SESSION['id'];
        $this->email = $_SESSION['email'];
        $this->firstName = $_SESSION['firstName'];
        $this->lastName = $_SESSION['lastName'];
        $this->address = $_SESSION['address'];
        $this->city = $_SESSION['city'];
        $this->zip = $_SESSION['zip'];
        $this->state = $_SESSION['state'];
        $this->phone = $_SESSION['phone'];
        $this->phoneType = $_SESSION['phoneType'];
        $this->isLoggedIn = $_SESSION['isLoggedIn'];
 } //end function initUser
} //end class User
```

Konstruktor untuk kelas User pertama-tama memeriksa untuk melihat apakah sesi dimulai (ini akan menjadi tema umum untuk sebagian besar fungsi di kelas). Memang, sesi harus sudah dimulai tetapi jika tidak, kita pasti tidak ingin dipusingkan dengan variabel terkait sesi. Jadi, jika sesi belum ada, mulailah. Selanjutnya di konstruktor, periksa untuk melihat apakah pengguna sudah masuk. Jika ya, jalankan fungsi initUser. Fungsi initUser mengambil informasi pengguna dari sesi dan menetapkan setiap elemen informasi mereka sebagai properti.

Fungsi authenticate digunakan untuk memeriksa kredensial yang dimasukkan pada formulir terhadap apa yang ada di database. Koneksi database dibuat dan kueri dibuat menggunakan alamat email yang dimasukkan pada formulir login. Jika tidak ada pengguna yang ditemukan dengan alamat email tersebut, kesalahan akan dicatat di belakang layar dan false dikembalikan dari fungsi tersebut. Dengan asumsi bahwa pengguna ditemukan, kata

sandinya diambil dari database. Kata sandi akan dienkripsi, sama seperti kita memasukkannya saat pengguna mendaftar. Oleh karena itu, kode perlu memanggil fungsi crypt() dengan kata sandi yang masuk dari formulir login dan kata sandi yang diambil dari database. Jika kedua versi terenkripsi cocok, maka kita tahu bahwa pengguna menggunakan kata sandi yang benar. Dengan pengguna yang berhasil diautentikasi, atur berbagai detail dari database ke dalam properti dan panggil fungsi setSession(). Fungsi setSession() mengambil properti dan menyetelnya ke dalam sesi sehingga dapat digunakan di halaman lain aplikasi. Itulah kelas User, setidaknya sejauh ini. Kita menambahkannya sesuai kebutuhan nanti.

Membangun file PHP proses login

Sekarang setelah calss user siap digunakan, kita dapat membuat file login-process.php. File login-process.php adalah tindakan formulir login. Ketika seseorang mengklik tombol Kirim Permintaan untuk masuk, dia akan dikirim ke file ini, yang akan melakukan bisnis untuk mengautentikasinya dan mengirimnya ke tempat yang sesuai. Kode proses login ditunjukkan pada daftar kode dibawah ini:

```
<?php
require_once('functions.inc');
//prevent access if they haven't submitted the form.
if (!isset($_POST['submit'])) {
die(header("Location: login.php"));
}
$_SESSION['formAttempt'] = true;
if (isset($_SESSION['error'])) {
        unset($_SESSION['error']);
}
$_SESSION['error'] = array();
$required = array("email","password");
//Check required fields
foreach ($required as $requiredField) {
if (!isset($_POST[$requiredField]) || $_POST[$requiredField]
== "") {
        $_SESSION['error'][] = $requiredField . " is
required.";
 }
}
if (!filter_var($_POST['email'],FILTER_VALIDATE_EMAIL)) {
        $_SESSION['error'][] = "Invalid e-mail address";
}
if (count($_SESSION['error']) > 0) {
        die(header("Location: login.php"));
} else {
                $user = new User;
if ($user->authenticate($_POST['email'],$_POST['password']))
   {
        unset($_SESSION['formAttempt']);
                        die(header("Location: authenticated.php"));
```

```
        } else {
                        $_SESSION['error'][] = "There was a problem
        with your username or password.";
                die(header("Location: login.php"));
        }
}
?>
```

Kode dari file login-process berbagi banyak logika yang sama dari file register-process sebelumnya di bab ini, untuk proses login adalah instantiasi kelas Pengguna dan penggunaan kelas Pengguna untuk otentikasi. Fungsi authenticate() di kelas User mengembalikan nilai true jika pengguna diautentikasi; oleh karena itu, ini dapat dibungkus dengan if() conditional. Seorang pengguna yang berhasil login akan diarahkan ke halaman yang disebut authenticated.php. Jika login tidak berhasil, pengguna akan dikirim kembali ke login.php dengan kesalahan.

Menambahkan Halaman yang Diautentikasi

Aplikasi kita memiliki kemampuan untuk mendaftarkan pengguna dan membuat mereka masuk. Aplikasi ini menggunakan kelas untuk menangani informasi Pengguna, tetapi sebenarnya tidak ada yang dapat dilakukan pengguna setelah mereka masuk. Pada titik ini, kita bahkan tidak memiliki halaman yang diautentikasi dibuat! Saatnya untuk memperbaikinya.

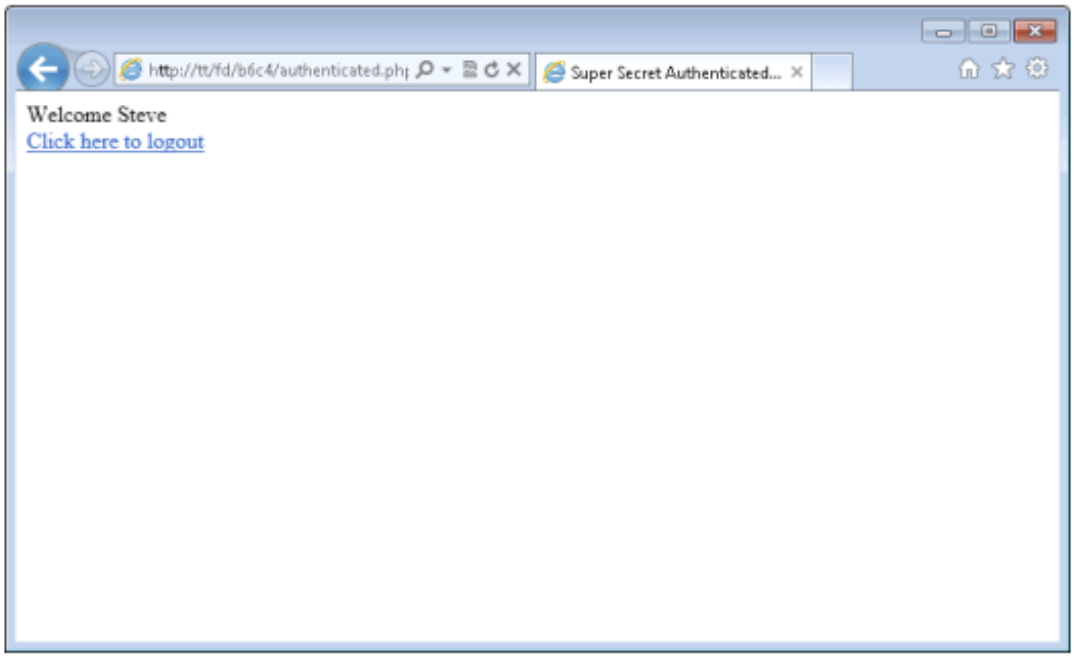
Membangun halaman yang dilindungi

Halaman yang perlu dilindungi — dengan kata lain, halaman yang pengguna harus masuk untuk mengaksesnya — dapat dibangun dengan mudah dengan bantuan kelas Pengguna. Setiap kali user masuk, properti yang disebut isLoggedIn disetel ke Boolean true. Itu berarti kita dapat secara efektif memeriksa apakah pengguna masuk ke halaman mana pun dengan memeriksa properti itu. Session banyak digunakan sebagai bagian dari aplikasi. Kita mungkin tergoda untuk mengakses hal-hal seperti parameter isLoggedIn langsung dari sesi. Namun, praktik terbaik adalah menggunakan antarmuka berorientasi objek (user class) bila memungkinkan. Ada kalanya antarmuka berorientasi objek mungkin perlu melakukan pemeriksaan tambahan untuk melihat apakah pengguna masuk (atau properti lain apa pun yang diminta). Oleh karena itu, dengan menggunakan antarmuka berorientasi objek, kita tetap sejalan dengan teknik abstraksi dan akan memungkinkan fleksibilitas terbesar nanti. Halaman terotentikasi yang digunakan oleh file proses login disebut authenticated.php. Kode untuk authenticated.php pada daftar skrip dibawah ini:

```
<?php
require_once("functions.inc");
$user = new User;
if (!$user->isLoggedIn) {
        die(header("Location: login.php"));
}
?>
<!doctype html>
<html>
<head>
```

```
<title>Super Secret Authenticated Page</title>
</head>
<body>
<div>
<?php print “Welcome {$user->firstName}<br />\n”; ?>
</div>
<div>
        <a href=”logout.php”>Click here to logout</a>
</div>
</body>
</html>
```

Inti dari kode halaman berada tepat di atas, di mana Pengguna baru dibuat dan properti isLoggedIn dicentang. Jika properti isLoggedIn salah, pengguna diarahkan kembali ke halaman login. Jika isLoggedInproperty bernilai true, maka eksekusi halaman akan dilanjutkan dan user disambut ke halaman tersebut, seperti terlihat pada gambar dibawah ini:



Gambar 5.3 Halaman yang diautentikasi.

Kita dapat melihat bahwa halaman yang diautentikasi mengacu pada file logout.php. File itu belum dibuat. Pada dasarnya, setiap halaman yang perlu dilindungi harus memiliki kode ini ditambahkan ke dalamnya:

```
<?php
require_once(“functions.inc”);
$user = new User;
if (!$user->isLoggedIn) {
        die(header(“Location: login.php”));
}
?>
```

Dengan kode itu (dan kelas serta file dukungan yang menyertainya), pengguna tidak dapat mengakses halaman kecuali properti isLoggedIn disetel ke true.

Membangun halaman keluar

Halaman untuk keluar dari aplikasi dengan aman sama pentingnya dengan masuk. Halaman perlu melakukan hal yang jelas, mengubah properti isLoggedIn menjadi false, tetapi juga harus menghapus data pengguna apa pun dari sesi juga. Dan untuk lapisan keamanan ekstra, sesi itu sendiri dapat dihancurkan, seperti yang direkomendasikan dalam manual PHP. Fungsi logout yang sebenarnya harus ditambahkan ke kelas Pengguna, karena itu pada dasarnya adalah bagian dari tugas terkait pengguna. Fungsionalitas logout juga dapat digunakan dari beberapa halaman, sehingga menjadikannya kandidat yang baik untuk abstraksi ke area umum. Ada dua tugas kemudian:

- Bangun fungsi logout dan tambahkan ke kelas Pengguna.
- Bangun halaman logout itu sendiri.

Kita menangani keduanya selanjutnya.

Membuat fungsi logout

Fungsi logout tidak hanya perlu menyetel properti isLoggedIn ke false, tetapi juga perlu menghapus variabel sesi yang terkait dengan login. Melakukan hal ini membantu mencegah pengguna dari kemungkinan masih login atau informasinya tetap berada di browser. Halaman manual PHP untuk session_destroy berisi beberapa kode berguna untuk menghapus sesi sepenuhnya, yang kita sesuaikan untuk fungsi logout Kita; tidak ada gunanya menemukan kembali roda di sini. Kita dapat melihat halaman session_destroy manual PHP di http://php.net/manual/en/function.session-destroy.php. Daftar skrip dibawah ini menunjukkan fungsi logout. Fungsi ini ditambahkan ke file ClassUser.php, di dalam kelas (tepat sebelum tanda kurung kurawal untuk mengakhiri kelas Pengguna).

```
public function logout() {
        $this->isLoggedIn = false;
        if (session_id() == '') {
        session_start();
        }
        $_SESSION['isLoggedIn'] = false;
        foreach ($_SESSION as $key => $value)
$_SESSION[$key] = "";
        unset($_SESSION[$key]);
        }
         $_SESSION = array();
         if (ini_get("session.use_cookies")) {
         $cookieParameters = session_get_cookie_params();
         setcookie(session_name(), '', time() - 28800,
         $cookieParameters['path'],$cookieParameters['
domain'],
         $cookieParameters['secure'],$cookieParameters
                 ['httponly']
                         );
                                 } //end if
        session_destroy();
```

} //end function logout

Fungsi ini menyetel properti isLoggedIn ke false dan kemudian melanjutkan untuk menghapus semua variabel sesi. Jika cookie HTTP digunakan untuk sesi tersebut, cookie baru dikirim ke browser, yang secara efektif membuat cookie kedaluwarsa.

Membangun halaman logout

Ketika pengguna mengklik tautan Logout di mana saja di situs, mereka akan dikirim ke halaman yang disebut logout.php, yang melakukan logout sebenarnya dan mengirim pengguna kembali ke halaman login. Kode untuk halaman logout, yang disebut logout.php, hanya empat baris dan ditunjukkan pada skrip dibawah ini:

```
<?php
require_once("functions.inc");
$user = new User;
$user->logout();
die(header("Location: login.php"));
?>
```

Dengan kode itu, pengguna dapat mendaftar, masuk, dan keluar dari aplikasi. Namun, dua bidang harus ditingkatkan. Pertama, jika pengguna membuka halaman login, kita harus memanggil fungsi logout; kedua, kita juga harus menyetel properti isLoggedIn ke false setiap kali metode autentikasi dipanggil.

Menambahkan Fungsi Email

Pengguna lupa kata sandi mereka. Kadang-kadang mereka bahkan lupa nama pengguna mereka, tetapi karena aplikasi kita menggunakan alamat email sebagai nama pengguna, skenario itu cenderung tidak terjadi. Kita dapat menambahkan kemampuan bagi pengguna untuk mengatur ulang kata sandinya. Melakukannya melibatkan beberapa pekerjaan database tambahan dan halaman baru, jadi kita memberi tahu kita cara melakukannya di sini. Mengirim email sebenarnya agak sepele; itu semua hal seputar pengaturan ulang kata sandi yang menjadi sedikit lebih rumit. Alur keseluruhan untuk pengaturan ulang kata sandi di situs ini akan meminta halaman pengaturan ulang, di mana pengguna dapat memasukkan alamat email mereka. Saat dikirimkan, formulir akan mencari alamat email untuk melihat apakah itu akun yang valid dan kemudian akan membuat URL unik untuk pengaturan ulang kata sandi. URL unik ini akan berisi string karakter acak semu dan juga akan disimpan dalam tabel database di server. Saat pengguna menerima tanggapan email atas permintaan reset kata sandi, dia mengikuti tautan dengan URL unik. Pengguna kemudian mengisi alamat emailnya lagi, bersama dengan kata sandi barunya. Informasi ini dicari di database, dan string acak dibandingkan dengan yang dari pengguna, bersama dengan alamat emailnya. Jika keduanya cocok, maka kita dapat cukup yakin bahwa orang yang sama yang meminta pengaturan ulang juga mengontrol alamat email tersebut dan mudah-mudahan diberi wewenang untuk melakukan pengaturan ulang kata sandi untuk akun tersebut. Dengan asumsi semuanya diperiksa, kata sandi diatur ulang dan pengguna dapat segera masuk dengan kata sandi baru. Kita membangun fungsi ini selanjutnya.

Membangun basis data pengaturan ulang kata sandi

Tabel database untuk pengaturan ulang kata sandi akan menyimpan karakter acak unik untuk URL, ID alamat email yang sedang diatur ulang, tanggal permintaan pengaturan ulang

diterima, dan apakah permintaan pengaturan ulang aktif. Pernyataan CREATE terlihat seperti ini:

