



Andre Pratama

# PHP Uncover

---

Panduan Belajar PHP untuk Pemula



DuniaIlkom

# **PHP Uncover**

Panduan Belajar PHP untuk Pemula

DuniaIlkom

© 2015 - 2016 DuniaIlkom

# Contents

<b>Ucapan Terimakasih</b>	<b>i</b>
<b>Tentang Penulis</b>	<b>ii</b>
<b>Kata Pengantar</b>	<b>iii</b>
<b>Asumsi / Pengetahuan Dasar</b>	<b>v</b>
<b>Contoh Kode Program</b>	<b>vi</b>
<b>1. Berkenalan Dengan PHP</b>	<b>1</b>
1.1 Pengertian PHP	1
1.2 PHP Sebagai Server Side Programming Language	5
1.3 Cara Kerja PHP dan Web Server	7
<b>2. Sejarah dan Perkembangan PHP</b>	<b>10</b>
2.1 Sejarah PHP	10
2.2 Logo dan Maskot PHP	14
2.3 Kepopuleran PHP	14
2.4 Framework PHP dan CMS	16
<b>3. Instalasi XAMPP, Web Browser dan Text Editor</b>	<b>17</b>
3.1 Mengenal Aplikasi XAMPP	17
3.2 Instalasi XAMPP	19
3.3 XAMPP Control Panel	24
3.4 XAMPP Troubleshooting	26
3.5 Melihat Folder Instalasi XAMPP	29
3.6 Instalasi Web Browser	31
3.7 Instalasi Text Editor	31
3.8 Text Editor Alternatif	33
<b>4. Menjalankan File PHP</b>	<b>35</b>
4.1 Menampilkan Teks Hello World dengan PHP	35
4.2 PHP Untuk Menghasilkan HTML	39
4.3 Cara PHP diproses di dalam Web Server	41
4.4 Menghasilkan HTML Dinamis dengan PHP	42
4.5 PHP di dalam HTML, atau HTML di dalam PHP?	44
4.6 Mempersiapkan Folder belajar_php	45

## CONTENTS

<b>5. Aturan Dasar Penulisan Kode PHP . . . . .</b>	<b>48</b>
5.1 Extension File PHP . . . . .	48
5.2 Mengenal PHP Tag . . . . .	48
5.3 Perintah echo dan print . . . . .	50
5.4 Statement PHP . . . . .	51
5.5 Case Sensitivity . . . . .	51
5.6 Whitespace . . . . .	52
5.7 Baris Komentar . . . . .	53
5.8 Fungsi phpinfo() . . . . .	55
5.9 Mengatasi Error . . . . .	56
<b>6. Variabel dan Konstanta . . . . .</b>	<b>60</b>
6.1 Pengertian Variabel . . . . .	60
6.2 Aturan Penulisan Variabel PHP . . . . .	60
6.3 Pengertian Konstanta . . . . .	62
6.4 Aturan Penulisan Konstanta PHP . . . . .	62
6.5 Predefined Variable dan Predefined Constant . . . . .	63
6.6 Cara Menampilkan Variabel dan Konstanta . . . . .	64
<b>7. Tipe Data PHP . . . . .</b>	<b>67</b>
7.1 Jenis-Jenis Tipe Data di dalam PHP . . . . .	67
7.2 PHP sebagai Typeless Programming Language . . . . .	68
7.3 Cara Memeriksa Tipe Data dengan var_dump() . . . . .	69
7.4 Tipe Data Integer . . . . .	70
7.5 Tipe Data Float . . . . .	72
7.6 Tipe Data String . . . . .	74
7.7 Tipe Data Boolean . . . . .	81
7.8 Tipe Data Array . . . . .	81
7.9 Tipe Data Object . . . . .	89
7.10 Tipe Data Resource . . . . .	90
7.11 Tipe Data NULL . . . . .	91
7.12 Type Casting . . . . .	92
<b>8. Operator PHP . . . . .</b>	<b>99</b>
8.1 Pengertian Operator dan Operand . . . . .	99
8.2 Jenis-Jenis Operator PHP . . . . .	99
8.3 Operator Aritmatika . . . . .	100
8.4 Operator Increment dan Decrement . . . . .	102
8.5 Operator Perbandingan . . . . .	103
8.6 Operator Logika . . . . .	107
8.7 Operator String . . . . .	109
8.8 Operator Bitwise . . . . .	110
8.9 Operator Assignment . . . . .	113
8.10 Operator Error Control . . . . .	117
8.11 Urutan Operator dalam PHP . . . . .	118
8.12 Type Juggling . . . . .	120