```
CREATE TABLE resetPassword (
 id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
 email_id INT,
 pass_key VARCHAR(255),
 date_created DATETIME,
 status VARCHAR(255)
);
```

Bidang status mungkin digunakan di kemudian hari untuk menyetel permintaan atur ulang lama menjadi tidak aktif. Perhatikan bahwa bidang email_id adalah tipe INT. ID unik dari tabel Pelanggan akan digunakan di sini, bukan alamat email yang sebenarnya. Melakukannya akan menghemat ruang disk dan menjaga integritas data pada saat yang sama.Tabel ini harus dibuat sebelum melanjutkan.

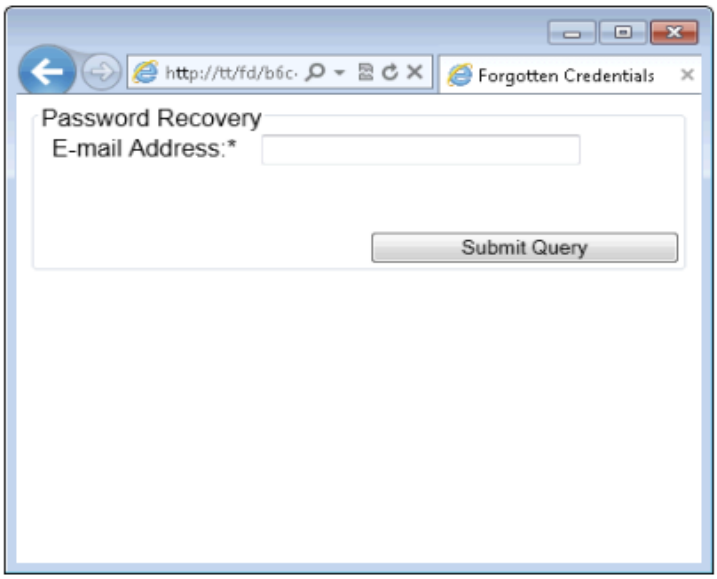
Membangun halaman pemulihan kata sandi

Halaman pemulihan kata sandi pertama adalah formulir sederhana yang hanya berisi satu bidang: alamat email. Formulir mengirimkan POST ke file bernama emailprocess.php, mengikuti pola yang digunakan di seluruh bab. Skrip di bawah ini menunjukkan kode untuk halaman kata sandi email awal.

```
<?php require_once("functions.inc"); ?>
<!doctype html>
<html>
<head>
<script type="text/javascript" src="https://ajax.googleapis.
com/ajax/libs/jquery/1.8.3/jquery.min.js"></script>
<script type="text/javascript" src="email.js"></script>
<link rel="stylesheet" type="text/css" href="form.css">
<title>Forgotten Credentials</title>
</head>
<body>
<form id="emailForm" method="POST" action="email-process.
php">
<div>
<fieldset>
        <legend>Password Recovery</legend>
        <div id="errorDiv">
        <?php
if (isset($_SESSION['error']) && isset($_
SESSION['formAttempt'])) {
        unset($_SESSION['formAttempt']);
        print "Errors encountered<br />\n";
        foreach ($_SESSION['error'] as $error) {
        print $error . "<br />\n";
```

```
        } //end foreach
        } //end if
        ?>
        </div>
        <label for=”email”>E-mail Address:* </label>
        <input type=”text” id=”email” name=”email”>
        <span class=”errorFeedback errorSpan” id=”emailError”>Email is required</span>
        <br />
        <input type=”submit” id=”submit” name=”submit”>
        </fieldset>
</div>
</form>
</body>
</html>
```

Saat dilihat di browser, halaman terlihat seperti gambar ini



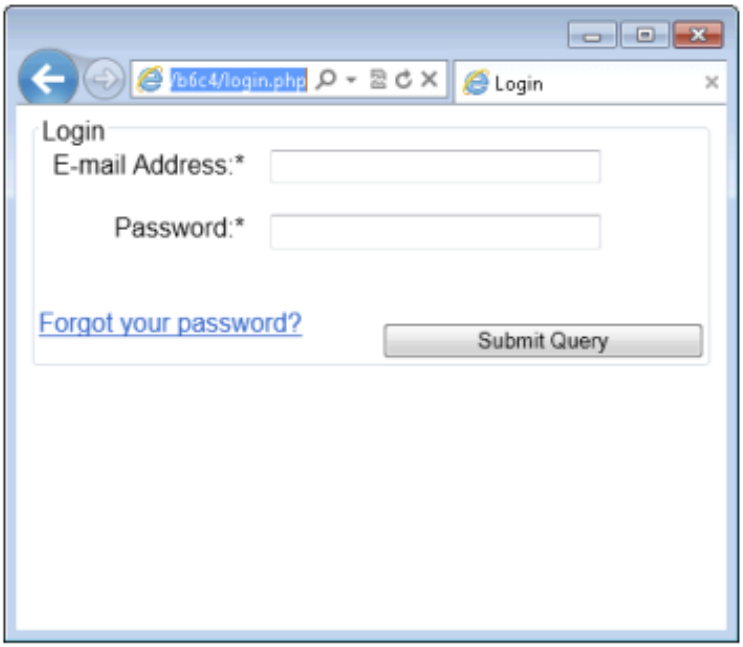
Gambar 5.4 halaman pemulihan password

Menambahkan tautan ke halaman pemulihan kata sandi

Halaman pemulihan kata sandi harus ditautkan dari halaman login, sehingga pengguna bisa sampai di sana dengan mudah. Kode berikut harus ditambahkan ke halaman login.php tepat di atas tag penutup </fieldset>:

```
<br />
<a href=”emailpass.php”>Forgot your password?</a>
```

Gambar dibawah ini menunjukkan halaman yang dihasilkan:



Gambar 5.5 Menambahkan tautan ke halaman lupa password

Membangun halaman pengaturan ulang kata sandi

Formulir pengaturan ulang kata sandi yang sebenarnya berisi bidang untuk alamat email dan kata sandi. Pengguna mengaksesnya ketika mereka mengikuti tautan di email mereka. (Kita akan menunjukkan kode di balik layar itu nanti.) Untuk saat ini, Skrip dibawah ini menunjukkan kode untuk halaman reset password, yang disebut reset.php.

```
<?php
require_once(“functions.inc”);
$invalidAccess = true;
if (isset($_GET[‘user’]) && $_GET[‘user’] != “”) {
        $invalidAccess = false;
        $hash = $_GET[‘user’];
}
//if they’ve attempted the form but had a problem, we need to
allow them in.
if (isset($_SESSION[‘formAttempt’]) && $_SESSION[‘formAttempt’]
== true) {
        $invalidAccess = false;
        $hash = $_SESSION[‘hash’];
}
        if ($invalidAccess) {
        die(header(“Location: login.php”));
}
?>
<!doctype html>
<html>
<head>
<link rel=”stylesheet” type=”text/css” href=”form.css”>
```

```
<title>Reset Password</title>
</head>
<body>
<form id=”loginForm” method=”POST” action=”reset-process.
php”>
<div>
        <fieldset>
        <legend>Reset Password</legend>
        <div id=”errorDiv”>
        <?php
if (isset($_SESSION[‘error’]) && isset($_
SESSION[‘formAttempt’])) {
        unset($_SESSION[‘formAttempt’]);
        print “Errors encountered<br />\n”;
        foreach ($_SESSION[‘error’] as $error) {
        print $error . “<br />\n”;
        } //end foreach
        } //end if
        ?>
        </div>
        <label for=”email”>E-mail Address:* </label>
        <input type=”text” id=”email” name=”email”>
        <span class=”errorFeedback errorSpan” id=”emailError”>Email is required</span>
        <br />
        <label for=”password1”>Password:* </label>
        <input type=”password” id=”password1” name=”password1”>
        <span class=”errorFeedback errorSpan”
id=”password1Error”>Password is required</span>
        <br />
        <label for=”password2”>Password:* </label>
        <input type=”password” id=”password2” name=”password2”>
        <span class=”errorFeedback errorSpan”
id=”password2Error”>Passwords don’t match</span>
        <br />
<?php
        print “<input type=\”hidden\” name=\”hash\”
value=\”{$hash}\”>\n”;
?>
        <input type=”submit” id=”submit” name=”submit”>
        </fieldset>
</div>
</form>
</body>
</html>
```

Kode ini membuat formulir, tetapi sebelum melakukannya terlihat untuk melihat bagaimana pengguna tiba di halaman. Hal pertama yang diperiksa adalah apakah indeks $_GET dari 'user' diatur dan tersedia. Jika demikian, itu berarti pengguna mungkin tiba dengan mengikuti tautan di emailnya. Indeks 'user' berisi nilai unik yang dihasilkan oleh program kita (yang kita lihat nanti).

Jika variabel $_GET['user'] tidak tersedia, lihat selanjutnya untuk melihat apakah pengguna sudah mencoba mengirimkan formulir dan mengalami masalah. Masalahnya mungkin sesederhana kata sandi yang dia masukkan tidak cocok. Apapun, jika dia telah mencoba untuk mengisi formulir, indeks formAttempt dari $_SESSION akan ditetapkan. Jika ya, maka kita mengizinkan pengguna untuk melanjutkan. Jika $_GET['user'] maupun $_SESSION['formAttempt'] tidak tersedia, maka pengguna mungkin tidak boleh berada di sini, jadi kita mengalihkannya. Dengan asumsi bahwa pengguna harus berada di sini dan mengisi formulir dengan benar, kita mengirimkan isi formulir ke file bernama reset-process.php.

Membangun halaman sukses

Seperti halaman lain, jika pengguna mengisi formulir dengan benar, dia akan dikirim ke halaman sukses, kali ini reset-success.php, ditunjukkan pada skrip kode dibawah ini:

```
<!doctype html>
<html>
<head>
<title>Reset Success</title>
</head>
<body>
<div>
        Your password has been reset
</div>
<div>
        <a href=”login.php”>Click here to login</a>
</div>
</body>
</html>
```

Membangun process file

Baik halaman pemulihan kata sandi awal dan halaman pengaturan ulang memiliki file pemrosesan sendiri yang menangani pekerjaan pengiriman email dan pengaturan ulang kata sandi, masing-masing. Sebenarnya, halaman proses memanggil kelas User untuk pekerjaan nyata, tetapi halaman proses berguna untuk validasi dan penanganan logika aturan bisnis. Teruslah membaca untuk instruksi tentang cara membuat kedua file ini.

Membuat file proses pemulihan kata sandi File pemrosesan pemulihan kata sandi, yang disebut email-process.php, ditunjukkan dalam skrip dibawah ini:

```
<?php
require_once(‘functions.inc’);
//prevent access if they haven’t submitted the form.
if (!isset($_POST[‘submit’])) {
        die(header(“Location: login.php”));
```

```
}
$_SESSION['formAttempt'] = true;
if (isset($_SESSION['error'])) {
        unset($_SESSION['error']);
}
$_SESSION['error'] = array();
$required = array("email");
//Check required fields
foreach ($required as $requiredField) {
if (!isset($_POST[$requiredField]) || $_POST
[$requiredField] == "") {
        $_SESSION['error'][] = $requiredField . " is
required.";
        }
}
if (!filter_var($_POST['email'],FILTER_VALIDATE_EMAIL)) {
        $_SESSION['error'][] = "Invalid e-mail address";
}
if (count($_SESSION['error']) > 0) {
        die(header("Location: emailpass.php"));
} else {
        $user = new User;
        if ($user->emailPass($_POST['email'])) {
        unset($_SESSION['formAttempt']);
        die(header("Location: email-success.php"));
        } else {
        $_SESSION['error'][] = "There was a problem locating
        the e-mail address.";
        die(header("Location: emailpass.php"));
        }
}
?>
```

Tidak banyak kerumitan yang terlibat dalam file ini — setidaknya tidak ada yang belum pernah kita lihat beberapa kali. Sebagian besar detail melibatkan logika validasi. Dengan asumsi semuanya valid, kelas Pengguna dipakai dan metode emailPass() dipanggil. Kita membangunnya nanti. Membuat file proses reset.

Membuat reset process file

File proses reset mengikuti pola yang sama seperti file proses email. Daftar dibawah ini menunjukkan kode untuk file proses reset.

```
<?php
require_once('functions.inc');
//prevent access if they haven't submitted the form.
if (!isset($_POST['submit'])) {
        die(header("Location: login.php"));
```

```
}
$_SESSION['formAttempt'] = true;
if (isset($_SESSION['error'])) {
        unset($_SESSION['error']);
}
$_SESSION['error'] = array();
$required = array("email","password1","password2");
//Check required fields
foreach ($required as $requiredField) {
        if (!isset($_POST[$requiredField]) || $_
POST[$requiredField] == "") {
                $_SESSION['error'][] = $requiredField . " is
        required.";
        }
}
if (!filter_var($_POST['email'],FILTER_VALIDATE_EMAIL)) {
 $_SESSION['error'][] = "Invalid e-mail address";
}
if (count($_SESSION['error']) > 0) {
        die(header("Location: reset.php"));
} else {
        $user = new User;
        if ($user->validateReset($_POST)) {
                unset($_SESSION['formAttempt']);
                die(header("Location: reset-success.php"));
 } else {
        if ($user->errorType = "nonfatal") {
                $_SESSION['hash'] = $_POST['hash'];
                $_SESSION['error'][] = "There was a problem with
the form.";
                        die(header("Location: reset.php"));
                } else {
                        $_SESSION['error'][] = "There was a problem with
                the form.";
                        die(header("Location: emailpass.php"));
                }
        }
}
?>
```

Satu item baru dalam file ini adalah konsep tipe kesalahan. Secara khusus, aplikasi sekarang mendefinisikan jenis kesalahan yang ditemui sebagai fatal, yang berarti bahwa pemrosesan tidak boleh dilanjutkan, dan tidak fatal, yang berarti pengguna dapat diperingatkan tentang masalah tersebut dan mungkin memperbaikinya. Kita dapat melihat ini tercermin dalam pemeriksaan errorType dalam kode. Jika ini adalah kesalahan yang tidak fatal, maka kita menyimpan ID unik dalam sesi dan membiarkan pengguna mencoba lagi. Jika

kita melihat apa yang kita yakini sebagai kesalahan fatal, maka kita tidak akan membiarkan pengguna mencoba lagi. Contoh kesalahan fatal mungkin sesuatu yang kita deteksi sebagai kemungkinan upaya untuk meretas aplikasi. Kita tidak ingin mengizinkan pengguna untuk melanjutkan dalam kasus itu, dan kita mungkin mengambil tindakan lain, seperti memblokir alamat IP-nya, dan sebagainya. Untuk saat ini, gunakan penunjukan nonfatal dalam file ini dan di dalam kelas Pengguna, yang kita lihat selanjutnya.

Membangun metode kelas

Langkah terakhir dalam proses reset kata sandi adalah membangun fungsi atau metode untuk menangani langkah-langkah yang terlibat. Kita telah membuat halaman dan file pemrosesan, jadi yang tersisa untuk dilakukan adalah menambahkan metode ke kelas Pengguna.

Menambahkan metode email

Metode emailPass, yang dipanggil dari dalam file email-process.php dari daftar skrip selanjutnya, bertanggung jawab untuk mencari alamat email dimasukkan oleh pengguna, menghasilkan hash yang unik, memasukkan informasi tersebut ke dalam database, dan mengirim e-mail instruksi reset kepada pengguna. Abstraksi yang berguna, yang tidak disertakan dalam bab ini, adalah membuat metode untuk setiap tugas tersebut, seperti satu untuk mengembalikan ID pengguna dan yang lainnya untuk menghasilkan hash unik. Daftar dibawah ini menunjukkan metode emailPass, yang harus ditambahkan ke class User.

```
public function emailPass($user) {
        $mysqli = new mysqli(DBHOST,DBUSER,DBPASS,DB);
        if ($mysqli->connect_errno) {
        error_log(“Cannot connect to MySQL: “ .
$mysqli->connect_error);
        return false;
        }
        // first, lookup the user to see if they exist.
        $safeUser = $mysqli->real_escape_string($user);
        $query = “SELECT id,email FROM Customer WHERE email =
        ‘{$safeUser}’”;
        if (!$result = $mysqli->query($query)) {
        $_SESSION[‘error’][] = “Unknown Error”;
        return false;
        }
        if ($result->num_rows == 0) {
        $_SESSION[‘error’][] = “User not found”;
        return false;
}
$row = $result->fetch_assoc();
$id = $row[‘id’];
$hash = uniqid(“”,TRUE);
$safeHash = $mysqli->real_escape_string($hash);
$insertQuery = “INSERT INTO resetPassword (email_
id,pass_key,date_created,status) “ .
                “ VALUES (‘{$id}’,’{$safeHash}’,NOW(),’A’)”;
```

```
                if (!$mysqli->query($insertQuery)) {
                error_log("Problem inserting resetPassword row
        for " . $id);
 $_SESSION['error'][] = "Unknown problem";
 return false;
 }
 $urlHash = urlencode($hash);
 $site = "http://localhost";
 $resetPage = "/reset.php";
 $fullURL = $site . $resetPage . "?user=" . $urlHash;
 //set up things related to the e-mail
 $to = $row['email'];
 $subject = "Password Reset for Site";
 $message = "Password reset requested for this site.\
r\n\r\n";
        $message .= "Please go to this link to reset your
password:\r\n";
        $message .= $fullURL;
        $headers = "From: webmaster@example.com\r\n";
        mail($to,$subject,$message,$headers);
        return true;
 } //end function emailPass
```

Fungsi PHP mail() digunakan dalam metode emailPass. Fungsi bawaan ini menerima empat argumen: tujuan (Kepada) untuk email, subjek email, pesan sebenarnya itu sendiri, dan header tambahan apa pun. Header tambahan tersebut mencakup hal-hal seperti header From: yang biasanya kita lihat di email, tetapi juga dapat menyertakan hal-hal seperti header Reply-To:, serta header CC dan BCC.

Membuat metode validasi

Metode validateReset() dipanggil dari file proses reset dan memiliki tugas memvalidasi semua yang dikirim oleh pengguna untuk permintaan ini dan juga menjalankan tugas menyetel ulang sandi. Daftar skrip dibawah ini menunjukkan metode validateReset(), yang harus ditambahkan ke kelas User:

```
public function validateReset($formInfo) {
 $pass1 = $formInfo['password1'];
 $pass2 = $formInfo['password2'];
 if ($pass1 != $pass2) {
 $this->errorType = "nonfatal";
 $_SESSION['error'][] = "Passwords don't match";
 return false;
 }
 $mysqli = new mysqli(DBHOST,DBUSER,DBPASS,DB);
 if ($mysqli->connect_errno) {
 error_log("Cannot connect to MySQL: " .
$mysqli->connect_error);
```

```
 return false;
 }
 $decodedHash = urldecode($formInfo['hash']);
 $safeEmail = $mysqli->real_escape_
string($formInfo['email']);
 $safeHash = $mysqli->real_escape_
string($decodedHash);
 $query = "SELECT c.id as id, c.email as email FROM
Customer c, resetPassword r WHERE " .
 "r.status = 'A' AND r.pass_key = '{$safeHash}' "
.
 " AND c.email = '{$safeEmail}' " .
 " AND c.id = r.email_id";
 if (!$result = $mysqli->query($query)) {
 $_SESSION['error'][] = "Unknown Error";
 $this->errorType = "fatal";
 error_log("database error: " . $formInfo['email']
. " - " . $formInfo['hash']);
 return false;
 } else if ($result->num_rows == 0) {
$_SESSION['error'][] = "Link not active or user
not found";
 $this->errorType = "fatal";
 error_log("Link not active: " .
$formInfo['email'] . " - " . $formInfo['hash']);
 return false;
 } else {
          $row = $result->fetch_assoc();
          $id = $row['id'];
          if ($this->_resetPass($id,$pass1)) {
                    return true;
               } else {
                    $this->errorType = "nonfatal";
                    $_SESSION['error'][] = "Error resetting
          password";
                    error_log("Error resetting password: " .
          $id);
                    return false;
               }
          }
 } //end function validateReset
```

Metode validasiReset pertama-tama memeriksa untuk melihat apakah kata sandi cocok. Tidak ada gunanya melanjutkan jika tidak. Sebuah query kompleks kemudian dibangun menggunakan informasi yang dimasukkan. Inilah pernyataan SELECT:

```
SELECT c.id as id, c.email as email
 FROM Customer c, resetPassword r
 WHERE
 r.status = 'A'
 AND r.pass_key = '{$safeHash}'
 AND c.email = '{$safeEmail}'
 AND c.id = r.email_id
```

Pernyataan SELECT terlihat untuk mengambil ID dan alamat email dari tabel Pelanggan. Masing-masing bidang tersebut diberi nama alias, yang membuat pengaksesannya secara terprogram sedikit kurang rumit. Tabel User dan resetPassword sendiri disebut sebagai c dan r, masing-masing. Melakukannya membantu mengidentifikasi secara unik bidang apa pun yang mungkin memiliki nama kolom yang sama di setiap tabel. Klausa WHERE mencari status A (Aktif) di tabel resetPassword dan mencari pass_key yang sama dengan yang dikirimkan dari formulir pengguna, bersama dengan alamat email yang sama dengan yang dikirimkan dari formulir pengguna. Terakhir, tabel digabungkan dengan kolom umum mereka, yaitu kolom id tabel Pelanggan dan kolom email_id tabel resetPassword.

BAB 6
PHP PRAKTIS

Bab-bab sebelumnya membahas elemen-elemen bahasa PHP. Bab ini dibangun di atas keterampilan pemrograman baru kita untuk mengajari kita beberapa tugas praktis yang umum tetapi penting. Kita akan mempelajari cara terbaik untuk mengelola penanganan string untuk mencapai kode yang jelas dan ringkas yang ditampilkan di browser web persis seperti yang kita inginkan, termasuk manajemen tanggal dan waktu lanjutan. Kita juga akan mengetahui cara membuat dan memodifikasi file, termasuk yang diunggah oleh pengguna.

6.1 MENGGUNAKAN PRINTF

Kita telah melihat fungsi print dan echo, yang hanya menampilkan teks ke browser. Tetapi fungsi yang jauh lebih kuat, printf, mengontrol format output dengan membiarkan kita menempatkan karakter pemformatan khusus dalam sebuah string. Untuk setiap karakter pemformatan, printf mengharapkan kita untuk memberikan argumen yang akan ditampilkan menggunakan format itu. Misalnya, contoh berikut menggunakan penentu konversi %d untuk menampilkan nilai 3 dalam desimal:

```
printf("There are %d items in your basket", 3);
```

Jika kita mengganti %d dengan %b, nilai 3 akan ditampilkan dalam biner (11). Tabel dibawah menunjukkan penentu konversi yang didukung.

Tabel 6.1 Penentu konversi printf

Specifier	Aksi konversi pada argumentasi arg	Contoh (untuk arg 123)
%	Menampilkan karakter % (tidak diperlukan argumen).	%
b	Tampilkan arg sebagai bilangan bulat biner.	1111011
c	Tampilkan karakter ASCII untuk argumen.	{
d	Tampilkan arg sebagai bilangan bulat desimal bertanda.	123
e	Tampilkan argumen menggunakan notasi ilmiah.	1.23000e+2
f	Tampilkan arg sebagai floating point.	123.000000
o	Tampilkan arg sebagai bilangan bulat oktal.	173
s	Tampilkan arg sebagai string.	123
u	Tampilkan arg sebagai desimal tak bertanda.	123
x	Tampilkan arg dalam heksadesimal huruf kecil.	7b
X	Tampilkan arg dalam heksadesimal huruf besar.	7b

Kita dapat memiliki penentu sebanyak yang kita suka dalam fungsi printf, selama kita memberikan sejumlah argumen yang cocok, dan selama setiap penentu diawali dengan simbol %. Oleh karena itu, kode berikut ini valid, dan akan menampilkan "Nama saya Agus. Saya berusia 33 tahun, yaitu 21 dalam heksadesimal":

```
printf("Namaku adalah %s. usia saya %d tahun, yaitu %X dalam heksadesimal", 'Agus', 33,
33);
```

Jika kita mengabaikan argumen apa pun, kita akan menerima kesalahan penguraian yang memberi tahu kita bahwa tanda kurung siku,), ditemukan secara tidak terduga. Contoh yang lebih praktis dari printf mengatur warna dalam HTML menggunakan desimal. Misalnya, kita tahu kita menginginkan warna yang memiliki nilai triplet 65 merah, 127 hijau, dan 245 biru, tetapi kita sendiri tidak ingin mengonversinya ke heksadesimal. Solusi mudahnya adalah:

```
printf("Hello", 65, 127, 245);
```

Periksa format spesifikasi warna antara apostrof ('') dengan cermat. Pertama datang tanda pound, atau hash, (#) yang diharapkan oleh spesifikasi warna. Kemudian datang tiga penentu format %X, satu untuk setiap nomor Kita. Output yang dihasilkan dari perintah ini adalah:

```
<span style='color:#417FF5'>Hello
```

Biasanya, kita akan merasa nyaman menggunakan variabel atau ekspresi sebagai argumen untuk printf. Misalnya, jika kita menyimpan nilai untuk warna kita dalam tiga variabel $r, $g, dan $b, kita dapat membuat warna yang lebih gelap dengan:

```
printf("Hello", $r-20, $g-20, $b-20);
```

6.2 PENGATURAN PRESISI

Kita tidak hanya dapat menentukan jenis konversi, kita juga dapat mengatur ketepatan hasil yang ditampilkan. Misalnya, jumlah mata uang biasanya ditampilkan hanya dengan dua digit presisi. Namun, setelah perhitungan, nilai mungkin memiliki presisi yang lebih besar dari ini, seperti 123,42 / 12, yang menghasilkan 10,285. Untuk memastikan bahwa nilai tersebut disimpan dengan benar secara internal, tetapi ditampilkan dengan presisi hanya dua digit, kita dapat menyisipkan string ".2" di antara simbol % dan penentu konversi:

```
printf("The result is: $%.2f", 123.42 / 12);
```

Output dari perintah ini adalah:

```
The result is $10.29
```

Tetapi kita sebenarnya memiliki kontrol lebih dari itu, karena kita juga dapat menentukan apakah akan memasukkan output dengan nol atau spasi dengan mengawali specifier dengan nilai-nilai tertentu. Contoh dibawah ini menunjukkan empat kemungkinan kombinasi.

Contoh Pengaturan presisi

```
<?php
    echo "<pre>"; // Enables viewing of the spaces
    // Pad to 15 spaces
    printf("The result is $%15f\n", 123.42 12);
    / Pad to 15 spaces, fill with zeros
    printf("The result is $%015f\n", 123.42 12);
    / Pad to 15 spaces, 2 decimal places precision
    printf("The result is $%15.2f\n", 123.42 12);
    / Pad to 15 spaces, 2 decimal places precision, fill with zero
```

```
    printf("The result is $%015.2f\n", 123.42 12); / Pad to 15 spaces, 2 decimal places
    precision, fill with # symbol printf("The result is $%'#15.2f\n", 123.42 / 12);
?>
```

Output dari contoh ini terlihat seperti ini:

```
The result is $          10.285000
The result is $ 00000010.285000
The result is $            10.29
The result is $000000000010.29
The result is $##########10.29
```

Cara kerjanya sederhana jika kita bergerak dari kanan ke kiri. Perhatikan bahwa:

- Karakter paling kanan adalah penentu konversi. Dalam hal ini adalah f untuk floating point.
- Tepat sebelum penentu konversi, jika ada periode dan angka bersama-sama, maka presisi output ditentukan sebagai nilai angka.
- Terlepas dari apakah ada penentu presisi, jika ada angka, maka itu mewakili jumlah karakter yang harus diisi dengan output. Pada contoh sebelumnya, ini adalah 15 karakter. Jika output sudah sama dengan atau lebih besar dari panjang padding, maka argumen ini diabaikan.
- Parameter paling kiri yang diizinkan setelah simbol % adalah 0, yang diabaikan kecuali jika nilai padding telah ditetapkan, dalam hal ini output diisi dengan nol, bukan spasi. Jika karakter pad selain nol atau spasi diperlukan, kita dapat menggunakan salah satu pilihan kita selama kita mengawalinya dengan tanda kutip tunggal, seperti ini: '#.
- Di sebelah kiri adalah simbol %, yang memulai konversi.

Tabel 6.2 Komponen penentu konversi

Mulai Konversi	Karakter pad	Karakter pad	Tampilan Presisi	KOnversi Specifier	Contoh
%		15		f	10.285000
%	0	15	.2	f	000000000010.29
%	‘#	15	.4	f	########10.2850

String Padding

Kita juga dapat memasukkan string ke panjang yang diperlukan (seperti yang kita bisa dengan angka), memilih karakter pengisi yang berbeda, dan bahkan memilih antara justifikasi kiri dan kanan.

Contoh String Padding

```
<?php
    echo "<pre>"; // Enables viewing of the spaces
    $h = 'Rasmus';
    printf("[%s]\n", $h); // Standard string output
    printf("[%12s]\n", $h); // Right justify with spaces to width 12
    printf("[%-12s]\n", $h); // Left justify with spaces
    printf("[%012s]\n", $h); // Zero padding
    printf("[%'#12s]\n\n", $h); // Use the custom padding character '#'
    $d = 'Rasmus Lerdorf'; // The original creator of PHP
```