## CONTENTS

<b>9. Struktur Control Pemrograman PHP . . . . .</b>	<b>122</b>
9.1    Struktur Logika IF . . . . .	122
9.2    Struktur Logika ELSE dan ELSE IF . . . . .	125
9.3    Latihan IF, ELSE, dan ELSE IF . . . . .	127
9.4    Struktur Logika SWITCH . . . . .	131
9.5    Operator Conditional . . . . .	136
9.6    Struktur Perulangan FOR . . . . .	138
9.7    Struktur Perulangan WHILE . . . . .	146
9.8    Struktur Perulangan DO WHILE . . . . .	150
9.9    Struktur Perulangan FOREACH . . . . .	151
9.10    Fungsi Keyword Break dan Continue dalam Perulangan . . . . .	153
<b>10. Function . . . . .</b>	<b>160</b>
10.1    Pengertian Function . . . . .	160
10.2    Membuat Function di PHP . . . . .	160
10.3    Memanggil Function . . . . .	161
10.4    Argumen Function . . . . .	162
10.5    Mengembalikan Nilai Function . . . . .	165
10.6    Variable Scope . . . . .	168
10.7    Static Variable . . . . .	170
10.8    Default Argument . . . . .	171
<b>11. PHP Manual . . . . .</b>	<b>175</b>
11.1    Mengenal PHP Manual . . . . .	175
11.2    Download PHP Manual . . . . .	176
11.3    Cara Menggunakan PHP Manual . . . . .	178
11.4    Membaca Format Penulisan Function di PHP Manual . . . . .	181
<b>12. Integer &amp; Float Function PHP . . . . .</b>	<b>185</b>
12.1    Function abs() . . . . .	185
12.2    Function ceil() dan floor() . . . . .	185
12.3    Function round() . . . . .	186
12.4    Function deg2rad() . . . . .	188
12.5    Function sin(), cos(), tan(), acos(), asin(), dan atan() . . . . .	188
12.6    Function pow() . . . . .	189
12.7    Function log() dan log10() . . . . .	189
12.8    Function sqrt() . . . . .	190
12.9    Function min() dan max() . . . . .	191
12.10    Function rand() . . . . .	191
<b>13. String Function PHP . . . . .</b>	<b>193</b>
13.1    Function strtolower() dan strtoupper() . . . . .	193
13.2    Function ucfirst() dan lcfirst() . . . . .	193
13.3    Function strlen() . . . . .	194
13.4    Function trim(), ltrim() dan rtrim() . . . . .	194
13.5    Function str_pad() . . . . .	197
13.6    Function strip_tags() . . . . .	199

## CONTENTS

13.7	Function str_shuffle() . . . . .	200
13.8	Function number_format() . . . . .	201
13.9	Function substr() . . . . .	201
13.10	Function strpos() . . . . .	203
13.11	Function str_replace() . . . . .	203
13.12	Function explode() . . . . .	204
13.13	Function implode() . . . . .	206
<b>14.</b>	<b>Array Function PHP . . . . .</b>	<b>207</b>
14.1	Function count() . . . . .	207
14.2	Function array_sum() . . . . .	208
14.3	Function sort() . . . . .	209
14.4	Function array_rand() . . . . .	210
14.5	Function shuffle() . . . . .	211
14.6	Function array_push() . . . . .	212
14.7	Function array_pop() . . . . .	215
14.8	Function array_unshift() . . . . .	217
14.9	Function array_shift() . . . . .	218
14.10	Function current(), next(), prev(), end() dan reset() . . . . .	219
14.11	Function in_array() . . . . .	222
14.12	Function array_key_exists() . . . . .	223
14.13	Function array_search() . . . . .	224
14.14	Function list() . . . . .	226
<b>15.</b>	<b>Latihan: Menggabungkan HTML dan PHP . . . . .</b>	<b>228</b>
15.1	Membuat tag HTML dari PHP . . . . .	228
15.2	Membuat tag HTML dari Variabel PHP . . . . .	232
15.3	Membuat tag HTML dari Array PHP . . . . .	233
15.4	Menampilkan Associative Array PHP . . . . .	240
<b>16.</b>	<b>HTML Entity dan Include Function . . . . .</b>	<b>250</b>
16.1	Sekilas tentang HTML Entity . . . . .	250
16.2	Function htmlspecialchars() . . . . .	252
16.3	Function htmlspecialchars_decode() . . . . .	253
16.4	Function htmlentities() . . . . .	254
16.5	Function html_entity_decode() . . . . .	256
16.6	Memecah Halaman HTML . . . . .	258
16.7	Function include() . . . . .	258
16.8	Function include_once() . . . . .	265
16.9	Function required() dan required_once() . . . . .	267
<b>17.</b>	<b>HTTP Header, Output Buffering dan php.ini . . . . .</b>	<b>269</b>
17.1	Pengertian HTTP Header . . . . .	269
17.2	Cara Melihat HTTP Header . . . . .	271
17.3	Function header() . . . . .	273
17.4	Redirect dengan fungsi header() . . . . .	275
17.5	Function die() dan exit() . . . . .	278