```
    printf("[%12.8s]\n", $d); // Right justify, cutoff of 8 characters
    printf("[%-12.12s]\n", $d); // Left justify, cutoff of 12 characters
    printf("[%-'@12.10s]\n", $d); // Left justify, pad '@', cutoff 10 chars
?>
```

Perhatikan bagaimana untuk tujuan tata letak di halaman web, saya telah menggunakan tag HTML <pre> untuk mempertahankan semua spasi dan \n karakter baris baru setelah setiap baris yang akan ditampilkan. Output dari contoh ini adalah sebagai berikut:

```
[Rasmus]
[     Rasmus]
[Rasmus     ]
[000000Rasmus]
[######Rasmus]
[ Rasmus L]
[Rasmus Lerdo]
[Rasmus Ler@@]
```

Saat kita menentukan nilai padding, jika string sudah memiliki panjang yang sama atau lebih besar dari nilai itu akan diabaikan, kecuali jika nilai cutoff diberikan yang memperpendek string kembali menjadi kurang dari nilai padding.

Tabel 6.3 Konversi String Komponen Penentu

Mulai Konversi	Justify left/right	Padding karakter	Jumlah padding karakter	Cutoff	Konversi Specifier	Contoh (menggunakan rasmus)
%					s	[Rasmus]
%	-		10		s	[Rasmus]
%		‘#	8	.4	s	[####Rasm]

Menggunakan printf

Seringkali, kita tidak ingin menampilkan hasil konversi tetapi membutuhkannya untuk digunakan di tempat lain dalam kode Kita. Di sinilah fungsi sprintf masuk. Dengannya, kita dapat mengirim output ke variabel lain daripada ke browser. Kita dapat menggunakannya hanya untuk membuat konversi, seperti pada contoh berikut, yang mengembalikan nilai string heksadesimal untuk grup warna RGB 65, 127, 245 dalam $hexstring:

```
$hexstring = sprintf("%X%X%X",  65, 127, 245);
```

Atau kita mungkin ingin menyimpan output yang siap ditampilkan nanti:

```
$out = sprintf("The result is: $%.2f", 123.42 / 12); echo $out;
```

Fungsi Date dan Time

Untuk melacak tanggal dan waktu, PHP menggunakan timestamp Unix standar, yang merupakan jumlah detik sejak awal 1 Januari 1970. Untuk menentukan timestamp saat ini, kita dapat menggunakan fungsi waktu: echo time(); Karena nilai disimpan sebagai detik, untuk mendapatkan timestamp untuk waktu ini minggu depan, kita akan menggunakan yang berikut ini, yang menambahkan 7 hari kali 24 jam kali 60 menit kali 60 detik ke nilai yang dikembalikan:

```
echo time() + 7 24 60 * 60;
```

Jika kita ingin membuat timestamp untuk tanggal tertentu, kita dapat menggunakan fungsi mktime. Outputnya adalah timestamp 946684800 untuk detik pertama menit pertama jam pertama hari pertama tahun 2000:

```
echo mktime(0, 0, 0, 1, 1, 2000);
```

Parameter untuk lulus adalah, dalam urutan dari kiri ke kanan:

- Jumlah jam (0–23)
- Jumlah menit (0–59)
- Jumlah detik (0–59)
- Nomor bulan (1-12)
- Jumlah hari (1–31)
- Tahun (1970–2038, atau 1901–2038 dengan PHP 5.1.0+ pada sistem bertanda 32-bit)

Catatan: kita mungkin bertanya mengapa kita dibatasi pada tahun 1970 hingga 2038. Ya, itu karena pengembang asli Unix memilih awal tahun 1970 sebagai tanggal dasar yang tidak perlu dilalui oleh programmer sebelumnya! Untungnya, karena (pada versi 5.1.0) PHP mendukung sistem yang menggunakan bilangan bulat 32-bit yang ditandatangani untuk timestamp, tanggal dari 1901 hingga 2038 diperbolehkan. Namun, itu menimbulkan masalah yang lebih buruk daripada yang asli *karena perancang Unix juga memutuskan bahwa tidak ada yang akan menggunakan Unix setelah sekitar 70 tahun atau lebih*, dan karena itu percaya bahwa mereka dapat menyimpan timestamp sebagai nilai 32-bit—yang akan habis pada 19 Januari 2038! Ini akan menciptakan apa yang kemudian dikenal sebagai bug Y2K38 (seperti bug milenium, yang disebabkan oleh penyimpanan tahun sebagai nilai dua digit, dan yang juga harus diperbaiki). PHP memperkenalkan kelas DateTime di versi 5.2 untuk mengatasi masalah ini, tetapi hanya akan berfungsi pada arsitektur 64-bit.

Untuk menampilkan date, gunakan fungsi tanggal, yang mendukung sejumlah besar opsi pemformatan, memungkinkan kita menampilkan tanggal dengan cara apa pun yang kita inginkan. Formatnya adalah sebagai berikut:

```
date($format, $timestamp);
```

Parameter $format harus berupa string yang berisi penentu pemformatan seperti yang dijelaskan dalam Tabel dibawah, dan $timestamp harus berupa timestamp Unix. Untuk daftar lengkap penentu, lihat http://php.net/manual/en/function.date.php. Perintah berikut akan menampilkan tanggal dan waktu saat ini dalam format "Kamis 6 Juli 2017 - 13:38":

```
echo date("l F jS, Y - g:ia", time());
```

Tabel 6.4 Penentu format fungsi tanggal utama

Format Deskripsi		**Pengembalian Nilai**
Hari	**Spesifier**	
d	Hari dalam sebulan, dua digit, dengan awalan nol	01 hingga 31
D	Hari dalam seminggu, tiga huruf	Mon hingga Sun
j	Hari dalam sebulan, tidak ada nol di depan	1 hingga 31
l	Hari dalam seminggu, nama lengkap	Sunday hingga Saturday

N	Hari dalam seminggu, numerik, Senin hingga Minggu	1 hingga 7
S	Akhiran untuk hari dalam sebulan (berguna dengan specifier j)	St, nd, rd atau th
w	Hari dalam seminggu, numerik, Minggu hingga Sabtu	0 hingga 6
z	Hari dalam setahun	0 hingga 365
Spesifier mingguan		
W	Jumlah minggu dalam tahun	01 hingga 52
Spesifier mingguan		
F	Nama bulan	Januari hingga Desember
m	Nomor bulan dengan angka nol di depan	01 hingga 12
M	Nama bulan, tiga huruf	Jan hingga Dec
n	Nomor bulan, tanpa angka nol di depan	1 hingga 12
T	Jumlah hari dalam bulan tertentu	28 hingga 31
Spesifier mingguan		
L	Tahun kabisat	1 = Yae, 0 = Tidak
y	Tahun, 2 digit	00 hingga 99
Y	Tahun, 4 digit	000 hingga 999
Spesifier mingguan		
a	Sebelum atau sesudah tengah hari, huruf kecil	am atau pm
A	Sebelum atau sesudah tengah hari, huruf besar	AM atau PM
g	Jam dalam sehari, format 12 jam, tanpa awalan nol	1 hingga 12
G	Jam dalam sehari, format 24 jam, tanpa awalan nol	00 hingga 23
h	Jam dalam sehari, format 12 jam, dengan angka nol di depan	01 hingga 12
H	Jam dalam sehari, format 24 jam, dengan angka nol di depan	00 hingga 23
i	Menit, dengan nol di depan	00 hingga 59
s	Detik, dengan nol di depan	00 hingga 59

6.3 KONSTANTA TANGGAL

Ada sejumlah konstanta berguna yang dapat kita gunakan dengan perintah tanggal untuk mengembalikan tanggal dalam format tertentu. Misalnya, date(DATE_RSS) mengembalikan tanggal dan waktu saat ini dalam format yang valid untuk umpan RSS. Beberapa konstanta yang lebih umum digunakan adalah:

DATE_ATOM

Ini adalah format untuk feed Atom. Format PHP adalah "Y-m-d\TH:i:sP" dan output contoh adalah "2018-08-16T12:00:00+00:00".

DATE_COOKIE

Ini adalah format cookie yang disetel dari server web atau JavaScript. Format PHP adalah "l, d-M-y H:i:s T" dan contoh outputnya adalah "Kamis, 16- Aug-18 12:00:00 UTC".

DATE_RSS

Ini adalah format untuk umpan RSS. Format PHP adalah "D, d M Y H:i:s O" dan output contoh adalah "Kamis, 16 Agustus 2018 12:00:00 UTC".

DATE_W3C

Ini adalah format untuk World Wide Web Consortium. Format PHP adalah "Y-m-d\TH:i:sP" dan contoh outputnya adalah "2018-08-16T12:00:00+00:00"

Daftar lengkapnya dapat ditemukan di http://php.net/manual/en/class.datetime.php.

Menggunakan checkdate

Kita telah melihat cara menampilkan tanggal yang valid dalam berbagai format. Tetapi bagaimana kita dapat memeriksa apakah pengguna telah mengirimkan tanggal yang valid ke program Kita? Jawabannya adalah meneruskan bulan, hari, dan tahun ke fungsi tanggal pemeriksaan, yang mengembalikan nilai TRUE jika tanggalnya valid, atau FALSE jika tidak. Misalnya, jika tanggal 30 Februari tahun apa pun dimasukkan, itu akan selalu menjadi tanggal yang tidak valid. Contoh dibawah ini menunjukkan kode yang dapat kita gunakan untuk ini. Seperti berdiri, itu akan menemukan tanggal yang diberikan tidak valid.

Contoh Memeriksa validitas checkdate

```
<?php
    $month = 9; // September (only has 30 days)
    $day = 31; // 31st
    $year = 2018; // 2018
    if (checkdate($month, $day, $year)) echo "Date is valid";
    else echo "Date is invalid";
?>
```

6.4 PENANGANAN BERKAS

Sekuat apa pun, MySQL bukan satu-satunya (atau tentu saja yang terbaik) cara untuk menyimpan semua data di server web. Terkadang bisa lebih cepat dan nyaman untuk mengakses file secara langsung di hard disk. Kasus di mana kita mungkin perlu melakukan ini adalah memodifikasi gambar seperti avatar pengguna yang diunggah, atau file log yang ingin kita proses. Namun, pertama, catatan tentang penamaan file: jika kita menulis kode yang mungkin digunakan pada berbagai instalasi PHP, tidak ada cara untuk mengetahui apakah sistem ini peka huruf besar/kecil. Misalnya, nama file Windows dan Mac OS X tidak peka huruf besar-kecil, tetapi Linux dan Unix. Oleh karena itu, kita harus selalu berasumsi bahwa sistem peka huruf besar-kecil dan tetap berpegang pada konvensi seperti semua nama file huruf kecil.

Memeriksa Apakah Ada File

Untuk menentukan apakah file sudah ada, kita dapat menggunakan fungsi file_exists, yang mengembalikan TRUE atau FALSE, dan digunakan seperti ini: if (file_exists("testfile.txt")) echo "File exist";

Membuat File

Pada titik ini, testfile.txt tidak ada, jadi mari kita buat dan tulis beberapa baris ke sana. Ketik Contoh dibawah ini lalu simpan sebagai testfile.php.

Contoh Membuat file teks sederhana

```
<?php // testfile.php
    $fh = fopen("testfile.txt", 'w') or die("Failed to create file");
    $text = <<_END
Line 1
Line 2
```

```
Line 3
_END;
    fwrite($fh, $text) or die("Could not write to file");
    fclose($fh);
    echo " File 'testfile.txt' yang ditulis dengan sukses ";
?>
```

Ketika kita menjalankan ini di browser, semuanya baik-baik saja, kita akan menerima pesan File 'testfile.txt' yang ditulis dengan sukses. Jika kita menerima pesan kesalahan, hard disk kita mungkin penuh atau, kemungkinan besar, kita mungkin tidak memiliki izin untuk membuat atau menulis ke file, dalam hal ini kita harus mengubah atribut folder tujuan sesuai dengan sistem operasi Kita. Jika tidak, file testfile.txt sekarang harus berada di folder yang sama di mana kita menyimpan program testfile.php. Coba buka file dalam editor teks atau program—isinya akan terlihat seperti ini:

```
Line 1
Line 2
Line 3
```

Contoh sederhana ini menunjukkan urutan semua penanganan file:

1. Selalu mulai dengan membuka file. Kita melakukan ini melalui panggilan ke fopen.
2. Kemudian kita dapat memanggil fungsi lain; di sini kita menulis ke file (fwrite), tetapi kita juga dapat membaca dari file yang ada (fread atau fgets) dan melakukan hal-hal lain.
3. Akhiri dengan menutup file (fclose). Meskipun program melakukan ini untuk kita ketika itu berakhir, kita harus membersihkannya sendiri dengan menutup file setelah kita selesai.

Setiap file yang terbuka membutuhkan sumber daya file agar PHP dapat mengakses dan mengelolanya. Contoh sebelumnya menetapkan variabel $fh (yang saya pilih untuk menangani file handle) ke nilai yang dikembalikan oleh fungsi fopen. Setelah itu, setiap fungsi penanganan file yang mengakses file yang dibuka, seperti fwrite atau fclose, harus melewati $fh sebagai parameter untuk mengidentifikasi file yang sedang diakses. Jangan khawatir tentang konten variabel $fh; ini adalah nomor yang digunakan PHP untuk merujuk ke informasi internal tentang file—Kita cukup meneruskan variabel ke fungsi lain.

Setelah gagal, FALSE akan dikembalikan oleh fopen. Contoh sebelumnya menunjukkan cara sederhana untuk menangkap dan menanggapi kegagalan: FALSE memanggil fungsi die untuk mengakhiri program dan memberikan pesan kesalahan kepada pengguna. Aplikasi web tidak akan pernah dibatalkan dengan cara yang kasar ini (Kita akan membuat halaman web dengan pesan kesalahan), tetapi ini baik-baik saja untuk tujuan pengujian kita.

Perhatikan parameter kedua untuk panggilan fopen. Ini hanyalah karakter w, yang memberi tahu fungsi untuk membuka file untuk ditulis. Fungsi membuat file jika belum ada. Berhati-hatilah saat bermain-main dengan fungsi-fungsi ini: jika file sudah ada, parameter mode w menyebabkan panggilan fopen menghapus konten lama (bahkan jika kita tidak menulis sesuatu yang baru!).

Tabel 6.5 Mode fopen yang didukung

Mode	Aksi	Deskripsi
‘r’	Baca dari awal file	Terbuka untuk membaca saja; tempatkan penunjuk file di awal file. Kembalikan FALSE jika file belum ada.

‘r+’	Baca dari awal file dan izinkan menulis.	Terbuka untuk membaca dan menulis; tempatkan penunjuk file di awal file. Kembalikan FALSE jika file belum ada.
‘w’	Tulis dari file mulai dan potong file.	Terbuka untuk menulis saja; tempatkan penunjuk file di awal file dan potong file menjadi nol. Jika file tidak ada, coba buat.
‘w+’	Tulis dari awal file, potong file, dan izinkan membaca.	Terbuka untuk membaca dan menulis; tempatkan penunjuk file di awal file dan potong file menjadi nol. Jika file tidak ada, coba buat.
‘a’	Tambahkan ke akhir file.	Terbuka untuk menulis saja; tempatkan penunjuk file di akhir file. Jika file tidak ada, coba buat.
‘a+’		Terbuka untuk membaca dan menulis; tempatkan penunjuk file di akhir file. Jika file tidak ada, coba buat.

Membaca dari File

Cara termudah untuk membaca dari file teks adalah dengan mengambil seluruh baris melalui fgets (pikirkan s terakhir sebagai singkatan dari string).

Contoh Membaca file dengan fgets

```
<?php
    $fh = fopen("testfile.txt", 'r') or
    die("File does not exist or you lack permission to open it");
    $line = fgets($fh);
    fclose($fh);
    echo $line;
?>
```

Jika kita membuat file seperti yang ditunjukkan pada Contoh membuat file teks simpel, kita akan mendapatkan baris pertama:

```
Line 1
```

Contoh Membaca file dengan fread

```
<?php
    $fh = fopen("testfile.txt", 'r') or
        die("File does not exist or you lack permission to open it");
    $text = fread($fh, 3);
    fclose($fh);
    echo $text;
?>
```

Saya telah meminta tiga karakter dalam panggilan fread, sehingga program menampilkan yang berikut:

```
Lin
```

Fungsi fread umumnya digunakan dengan data biner. Tetapi jika kita menggunakannya pada data teks yang membentang lebih dari satu baris, ingatlah untuk menghitung karakter baris baru.

copyfile

Mari kita coba fungsi copy PHP untuk membuat tiruan dari *testfile.txt*. Ketik ulang contoh dibawahb ini dan simpan sebagai *copyfile.php,* lalu panggil program di browser Anda.

Contoh Copy file

```
<?php // copyfile.php
    copy('testfile.txt', 'testfile2.txt') or die("Could not copy file");
    echo "File successfully copied to 'testfile2.txt'";
?>
```

Jika kita memeriksa folder kita lagi, kita akan melihat bahwa kita sekarang memiliki file *testfile2.txt* baru di dalamnya. Omong-omong, jika kita tidak ingin program kita keluar karena upaya penyalinan yang gagal, kita dapat mencoba sintaks alternatif dalam Contoh dibawah ini.

Contoh Sintaks alternatif untuk menyalin file

```
<?php // copyfile2.php
      if (!copy('testfile.txt', 'testfile2.txt')) echo "Could not copy file";
      else echo "File successfully copied to 'testfile2.txt'";
?>
```

movefile

Untuk memindahkan file, ganti namanya dengan fungsi rename, seperti yang terlihat pada contoh dibawah ini.

Contoh Memindahkan file

```
<?php // movefile.php
    if (!rename('testfile2.txt', 'testfile2.new'))
        echo "Could not rename file";
    else echo "File successfully renamed to 'testfile2.new'";
?>
```

Kita juga dapat menggunakan fungsi rename pada direktori. Untuk menghindari pesan peringatan, jika file asli tidak ada, kita dapat memanggil fungsi file_exists terlebih dahulu untuk memeriksa.

deletefile

Menghapus file hanyalah masalah menggunakan fungsi unlink untuk menghapusnya dari sistem file.

Contoh Menghapus file

```
<?php // deletefile.php
    if (!unlink('testfile2.new')) echo "Could not delete file";
    else echo "File 'testfile2.new' successfully deleted";
?>
```

Seperti halnya memindahkan file, pesan peringatan akan ditampilkan jika file tidak ada, yang dapat kita hindari dengan menggunakan file_exists untuk terlebih dahulu memeriksa keberadaannya sebelum memanggil unlink.

Catatan: Setiap kali kita mengakses file di hard disk kita secara langsung, kita juga harus selalu memastikan bahwa sistem file kita tidak mungkin disusupi. Misalnya, jika kita menghapus file berdasarkan input pengguna, kita harus benar-benar yakin bahwa itu adalah file yang dapat dihapus dengan aman dan pengguna diizinkan untuk menghapusnya.

Updatefile

Seringkali, kita ingin menambahkan lebih banyak data ke file yang disimpan, yang dapat kita lakukan dengan banyak cara. Kita dapat menggunakan salah satu mode penambahan tulis atau kita cukup membuka file untuk membaca dan menulis dengan salah satu mode lain yang mendukung penulisan, dan memindahkan penunjuk file ke tempat yang benar di dalam file yang ingin kita tulis atau baca. Penunjuk file adalah posisi di dalam file di mana akses file berikutnya akan dilakukan, apakah itu membaca atau menulis. Ini tidak sama dengan pegangan file (seperti yang disimpan dalam variabel $fh), yang berisi detail tentang file yang sedang diakses. Kita dapat melihat updatefile ini dengan mengetik ulang contoh dibawah ini lalu menyimpannya sebagai update.php. Kemudian panggil di browser Kita.

Contoh Memperbarui file

```
<?php // update.php
    $fh = fopen("testfile.txt", 'r+') or die("Failed to open file");
    $text = fgets($fh);
    fseek($fh, 0, SEEK_END);
    fwrite($fh, "$text") or die("Could not write to file");
    fclose($fh);
    echo "File 'testfile.txt' successfully updated";
?>
```

Program ini membuka testfile.txt untuk membaca dan menulis dengan menyetel mode dengan 'r+', yang menempatkan penunjuk file tepat di awal. Kemudian menggunakan fungsi fgets untuk membaca dalam satu baris dari file (hingga feed baris pertama). Setelah itu, fungsi fseek dipanggil untuk memindahkan penunjuk file ke kanan ke akhir file, di mana baris teks yang diekstraksi dari awal file (disimpan dalam $text) kemudian ditambahkan ke akhir file dan file ditutup. File yang dihasilkan sekarang terlihat seperti ini:

```
Line 1
Line 2
Line 3
Line 1
```

Baris pertama telah berhasil disalin dan kemudian ditambahkan ke akhir file. Seperti yang digunakan di sini, selain pegangan file $fh, fungsi fseek melewati dua parameter lainnya, 0 dan SEEK_END. SEEK_END memberi tahu fungsi untuk memindahkan penunjuk file ke akhir file dan 0 memberi tahunya berapa banyak posisi yang harus dipindahkan ke belakang dari titik itu. Dalam kasus contoh *update file*, nilai 0 digunakan, karena penunjuk diperlukan untuk tetap berada di akhir file.

Ada dua opsi pencarian lain yang tersedia untuk fungsi fseek: SEEK_SET dan SEEK_CUR. Opsi SEEK_SET memberi tahu fungsi untuk mengatur penunjuk file ke posisi tepat yang

diberikan oleh parameter sebelumnya. Jadi, contoh berikut memindahkan penunjuk file ke posisi 18:

```
fseek($fh, 18, SEEK_SET);
```

SEEK_CUR menyetel penunjuk file ke posisi saat ini ditambah nilai offset yang diberikan. Oleh karena itu, jika penunjuk file saat ini berada di posisi 18, panggilan berikut akan memindahkannya ke posisi 23:

```
fseek($fh, 5, SEEK_CUR);
```

Meskipun ini tidak disarankan kecuali kita memiliki alasan yang sangat spesifik untuk itu, bahkan dimungkinkan untuk menggunakan file teks seperti ini (tetapi dengan panjang garis tetap) sebagai database file datar sederhana. Program kita kemudian dapat menggunakan fseek untuk bergerak maju mundur dalam file tersebut untuk mengambil, memperbarui, dan menambahkan catatan baru. Kita juga dapat menghapus rekaman dengan menimpanya dengan nol karakter, dan seterusnya.

6.5 MENGUNCI FILE UNTUK BEBERAPA AKSES

Program web sering dipanggil oleh banyak pengguna pada saat yang bersamaan. Jika lebih dari satu orang mencoba menulis ke file secara bersamaan, itu bisa menjadi rusak. Dan jika satu orang menulisnya sementara yang lain membaca darinya, file itu baik-baik saja tetapi orang yang membacanya bisa mendapatkan hasil yang aneh. Untuk menangani pengguna simultan, kita harus menggunakan fungsi penguncian file kawanan. Fungsi ini mengantrekan semua permintaan lain untuk mengakses file hingga program kita melepaskan kunci. Jadi, setiap kali program kita menggunakan akses tulis pada file yang dapat diakses secara bersamaan oleh banyak pengguna.

Contoh Memperbarui file dengan penguncian file

```
<?php
    $fh = fopen("testfile.txt", 'r+') or die("Failed to open file");
    $text = fgets($fh);
    if (flock($fh, LOCK_EX))
    {
    fseek($fh, 0, SEEK_END);
    fwrite($fh, "$text") or die("Could not write to file");
    flock($fh, LOCK_UN); }
    fclose($fh);
    echo "File 'testfile.txt' successfully updated";
?>
```

Ada trik untuk mengunci file untuk mempertahankan waktu respons terbaik bagi pengunjung situs web Kita: lakukan secara langsung sebelum kita membuat perubahan pada file, lalu buka kuncinya segera setelahnya. Memiliki file yang terkunci lebih lama dari ini akan memperlambat aplikasi kita secara tidak perlu. Inilah sebabnya mengapa panggilan untuk flock dalam Contoh diatas dilakukan secara langsung sebelum dan sesudah panggilan fwrite. Panggilan pertama ke kawanan menyetel kunci file eksklusif pada file yang dirujuk oleh $fh menggunakan parameter LOCK_EX: flock($fh, LOCK_EX); Mulai saat ini, tidak ada proses lain

yang dapat menulis ke (atau bahkan membaca dari) file hingga kita melepaskan kunci dengan menggunakan parameter LOCK_UN, seperti ini:

```
flock($fh, LOCK_UN);
```

Segera setelah kunci dilepaskan, proses lain kembali diizinkan mengakses file. Ini adalah salah satu alasan mengapa kita harus mencari kembali ke titik yang ingin kita akses dalam sebuah file setiap kali kita perlu membaca atau menulis data, karena proses lain bisa saja mengubah file tersebut sejak akses terakhir. Namun, apakah kita memperhatikan bahwa panggilan untuk meminta kunci eksklusif disarangkan sebagai bagian dari pernyataan if? Ini karena kawanan tidak didukung di semua sistem; dengan demikian, adalah bijaksana untuk memeriksa apakah kita berhasil mengamankan kunci, untuk berjaga-jaga jika kunci tidak dapat diperoleh. Hal lain yang harus kita pertimbangkan adalah bahwa flock adalah apa yang dikenal sebagai kunci penasehat. Ini berarti hanya mengunci proses lain yang memanggil fungsi tersebut. Jika kita memiliki kode apa pun yang langsung masuk dan memodifikasi file tanpa menerapkan penguncian file flock, kode itu akan selalu mengesampingkan penguncian dan dapat merusak file Kita. Omong-omong, menerapkan penguncian file dan kemudian secara tidak sengaja meninggalkannya di satu bagian kode dapat menyebabkan bug yang sangat sulit ditemukan.
Catatan: flock tidak akan bekerja pada NFS dan banyak sistem file jaringan lainnya. Juga, saat menggunakan server multithreaded seperti ISAPI, kita mungkin tidak dapat mengandalkan flock untuk melindungi file dari skrip PHP lain yang berjalan di utas paralel dari instance server yang sama. Selain itu, kawanan tidak didukung pada sistem apa pun yang menggunakan sistem file FAT lama (seperti versi Windows yang lebih lama).

Membaca Seluruh File

Fungsi praktis untuk membaca seluruh file tanpa harus menggunakan pegangan file adalah file_get_contents. Ini sangat mudah digunakan.

***Contoh Menggunakan* file_get_contents**

```
<?php
      echo "<pre>"; // Enables display of line feeds
      echo file_get_contents("testfile.txt");
      echo "</pre>"; // Terminates pre tag
?>
```

Tetapi fungsi ini sebenarnya jauh lebih berguna daripada itu, karena kita juga dapat menggunakannya untuk mengambil file dari server di Internet.

6.6 MENGUNGGAH FILE - UPLOAD

Mengunggah file ke server web adalah subjek yang tampaknya menakutkan bagi banyak orang, tetapi sebenarnya tidak bisa lebih mudah. Yang perlu kita lakukan untuk mengunggah file dari formulir adalah memilih jenis pengkodean khusus yang disebut multipart/form-data, dan browser kita akan menangani sisanya. Untuk melihat cara kerjanya, ketik program pada Contoh dibawah ini dan simpan sebagai upload.php. Saat kita menjalankannya, kita akan melihat formulir di browser kita yang memungkinkan kita mengunggah file pilihan Kita.

Contoh uploader gambar upload.php