## CONTENTS

17.6	Output Buffering . . . . .	280
17.7	File Settingan PHP: php.ini . . . . .	283
17.8	Function ob_start() dan ob_end_flush() . . . . .	285
<b>18.</b>	<b>Date and Time Function . . . . .</b>	<b>287</b>
18.1	Function time() . . . . .	287
18.2	Function date() . . . . .	288
18.3	Function date_default_timezone_set() . . . . .	291
18.4	Function mktime() . . . . .	294
18.5	Function getdate() . . . . .	295
18.6	Function strtotime() . . . . .	297
18.7	Case Study: Menghitung Selisih Tanggal . . . . .	299
18.8	DateTime Object PHP . . . . .	304
18.9	Case Study: Refactoring Fungsi cari_selisih_tanggal() . . . . .	307
<b>19.</b>	<b>Magic Constants dan Superglobals Variable . . . . .</b>	<b>310</b>
19.1	Magic Constant PHP . . . . .	310
19.2	Superglobals Variable . . . . .	311
19.3	Superglobals Variable \$GLOBALS . . . . .	312
19.4	Superglobals Variable \$_ENV . . . . .	314
19.5	Superglobals Variable \$_SERVER . . . . .	314
19.6	Superglobals Variable \$_GET . . . . .	319
19.7	Mengirim Pesan Antar Halaman Dengan Query String . . . . .	322
19.8	Function urlencode() dan rawurlencode() . . . . .	325
<b>20.</b>	<b>Form Processing . . . . .</b>	<b>332</b>
20.1	Menampilkan Hasil Form . . . . .	332
20.2	Perbedaan Method GET dan POST . . . . .	335
20.3	Menampilkan Hasil Form Secara Individu . . . . .	335
20.4	Pengertian Validasi Form . . . . .	342
20.5	Function isset() . . . . .	343
20.6	Mencegah Akses ke Halaman proses.php . . . . .	347
20.7	Function empty() . . . . .	351
20.8	Menampilkan Pesan Kesalahan Form . . . . .	354
20.9	Mengisi Kembali Form (form re-populate) . . . . .	357
20.10	Memproses Form Dalam Halaman Yang Sama . . . . .	361
20.11	Proteksi Form dari Cross-Site Scripting dan HTML Injection . . . . .	364
20.12	Membuat Batasan untuk Validasi Form . . . . .	366
20.13	Regular Expression . . . . .	373
20.14	Validasi Form Menggunakan Regular Expression . . . . .	380
20.15	Repopulate Isian Form Select . . . . .	383
20.16	Repopulate Isian Form Checkbox . . . . .	395
20.17	Repopulate Isian Form Radio Button . . . . .	400
20.18	Case Study: Sebuah Form Utuh . . . . .	402
<b>21.</b>	<b>Form Upload . . . . .</b>	<b>421</b>
21.1	Superglobals Variable \$_FILES . . . . .	421