```
<?php // upload.php
    echo <<<_END
```

```
<html><head><title>PHP Form Upload</title></head><body>
<form method='post' action='upload.php' enctype='multipart/form-data'>
Select File: <input type=’file’ name=’filename’ size=’10’>
<input type='submit' value='Upload'>
</form>
END;
if ($FILES)
{
    $name = $_FILES['filename']['name'];
    move_uploaded_file($_FILES['filename']['tmp_name'], $name);
    echo "Uploaded image '$name'>”;
}
Echo “</body></html>’;
?>
```

Mari kita periksa program ini satu per satu. Baris pertama dari pernyataan echo multiline memulai dokumen HTML, menampilkan judul, dan kemudian memulai isi dokumen. Selanjutnya kita sampai pada formulir yang memilih metode POST pengiriman formulir, menetapkan target untuk data yang diposting ke program upload.php (program itu sendiri), dan memberi tahu browser web bahwa data yang diposting harus dikodekan melalui tipe konten multipart/form-data. Dengan pengaturan formulir, baris berikutnya menampilkan prompt "Pilih File:" dan kemudian meminta dua input. Permintaan pertama adalah untuk file; ia menggunakan jenis input file, nama nama file, dan bidang input dengan lebar 10 karakter. Input kedua yang diminta hanyalah tombol Submit yang diberi label Upload (yang menggantikan teks tombol default Submit Query). Dan kemudian formulir ditutup.

Program singkat ini menunjukkan teknik umum dalam pemrograman web di mana satu program dipanggil dua kali: sekali saat pengguna pertama kali mengunjungi halaman, dan lagi saat pengguna menekan tombol Kirim. Kode PHP untuk menerima data yang diunggah cukup sederhana, karena semua file yang diunggah ditempatkan ke dalam array sistem asosiatif $_FILES. Oleh karena itu, pemeriksaan cepat untuk melihat apakah $_FILES berisi sesuatu sudah cukup untuk menentukan apakah pengguna telah mengunggah file. Ini dilakukan dengan pernyataan if($_FILES).

Pertama kali pengguna mengunjungi halaman, sebelum mengupload file, $_FILES kosong, sehingga program melewati blok kode ini. Ketika pengguna mengunggah file, program berjalan lagi dan menemukan elemen dalam array $_FILES. Setelah program menyadari bahwa file telah diunggah, nama sebenarnya, seperti yang dibaca dari komputer pengunggah, diambil dan ditempatkan ke dalam variabel $name. Sekarang yang diperlukan hanyalah memindahkan file dari lokasi sementara di mana PHP menyimpan file yang diunggah ke yang lebih permanen. Kita melakukan ini menggunakan fungsi move_uploaded_file, meneruskannya dengan nama asli file, yang dengannya file tersebut disimpan ke direktori saat ini.

Terakhir, gambar yang diunggah ditampilkan dalam tag IMG, dan hasilnya akan terlihat seperti Gambar dibawah ini.

Gambar 6.1 Mengunggah gambar sebagai data formulir

Catatan: Jika kita menjalankan program ini dan kemudian menerima pesan peringatan seperti Izin ditolak untuk pemanggilan fungsi move_uploaded_file, maka kita mungkin tidak memiliki izin yang benar untuk folder tempat program dijalankan.

Menggunakan $_FILES

Lima hal Lima hal disimpan dalam larik $_FILES ketika file diunggah, seperti yang ditunjukkan pada Tabel dibawah ini (di mana file adalah nama bidang unggah file yang disediakan oleh formulir pengiriman). Isi dari array $_FILES disimpan dalam array $_FILES ketika sebuah file diupload, seperti yang ditunjukkan pada:

Tabel 6.6 Isi dari array $_FILES

Elemen array	Konten/isi
$_FILES['file']['name']	Nama file yang diunggah (mis., smiley.jpg)
$_FILES['file']['type']	Jenis konten file (mis., image/jpeg)
$_FILES['file']['size']	Ukuran file dalam byte
$_FILES['file']['tmp_name']	Nama file sementara yang disimpan di server
$_FILES['file']['error']	Kode kesalahan yang dihasilkan dari unggahan file

Tipe konten dulu dikenal sebagai tipe MIME (Multipurpose Internet Mail Extension), tetapi karena penggunaannya kemudian meluas ke seluruh Internet, sekarang sering disebut tipe media Internet. Tabel diatas menunjukkan beberapa tipe yang lebih sering digunakan yang muncul di $_FILES['file']['type'].

6.7 VALIDASI

Saya harap sekarang tidak perlu dikatakan lagi (walaupun saya akan tetap melakukannya) bahwa validasi data formulir adalah yang paling penting, karena kemungkinan pengguna mencoba meretas ke server Kita.

Selain data input yang dibentuk dengan jahat, beberapa hal yang juga harus kita periksa adalah apakah file benar-benar diterima dan, jika demikian, apakah jenis data yang dikirim benar.

Contoh Versi upload.php yang lebih aman

Bagian kode non-HTML telah diperluas dari setengah lusin baris Contoh diatas menjadi lebih dari 20 baris, mulai dari if ($_FILES). Seperti versi sebelumnya, baris if ini memeriksa

apakah ada data yang benar-benar diposting, tetapi sekarang ada yang lain yang cocok di dekat bagian bawah program yang menggemakan pesan ke layar saat tidak ada yang diunggah. Dalam pernyataan if, variabel $name diberi nilai nama file yang diambil dari komputer pengunggah (seperti sebelumnya), tetapi kali ini kita tidak akan bergantung pada pengguna yang telah mengirimkan data yang valid kepada kita. Alih-alih, pernyataan sakelar digunakan untuk memeriksa jenis konten yang diunggah terhadap empat jenis gambar yang didukung program ini. Jika kecocokan dibuat, variabel $ext disetel ke ekstensi file tiga huruf untuk jenis itu. Jika tidak ditemukan kecocokan, file yang diunggah bukan dari jenis yang diterima dan variabel $ext disetel ke string kosong "".

Bagian kode berikutnya kemudian memeriksa variabel $ext untuk melihat apakah variabel tersebut berisi string dan, jika demikian, membuat nama file baru bernama $n dengan gambar nama dasar dan ekstensi yang disimpan di $ext. Ini berarti bahwa program memiliki kendali penuh atas nama file yang akan dibuat, karena hanya dapat berupa salah satu dari image.jpg, image.gif, image.png, atau image.tif.

Aman karena mengetahui bahwa program tersebut tidak disusupi, kode PHP lainnya hampir sama dengan versi sebelumnya. Ini memindahkan gambar sementara yang diunggah ke lokasi barunya dan kemudian menampilkannya, sambil juga menampilkan nama gambar lama dan baru.

Catatan: Jangan khawatir harus menghapus file temporary yang dibuat PHP selama proses upload, karena jika file tersebut belum dipindahkan atau diganti namanya, maka secara otomatis akan terhapus saat program keluar.

Setelah pernyataan if ada pencocokan else, yang dijalankan hanya jika jenis gambar yang tidak didukung telah diunggah, dalam hal ini akan menampilkan pesan kesalahan yang sesuai. Saat kita menulis rutinitas pengunggahan file kita sendiri, saya sangat menyarankan kita untuk menggunakan pendekatan serupa dan memiliki nama dan lokasi yang telah dipilih sebelumnya untuk file yang diunggah. Dengan begitu, tidak ada upaya untuk menambahkan nama path dan data berbahaya lainnya ke variabel yang kita gunakan dapat melewatinya. Jika ini berarti bahwa lebih dari satu pengguna dapat memiliki file yang diunggah dengan nama yang sama, kita dapat mengawali file tersebut dengan nama pengguna mereka, atau menyimpannya ke folder yang dibuat secara individual untuk setiap pengguna. Tetapi jika kita harus menggunakan nama file yang disediakan, kita harus membersihkannya dengan hanya mengizinkan karakter alfanumerik dan titik, yang dapat kita lakukan dengan perintah berikut, menggunakan ekspresi reguler untuk melakukan pencarian dan penggantian pada $name:

```
$name = preg_replace("[^A-Za-z0-9.]", "", $name);
```

Ini hanya menyisakan karakter A–Z, a–z, 0–9, dan titik dalam string $name, dan menghapus yang lainnya.

Lebih baik lagi, untuk memastikan bahwa program kita akan bekerja pada semua sistem, terlepas dari apakah mereka peka huruf besar atau kecil, kita mungkin harus menggunakan perintah berikut sebagai gantinya, yang mengubah semua karakter huruf besar menjadi huruf kecil pada saat yang sama: $name = strtolower(ereg_replace("[^A-Za-z0-9.]", "", $name));

Catatan: Terkadang kita mungkin menemukan jenis media image/pjpeg, yang menunjukkan JPEG progresif, tetapi kita dapat dengan aman menambahkan ini ke kode kita sebagai alias image/jpeg, seperti ini:

```
case 'image/pjpeg':
case 'image/jpeg': $ext = 'jpg'; break;
```

6.8 PANGGILAN SISTEM

Terkadang PHP tidak memiliki fungsi yang kita perlukan untuk melakukan tindakan tertentu, tetapi sistem operasi yang menjalankannya mungkin. Dalam kasus seperti itu, kita dapat menggunakan panggilan sistem exec untuk melakukan pekerjaan itu. Misalnya, untuk melihat isi direktori saat ini dengan cepat, kita dapat menggunakan program seperti Contoh dibawah ini. Jika kita menggunakan sistem Windows, itu akan berjalan seperti menggunakan perintah dir Windows. Di Linux, Unix, atau Mac OS X, beri komentar atau hapus baris pertama dan batalkan komentar pada baris kedua untuk menggunakan perintah sistem ls. Kita mungkin ingin mengetik program ini, menyimpannya sebagai exec.php, dan memanggilnya di browser Kita

Contoh Menjalankan perintah sistem

```
<?php // exec.php
    $cmd = "dir"; // Windows
    // $cmd = "ls"; // Linux, Unix & Mac
    exec(escapeshellcmd($cmd), $output, $status);
    if ($status) echo "Exec command failed";
    else
    {
        echo "<pre>";
        foreach($output as $line) echo htmlspecialchars("$line\n");
        echo "</pre>";
?>
```

Fungsi htmlspecialchars dipanggil untuk mengubah karakter khusus apa pun yang dikembalikan oleh sistem menjadi karakter yang dapat dipahami dan ditampilkan oleh HTML dengan benar, sehingga menyempurnakan output. Tergantung pada sistem yang kita gunakan, hasil menjalankan program ini akan terlihat seperti ini (dari perintah dir Windows):

```
Volume in drive C is Hard Disk
Volume Serial Number is DC63-0E29
Directory of C:\Program Files (x86)\Zend\Apache2\htdocs
09/02/2014 12:03  <DIR>          .
09/02/2014 12:03  <DIR>          ..
28/04/2013 08:30                 5,336 chars.php
12/02/2012 13:08                 1,406 favicon.ico
20/01/2014 12:52                 4,202 index.html
09/02/2014 11:49                 76 info.php
21/03/2013 09:52                 110 test.htm
01/04/2013 13:06                 182,459 test.php
               6 File(s)        193,589 bytes
               9 Dir(s)  1,811,290,472,448 bytes free
```

exec mengambil tiga argumen:

- Perintah itu sendiri (dalam kasus sebelumnya, $cmd)
- Array di mana sistem akan menempatkan output dari perintah (dalam kasus sebelumnya, $output)
- Variabel untuk memuat status panggilan yang dikembalikan (dalam kasus sebelumnya, $status)

Jika diinginkan, kita dapat menghilangkan parameter $output dan $status, tetapi kita tidak akan mengetahui output yang dibuat oleh panggilan atau bahkan apakah itu berhasil diselesaikan. Kita juga harus memperhatikan penggunaan fungsi escapeshellcmd. Ini adalah kebiasaan yang baik untuk selalu menggunakan ini saat mengeluarkan panggilan exec, karena itu membersihkan string perintah, mencegah eksekusi perintah arbitrer, jika kita memberikan input pengguna ke panggilan.

Fungsi panggilan sistem biasanya dinonaktifkan pada host web bersama, karena menimbulkan risiko keamanan. Kita harus selalu mencoba untuk memecahkan masalah kita dalam PHP jika kita bisa, dan pergi ke sistem secara langsung hanya jika benar-benar diperlukan. Juga, masuk ke sistem relatif lambat dan kita perlu mengkodekan dua implementasi jika aplikasi kita diharapkan berjalan di sistem Windows dan Linux/Unix.

6.9 KONFIGURASI PHP

Ketika PHP diinstal, pengaturan default tertentu dipilih. Pengaturan ini didasarkan pada nilai-nilai umum yang digunakan secara luas. Misalnya, pengaturan PHP default mungkin menampilkan kesalahan ke layar tergantung pada sistem. Ada kalanya kita mungkin perlu mengubah pengaturan ini. Untuk melakukannya, kita menggunakan file konfigurasi bernama php.ini. Bab ini membahas php.ini secara lebih rinci dan menunjukkan beberapa perubahan konfigurasi umum yang mungkin kita lakukan pada sistem kita.

Memahami php.ini

Perilaku PHP dikendalikan melalui file inisialisasi yang disebut php.ini. Pengaturan seperti bagaimana sesi ditangani, bagaimana kesalahan ditampilkan, dan modul apa yang tersedia semuanya dikendalikan melalui file php.ini. Lokasi sebenarnya dari file php.ini bervariasi tergantung pada sistem operasi dan bagaimana PHP diinstal.

Bekerja dengan php.ini

File php.ini adalah file teks biasa dan harus diedit dengan editor teks biasa seperti Notepad, Textpad, atau Vi. Praktik yang baik adalah membuat salinan php.ini saat ini sebelum kita mulai mengedit. Melakukannya memudahkan untuk kembali ke salinan asli jika kita menemukan perubahan kita menyebabkan masalah. Saat kita membuat perubahan pada php.ini, kita harus memuat ulang server web Apache untuk mengaktifkan perubahan.

Membuat perubahan di luar php.ini

Perubahan yang kita buat pada php.ini berlaku secara global, ke semua situs di server. Namun, ada kalanya kita ingin menerapkan perubahan ke situs atau halaman individual. Ketika ini terjadi, kita memiliki beberapa opsi, dua di antaranya akan kita bahas di sini.

Menggunakan .htaccess atau konfigurasi Apache

Beberapa sistem memungkinkan kita menggunakan file .htaccess untuk mengatur opsi PHP. Atau, jika kita mengontrol server, kita dapat membuat perubahan tingkat situs dalam wadah Apache VirtualHost. Arahan php_value menerapkan perubahan pada konfigurasi PHP. Misalnya, jika kita memiliki situs yang perlu mengunggah file besar, kita dapat mengatur direktif upload_max_filesize PHP seperti:

```
php_value upload_max_filesize 100M
```

Direktif tidak akan diterapkan di seluruh server, melainkan hanya ke file atau situs tempat direktif php_value berlaku. Saat kita menggunakan file .htaccess, perubahan akan segera

diterapkan. Jika kita membuat perubahan di file konfigurasi Apache, maka server Apache perlu dimuat ulang agar perubahan diterapkan.

Membuat perubahan di PHP

PHP menawarkan dua fungsi terkait konfigurasi yang berguna untuk diskusi ini: ini_get() dan ini_set(). Fungsi ini_get() mengambil nilai saat ini dari arahan konfigurasi yang diberikan, dan ini_set() menetapkan nilainya. *Sebagai contoh:*

```
ini_set('upload_max_filesize','100M');
```

Memahami Perubahan Konfigurasi Umum

Mengubah batas waktu sesi

Saat kita menggunakan sesi untuk aplikasi Kita, data biasanya disimpan dalam file di server (meskipun ini juga dapat dikonfigurasi di php.ini). Sesi dipengaruhi oleh proses pengumpulan sampah yang membersihkan sesi mati, seperti sesi yang belum digunakan selama beberapa menit.

Secara default, proses pengumpulan sampah melihat sesi dengan masa pakai 1.440 detik. Ini berarti bahwa pengguna harus diam selama 1.440 detik, dan pada upaya berikutnya, sesinya mungkin kedaluwarsa atau tidak. Perubahan umum adalah pada proses pengumpulan sampah itu, biasanya untuk memperpanjangnya. Perubahan ini biasanya diterapkan dalam konfigurasi seluruh server tetapi mungkin juga berlaku di tingkat situs. Pengaturan php.ini untuk mengontrol perilaku ini adalah:

```
session.gc_maxlifetime = 1440
```

Mengubah parameter sesi lainnya

Banyak parameter lain dapat diatur untuk mengontrol bagaimana sesi berperilaku. Hal-hal seperti di mana file sesi disimpan di server dan apakah mereka menggunakan cookie tersedia untuk diubah. Beberapa perubahan yang lebih umum termasuk pengaturan domain untuk cookie sesi dan nama sesi. Kedua hal ini biasanya diatur di tingkat situs. Nilai default untuk cookie_domain kosong, seperti yang tercermin di sini:

```
session.name = PHPSESSID
session.cookie_domain =
```

6.10 MENONAKTIFKAN FUNGSI DAN KELAS

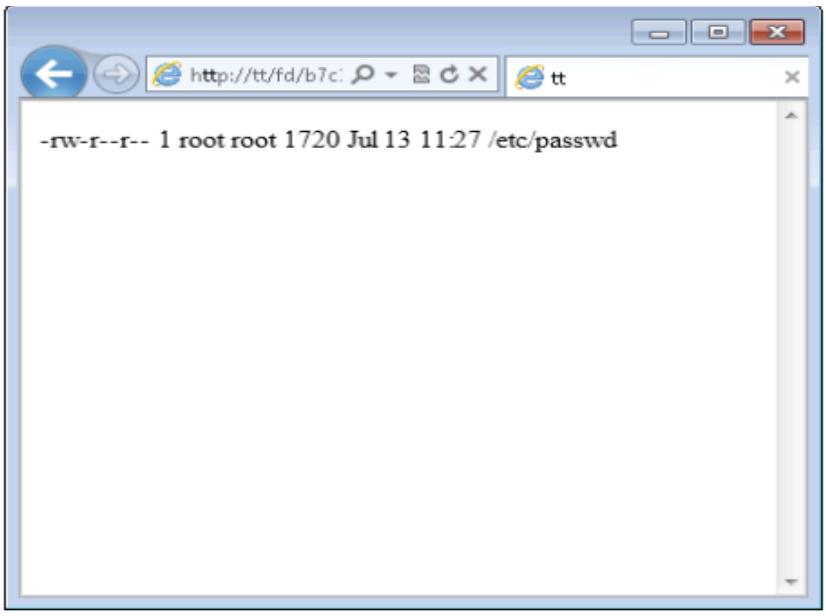
Kita dapat menggunakan php.ini untuk menonaktifkan fungsi atau kelas bawaan. Kita mungkin menemukan bahwa kita tidak ingin orang menggunakan fungsi PHP tertentu atau mungkin ada kerentanan keamanan yang ditemukan dalam fungsi tertentu. Bagaimanapun, kita dapat menonaktifkan fungsi atau kelas menggunakan arahan ini:

```
disable_functions =
disable_classes =
```

Setiap fungsi mengharapkan daftar fungsi atau kelas yang dipisahkan koma untuk dinonaktifkan. Misalnya, kita mungkin ingin menonaktifkan fungsi exec(). Daftar dibawah ini menunjukkan halaman PHP sederhana untuk menguji fungsi ini.

```
<?php
$passwd = exec("ls -la /etc/passwd");
print "{$passwd}<br />\n";
?>
```

Jika dilihat di browser, halaman tersebut terlihat seperti pada gambar dibawah ini.

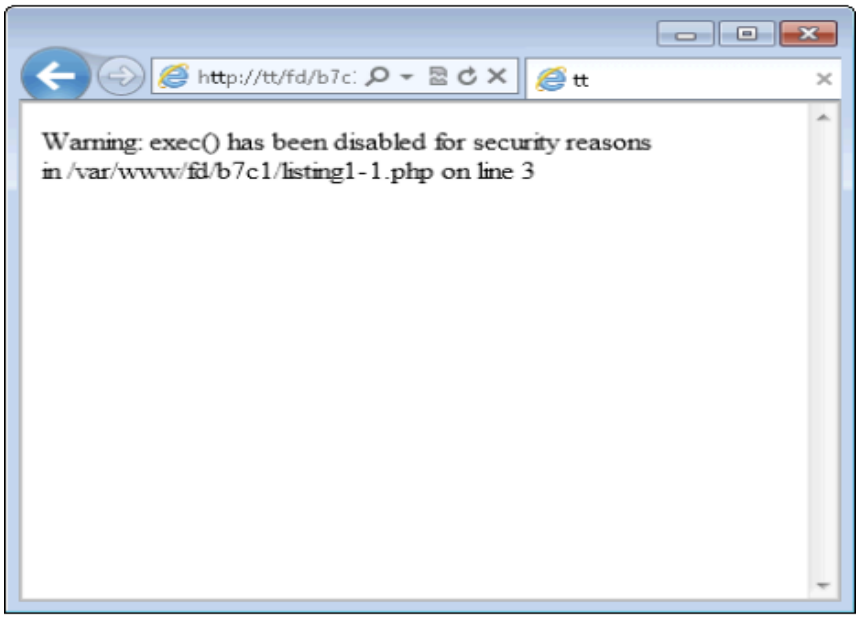


Gambar 6.2 Menggunakan fungsi exec() untuk melihat daftar file.

Mengubah php.ini untuk menonaktifkan fungsi itu berarti menggunakan arahan ini:

disable_functions = exec

Setelah Apache di-restart, perubahan akan berlaku. Memuat ulang halaman sekarang hasil peringatannya seperti pada Gambar dibawah ini:



Gambar 6.3 Fungsi exec() telah dinonaktifkan.

Jika kita menggunakan penyedia hosting, fungsi exec() mungkin sudah dinonaktifkan. Juga, kita mungkin tidak melihat peringatan dari Gambar diatas jika konfigurasi PHP kita tidak menampilkan kesalahan.

6.11 MENGUBAH TAMPILAN KESALAHAN

Ada beberapa arahan konfigurasi di sekitar tampilan kesalahan untuk PHP. Misalnya, server pengembangan kemungkinan akan menampilkan kesalahan setiap saat. Ini diatur dengan direktif display_errors, server produksi kemungkinan tidak akan pernah menampilkan kesalahan kepada pengguna.

Arahan terkait adalah arahan error_reporting. Arahan kompleks ini memberi tahu PHP apa yang harus ditampilkan untuk kesalahan. Kita dapat mengonfigurasi PHP untuk hanya melaporkan kesalahan yang fatal atau kita dapat menampilkan lebih banyak kesalahan kecil seperti pemberitahuan. Arahan error_reporting agak rumit. Lihat http://php.net/error-reporting untuk informasi lebih lanjut jika kita perlu mengubah arahan ini.

6.12 MENGUBAH BATAS SUMBER DAYA

Ada kalanya kita perlu mengubah ukuran file maksimum yang diizinkan, untuk saat file diterima melalui formulir POST atau diunggah langsung atau diterima dengan cara lain sama sekali. Direktif upload_max_filesize menetapkan ukuran file maksimum yang dapat diunggah, sedangkan post_max_sizedirective menetapkan ukuran maksimum formulir POST. Jika kita mengizinkan formulir untuk mengunggah file, kemungkinan kita perlu mengubah kedua arahan. Selain itu, kita mungkin menemukan bahwa kita perlu mengubah batas memori yang dikenakan pada skrip PHP tertentu atau waktu eksekusi skrip berjalan. Misalnya, jika pengguna mengunggah file besar, mungkin perlu beberapa menit. Direktif memory_limit mengatur jumlah memori yang dapat digunakan oleh program PHP, dan direktif max_execution_time mengatur berapa lama program dapat berjalan. Kita dapat mengubah waktu maksimum untuk skrip dengan mengubah max_execution_time di php.ini atau dengan menggunakan fungsi set_time_limit() dalam skrip individual. Fungsi set_time_limit() adalah cara umum untuk memecahkan masalah skrip yang berjalan lama sambil mempertahankan nilai direktif max_execution_time di seluruh server.

6.13 MEMBANGUN SISTEM TEMPELATE PHP

Gambar 6.4 Mengonversi halaman menjadi sistem template

Halaman ini memiliki bagian header yang berisi menu navigasi dengan tautan ke Beranda, Tentang, dan Hubungi Saya. Halaman juga memiliki area konten utama dan footer.

Membuat kelas template

Inti dari sistem template adalah kelas PHP yang bertanggung jawab untuk mengumpulkan berbagai bagian dari halaman tertentu. Kelas Halaman mencakup beberapa metode dan properti. Kita membuat instance kelas Halaman sebagai bagian dari membangun setiap halaman. Ikuti langkah-langkah ini untuk latihan ini:

1. Buka editor teks kita dan buat file kosong baru.
2. Tempatkan kode PHP berikut dalam file:

```
<?php
class Page
{
        public $type = “default”;
        public $title = “My Web Site”;
        public $titleExtra = “”;
} //end Page class
?>
```

3. Simpan file sebagai classPage.php di root dokumen Kita. Lihat bagaimana bagian pertama dari kode ini rusak. Halaman kelas dibuat dan ketiga properti ini dideklarasikan:
    a. Type/Jenis: Ini sesuai dengan jenis halaman yang ditampilkan. Dengan menambahkan properti tipe, kita dapat mengubah perilaku berbagai metode berdasarkan apakah tipe tersebut default atau tipe lain. (Contoh ini hanya memiliki tipe default.)
    b. Title/Judul: Ini muncul di bilah menu browser.
    c. Extra title/Judul tambahan: Gunakan ini untuk halaman tambahan, sehingga halaman dapat memiliki judul yang berbeda.

Membuat bagian atas halaman

Bagian atas halaman adalah salah satu bagian yang lebih kompleks untuk ditangani oleh sistem template. Bagian atas halaman web berisi deklarasi tipe dokumen/*Document type Declaration* (DTD) bersama dengan tautan ke CSS dan JavaScript apa pun yang akan digunakan pada halaman. Bagian atas halaman juga berisi judul dan informasi meta lainnya tentang halaman tersebut. Selain informasi di bagian <head> halaman, bagian atas halaman yang kita gunakan dalam bab ini sebagai contoh juga berisi menu, dengan tautan ke halaman lain di situs.

Kelas yang akan kita buat dalam latihan awal untuk bab ini memiliki empat metode untuk bagian atas halaman, termasuk bagian <head> dan menu. Namun, saat menggunakan kelas, kita tidak ingin harus memanggil (atau ingat untuk memanggil) semua berbagai metode dalam urutan yang benar untuk membuat bagian atas halaman. Yang kita pedulikan hanyalah membuat bagian atas halaman. Oleh karena itu, hanya ada satu metode publik, yang disebut getTop. Metode getTop bertanggung jawab untuk mengumpulkan semua bit untuk membuat seluruh bagian atas halaman.

1. Buka classPage.php jika belum terbuka.

2. Di dalam classPage, tepat di bawah public $titleExtra = “”; baris, masukkan kode berikut:

```
public function getTop() {
        $output = “”;
        $output .= $this->_getDocType();
        $output .= $this->_getHtmlOpen();
        $output .= $this->_getHead();
        $output .= file_get_contents(“pageTop.txt”);
        return $output;
} //end function getTop()
```

3. Simpan classPage.php. Metode getTop() membuat variabel untuk output. Ini memberikan fleksibilitas untuk menambah atau menghapus dari variabel yang kita butuhkan. Metode ini memanggil tiga metode tambahan, mengambil beberapa HTML biasa dari file bernama pageTop.txt, dan mengembalikan hasilnya.
4. Di dalam classPage.php di bawah kurung kurawal penutup metode getTop(), masukkan kode berikut:

```
protected function _getDocType($doctype = “html5”) {
 if ($doctype == “html5”) {
 $dtd = “<!DOCTYPE html>”;
 }
 return $dtd . “\n”;
}
protected function _getHtmlOpen($lang = “en-us”) {
 if ($lang == “en-us”) {
 $htmlopen = “<html lang=\”en\”>”;
 }
 return $htmlopen . “\n”;
}
protected function _getHead() {
 $output = “”;
 $output .= file_get_contents(“pageHead.txt”);
 if ($this->titleExtra != “”) {
 $title = $this->titleExtra . “|” . $this-
>title;
 } else {
 $title = $this->title;
 }
 $output .= “<title>” . $title . “</title>”;
 $output .= “</head>”;
 return $output;
} //end function _getHead()
```

Tiga metode yang kita tambahkan di Langkah 4 bertanggung jawab untuk membangun bagian <head> halaman. Metode pertama, _getDocType, mengembalikan DTD, yang untuk kasus kita adalah HTML5, tetapi bisa berupa jenis dokumen valid lainnya. DTD memberi tahu browser jenis dokumen apa yang diharapkan dan aturan apa yang akan dihormati oleh dokumen itu. Ini membantu browser untuk membuat keputusan tentang cara menampilkan dokumen. Metode berikutnya yang dipanggil adalah _getHtmlOpen(), yang membuat elemen <html> halaman dan menyetel bahasanya. Seperti metode lainnya, bahasa dapat dikustomisasi di sini jika perlu.

Metode terakhir yang dipanggil adalah metode _getHead(). Metode ini menggabungkan file lain, yang disebut pageHead.txt. File pageHead.txt menyertakan tautan ke CSS dan JavaScript. Ingat properti $type yang disetel di Pageclass? Inilah satu tempat di mana kita dapat menggunakannya. Jika kita memiliki jenis halaman khusus yang memerlukan CSS atau JavaScript tambahan, kita dapat menambahkan pernyataan bersyarat di sini seperti, "Jika jenisnya khusus, maka gunakan pageSpecialHead.txt." Metode _getHead() juga merupakan tempat judul halaman mengatur; jika properti $titleExtra disetel, maka properti itu akan digunakan di sini juga.Sekarang kita memiliki kemampuan untuk membangun bagian atas halaman, atau mendekatinya, karena kita masih memerlukan kode untuk kedua file teks tersebut, pageHead.txt dan pageTop.txt. Kita membuatnya menggunakan langkah-langkah berikut.

1. Buat file kosong baru di editor teks Kita.
2. Di dalam file, masukkan markup berikut:
3. Simpan file sebagai pageHead.txt di root dokumen kita dan tahan godaan untuk menutup elemen <head> itu! Elemen <head> dibuka di file ini (meskipun bisa juga dibuka di dalam metode _getHead()). Namun, karena kita perlu menambahkan elemen lain, seperti judul, ke bagian <head>, jangan tutup elemen <head> di file ini. Sebagai gantinya, biarkan metode _getHead() melakukannya. Ini memberi kita fleksibilitas terbesar untuk perubahan dan penambahan nanti.

Sekarang buat file pageTop.txt yang membuat struktur menu dan digabungkan dari metode pageTop().

1. Buat file kosong baru di editor teks Kita.
2. Di dalam file, tambahkan markup berikut:

```
<body>
<div id="menu">
<ul>
<li><a href="home.php">Home</a></li>
<li><a href="about.php">About</a></li>
<li><a href="contact.php">Contact Me</a></li>
</ul>
</div> <!-- end menu -->
```

3. Simpan file sebagai pageTop.txt di root dokumen Kita.

Membuat bagian bawah halaman

Dengan bagian atas halaman dibuat dalam formulir templat, buat bagian bawah dengan mengikuti langkah-langkah ini.

1. Buka classPage.php jika belum dibuka.

2. Di dalam classPage.php, letakkan kode berikut, di bawah kurung kurawal penutup untuk metode _getHead():

```
public function getBottom() {
        return file_get_contents("pageBottom.txt");
 } //end function getBottom()
```

3. Simpan file.

Kode ini hanya mengambil isi file bernama pageBottom.txt. Sekarang adalah waktu yang tepat untuk membuat file itu. Ikuti langkah ini:

1. Buat file kosong baru di dalam editor teks Kita.
2. Di dalam file, tempatkan HTML berikut:

```
<div id="footer">
Copyright (c) 2013 Steve Suehring.
</div> <!-- end footer -->
</body>
</html>
```

3. Simpan file sebagai pageBottom.txt di root dokumen Kita.

Menghubungkan bagian atas, bawah, dan tengah

File classPage.php terakhir akan terlihat seperti dibawah ini:

```
<?php
class Page
{
        public $type = "default";
        public $title = "My Web Site";
        public $titleExtra = "";
        public function getTop() {
                $output = "";
                $output .= $this->_getDocType();
                $output .= $this->_getHtmlOpen();
                $output .= $this->_getHead();
                $output .= file_get_contents("pageTop.txt");
                return $output;
        } //end function getTop()
        protected function _getDocType($doctype = "html5") {
                if ($doctype == "html5") {
                        $dtd = "<!DOCTYPE html>";
        }
        return $dtd . "\n";
 }
 protected function _getHtmlOpen($lang = "en-us") {
 if ($lang == "en-us") {
 $htmlopen = "<html lang=\"en\">";
 }
```

```
return $htmlopen . "\n";
        }
        protected function _getHead() {
                $output = "";
                $output .= file_get_contents("pageHead.txt");
                if ($this->titleExtra != "") {
                $title = $this->titleExtra . "|" . $this->title;
                } else {
                $title = $this->title;
                }
                $output .= "<title>" . $title . "</title>";
                $output .= "</head>";
                return $output;
} //end function _getHead()
public function getBottom() {
return file_get_contents("pageBottom.txt");
} //end function getBottom()
} //end class Page
?>
```

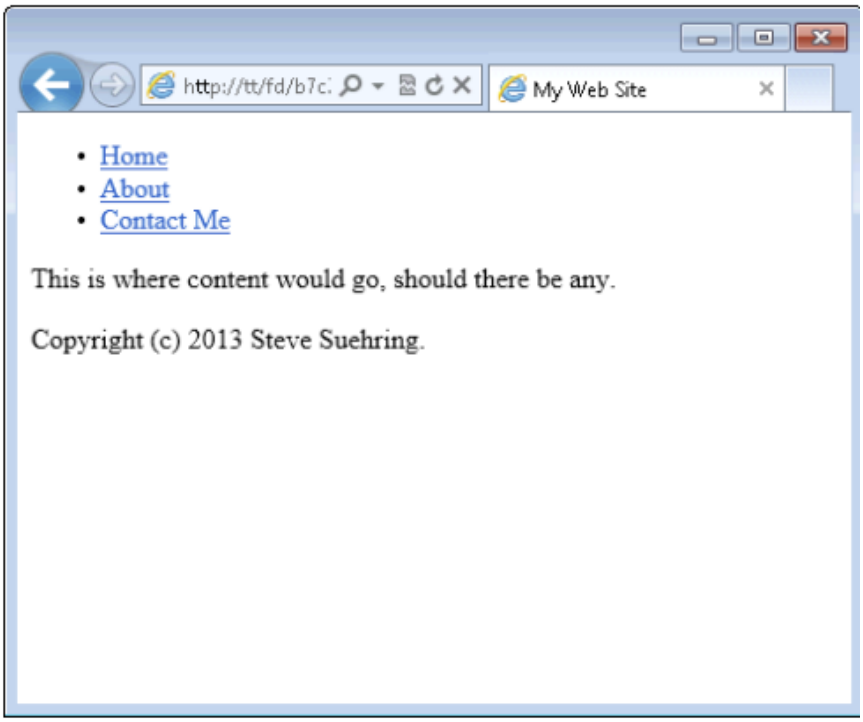
Kita siap membuat halaman dengan sistem templating baru. Ikuti langkah ini:

1. Buat file kosong baru di editor teks kita.
2. Di dalam file, masukkan kode dan HTML berikut:

```
<?php
require_once("classPage.php");
$page = new Page();
print $page->getTop();
print <<<EOF
<div id="mainContent">
<p>This is where content would go, should there be
any.</p>
</div> <!-- end main content -->
EOF;
print $page->getBottom();
?>
```

3. Simpan file sebagai home.php di root dokumen. File ini membuat instance baru dari kelas Page dan kemudian memanggil metode getTop(). Setelah itu selesai, halaman yang sedang dibangun akan memiliki semua yang dibutuhkan hingga ke area konten utama. Area konten utama disediakan dalam file ini dan dilambangkan dengan print pernyataan <<< heredoc EOF. Jenis pernyataan ini memberi tahu PHP untuk hanya menampilkan apa pun yang mengikuti sampai ia melihat EOF penutup, yang muncul di barisnya sendiri, rata kiri. Terakhir, metode getBottom() dipanggil untuk melengkapi halaman. Saatnya untuk melihat halaman. Buka browser web kita dan arahkan ke

http://localhost/home.php. Jika dilihat di web browser, halaman tersebut terlihat seperti pada gambar dibawah ini.



Gambar 6.5 Halaman, dilayani dari sistem templating

Memperluas Template

Dengan halaman pertama yang dibuat, kita dapat mengalihkan perhatian kita ke halaman lain untuk situs tersebut. Halaman yang kita buat terhubung ke dua halaman lain, Tentang dan Hubungi Saya, jadi sekarang saatnya untuk membangun keduanya.

Membangun halaman Tentang

Membangun halaman Tentang adalah masalah sederhana membuat file baru, membuat instance kelas Halaman, dan menambahkan konten. Ikuti langkah ini:

1. Buat file kosong baru di editor kita.
2. Dalam file tersebut, tempatkan kode berikut:

```
<?php
require_once(“classPage.php”);
$page = new Page();
$page->titleExtra = “About”;
print $page->getTop();
print <<<EOF
<div id=”mainContent”>
<p>It’s all about me.</p>
</div> <!-- end main content -->
EOF;
print $page->getBottom();
?>
```

3. Simpan file sebagai about.php di root dokumen Kita.

4. Lihat halaman di browser kita dengan membuka http://localhost/about.php.

Gambar 6.6 Membuat halaman baru dengan sistem templating.

Melihat kode yang kita buat dalam latihan ini, perhatikan bahwa kode itu mirip dengan kode untuk halaman Beranda. Satu-satunya perubahan adalah mengatur properti titleExtra dan mengubah konten HTML halaman yang sebenarnya. Itulah keindahan sistem templating: kita sekarang dapat membuat banyak, banyak halaman, dengan cepat dan mudah. Jika kita perlu mengubah sesuatu atau menambahkan item menu baru, kita dapat melakukannya di satu lokasi dan itu akan diperbarui secara otomatis dan instan di semua halaman.

Membangun halaman Kontak

Halaman kontak untuk situs web terkadang menyertakan elemen lain, mungkin bentuk atau cara lain untuk berinteraksi. Ini berarti kita mungkin perlu menyertakan file JavaScript lain atau CSS yang berbeda. Untungnya, kita dapat melakukannya dengan memperluas kelas templating dan menggunakan properti tipe. Ikuti langkah-langkah ini untuk membuat halaman Kontak:

1. Buka classPage.php.
2. Di dalam metode _getHead(), tambahkan kondisi untuk tipe halaman baru. Seluruh metode _getHead akan terlihat seperti ini:

```
protected function _getHead() {
 $output = “”;
 if ($this->type == “contact”) {
 $output .= file_get_contents(“pageHeadContact.
txt”);
        } else {
        $output .= file_get_contents(“pageHead.txt”);
        }
        if ($this->titleExtra != “”) {
$title = $this->titleExtra . “|” . $this->title;
        } else {
        $title = $this->title;
```

```
            }
            $output .= “<title>” . $title . “</title>”;
            $output .= “</head>”;
     return $output;
    } //end function _getHead()
```

Kode ini memeriksa untuk melihat apakah properti type ($this->type) diatur ke contact. Jika ya, maka file bagian <head> baru disertakan. Jika tidak, bagian <head> normal disertakan.

3. Simpan classPage.php.
4. Buat file kosong baru di editor teks Kita.
5. Di dalam file, tambahkan markup berikut:

```
<head>
<link rel=”stylesheet” href=”style.css” type=”text/css”
/>
<link rel=”stylesheet” href=”contact.css” type=”text/
css” />
<script type=”text/javascript” src=”https://ajax.
googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.
js”></script>
```

6. Simpan file sebagai pageHeadContact.txt di root dokumen Kita.
7. Buat file kosong baru di editor teks Kita.
8. Di dalam file tersebut, letakkan CSS berikut:

```
.contactMethod {
 font-style: italic;
 font-weight: bold;
}
```

9. Simpan file sebagai contact.css di root dokumen Kita.
10. Buat file kosong baru di editor Kita.
11. Di dalam file, tempatkan kode berikut dan HTML:

```
<?php
require_once(“classPage.php”);
$page = new Page();
$page->type = “contact”;
$page->titleExtra = “Contact Me”;
print $page->getTop();
print <<<EOF
<div id=”mainContent”>
<h1>Contacting me is easy</h1>
<p class=”contactMethod”>suehring@braingia.com</p>
<p class=”contactMethod”>Twitter: @stevesuehring</p>
</div> <!-- end main content -->
```

```
EOF;
print $page->getBottom();
?>
```

12. Simpan file sebagai contact.php di root dokumen Kita.
13. Lihat file di browser Kita
14. Klik setiap tautan: Beranda, Tentang, dan Hubungi Saya. Halaman-halaman tersebut harus berfungsi dan terhubung satu sama lain.

DAFTAR PUSTAKA

Abdul Kadir, 2002, Dasar Pemrograman Web Dinamis Menggunakan. PHP, Andi Yogyakarta.

Abdulloh, R. (2018). 7 in 1 Pemrograman Web Untuk Pemula. Jakarta: Elex Media Komputindo.

Doyle Matt (2010). Beginning PHP 5.3, Wiley Publishing, Inc, Indianapolis.

Hidayatulloh, P., & Kawistara, J. K. (2014). Perancangan Web. Bandung: Informatika.

Powers David (2010). PHP Solutions: Dynamic Web Design Made Easy. Second edition. friendsof. United States of America.

Robon Nixon, 2014. Lerning PHP, MySQL, JavaScript, CSS & HTML5, A step by step guide to creating dynamic Websites. Third Edition. O 'Reilly Media, Inc. Canada.

Simarmata, J. (2010). Rekayasa Web. Yogyakarta: Penerbit Andi.

Steve Suehring and Janet Valade, 2013. PHP, MySQL, Javascript & HTML5 All In One for DUMMIES. Published by John Wiley & Sons, Inc., Hoboken, New Jersey.

Suyanto. 2006. Dasar-dasar Perancangan Web Dari Pemula Sampai Mahir. Jakarta : PT. Elex Media Komputindo.

PHP Resource Sites:

http://codewalkers.com

http://developer.yahoo.com/php/

http://easyphp.orghttp://forums.de

http://vshed.com

http://free-php.net

http://hotscripts.com/category/php/

http://htmlgoodies.com/beyond/php/

http://php.net

http://php.resourceindex.com

http://php-editors.com

http://phpbuilder.com

http://phpfreaks.com

http://phpunit.de

http://w3schools.com/php/

http://zend.com

Pengembangan Web PHP

(Hypertext Prepocessor)

Dr. Joseph Teguh Santoso, S.Kom, M.Kom

BIODATA PENULIS

Dr. Joseph Teguh Santoso, S.Kom, M.Kom adalah Rektor dari Universitas Sains & Teknologi Komputer (Universitas STEKOM) Semarang yang memiliki banyak pengalaman praktis dalam bidang *e-commerce*sejak Tahun 2002. Beliau mempunyai 3 (tiga) toko *Official Online Store* di China untuk merek Sepeda Raleigh, dengan omzet tahunan pada Tahun 2019 mencapai lebih dari Rp. 35 Milyar rupiah dan terus meningkat. Dr. Joseph T.S memiliki lisensi tunggal sepeda merek “Raleigh” untuk penjualan *Online* di seluruh China. Di samping itu beliau juga memiliki pabrik sepeda dan sepeda listrik merek “Fengjiu”, yaitu Pabrik Sepeda Listrik yang masih tergolong kecil di China. Pengalaman beliau malang melintang di dunia *online store* di China seperti Alibaba, Tmall, Taobao, JD, Aliexpress sangat membantu mahasiswa untuk memiliki pengalaman teknis dan praktis untuk membuka toko *online* bersama beliau.

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

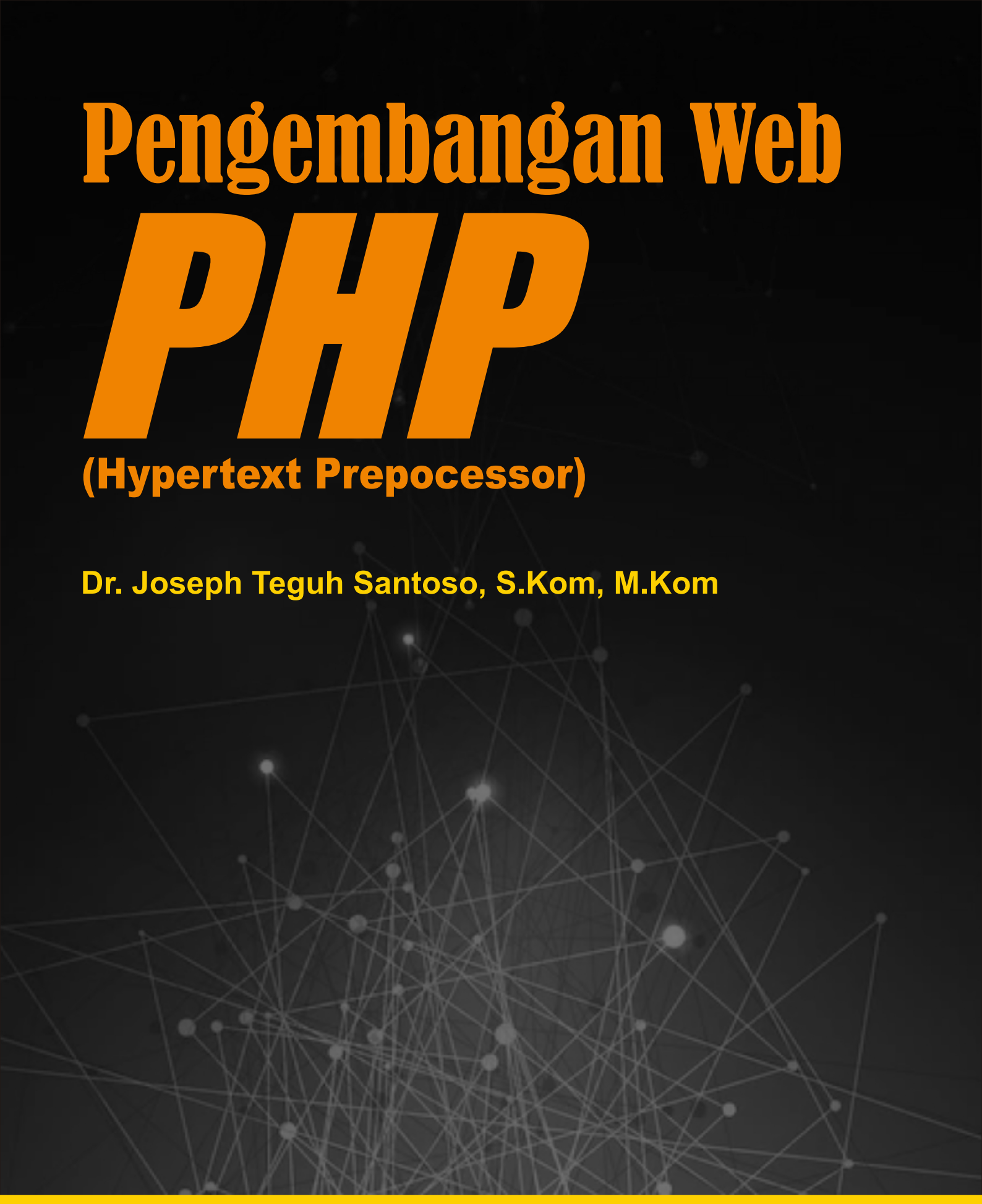
ISBN 978-623-5734-89-7 (PDF)

9 786235 734897

Pengembangan Web PHP

(Hypertext Prepocessor)

Dr. Joseph Teguh Santoso, S.Kom, M.Kom



YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id