## CONTENTS

21.2	Pengaturan Terkait File Upload (php.ini) . . . . .	426
21.3	Memahami Kode Error File Upload . . . . .	431
21.4	Membatasi Ukuran File Upload dengan MAX_FILE_SIZE . . . . .	434
21.5	Mengatasi Error Warning: POST Content-Length . . . . .	436
21.6	Memindahkan file Upload . . . . .	439
21.7	Validasi untuk File Upload yang Memiliki Nama Sama . . . . .	441
21.8	Membatasi Ukuran File Upload . . . . .	443
21.9	Membatasi Jenis File Upload . . . . .	446
21.10	Multiple Upload File . . . . .	453
21.11	Case study: Buku Tamu DuniaIlkom . . . . .	458
<b>22.</b>	<b>Cookie dan Session . . . . .</b>	<b>468</b>
22.1	Apa itu Cookie? . . . . .	468
22.2	Cara Kerja Cookie . . . . .	470
22.3	Cara Membuat Cookie . . . . .	473
22.4	Superglobals Variable \$_COOKIE . . . . .	476
22.5	Jeda Antara Set Cookie dan Menampilkan Cookie . . . . .	478
22.6	Function setcookie() . . . . .	481
22.7	Jeda Ketika Menghapus Cookie . . . . .	488
22.8	Membuat Multiple Cookie . . . . .	490
22.9	Mengenal Session . . . . .	491
22.10	Prinsip Kerja dan Cara Pembuatan Session . . . . .	492
22.11	Menghapus Session . . . . .	495
22.12	Jeda Antara Set Session dan Menampilkan Session . . . . .	498
22.13	Case Study: Form Login dengan Cookie . . . . .	499
<b>23.</b>	<b>Koneksi PHP dengan MySQL . . . . .</b>	<b>511</b>
23.1	Kenapa Harus MySQL? . . . . .	511
23.2	MySQL atau MariaDB? . . . . .	512
23.3	Mengenal SQL (Structured Query Language) . . . . .	512
23.4	Menjalankan MySQL Server dan MySQL Client . . . . .	512
23.5	Sekilas Tentang Teori Database . . . . .	515
23.6	Mengenal Perintah Dasar SQL di MySQL . . . . .	517
23.7	Mengenal MySQL API (mysql, mysqli, dan PDO) . . . . .	536
23.8	Langkah-langkah Koneksi PHP-MySQL . . . . .	539
23.9	Function mysqli_connect() . . . . .	540
23.10	Function mysqli_connect_errno() dan mysqli_connect_error() . . . . .	542
23.11	Function mysqli_close() . . . . .	545
23.12	Function mysqli_query() . . . . .	546
23.13	Function mysqli_errno() dan mysqli_error() . . . . .	549
23.14	Function mysqli_free_result() . . . . .	551
23.15	Mempersiapkan Tabel Mahasiswa . . . . .	552
23.16	Function mysqli_fetch_rows() . . . . .	554
23.17	Function mysqli_fetch_assoc() . . . . .	559
23.18	Function mysqli_fetch_array() . . . . .	560
23.19	Function mysqli_num_rows() . . . . .	562

## CONTENTS

23.20	Function mysqli_affected_rows() . . . . .	563
23.21	Case Study: Membuat Tabel mahasiswa_baru . . . . .	569
23.22	SQL Injection . . . . .	573
23.23	Function mysqli_real_escape_string() . . . . .	574
23.24	Mengenal Prepared Statement . . . . .	576
23.25	Mysql Extension . . . . .	582
<b>24.</b>	<b>Aplikasi CRUD – Sistem Informasi Kampusku . . . . .</b>	<b>585</b>
24.1	Tampilan Aplikasi Sistem Informasi Kampusku . . . . .	585
24.2	Membuat dan Mengisi Tabel Mahasiswa (generate.php) . . . . .	589
24.3	Mengenal Fungsi Hashing PHP . . . . .	594
24.4	Membuat File Koneksi Global (connection.php) . . . . .	600
24.5	Halaman Login (login.php) . . . . .	601
24.6	Menampilkan Data Mahasiswa (tampil_mahasiswa.php) . . . . .	603
24.7	Menambah Data Baru (tambah_mahasiswa.php) . . . . .	607
24.8	Menghapus Data Mahasiswa (hapus_mahasiswa.php) . . . . .	611
24.9	Edit Data Mahasiswa (edit_mahasiswa.php) . . . . .	615
24.10	Halaman Logout (logout.php) . . . . .	620
24.11	Halaman Utama (index.php) . . . . .	620
24.12	Modifikasi Sistem Informasi Kampusku . . . . .	621
	<b>Penutup: PHP Uncover . . . . .</b>	<b>623</b>

# Ucapan Terimakasih

Dalam kesempatan ini saya ingin mengucapkan terimakasih kepada Allah S.W.T karena dengan karuniaNya saya masih diberi kesempatan dan kesehatan untuk bisa menulis buku ketiga DuniaIlkom: **PHP Uncover**.

Selanjutnya kepada keluarga saya yang terus memberi motivasi dan dukungan tiada henti untuk terus mengembangkan DuniaIlkom.

Terakhir kepada rekan-rekan pembaca dan pengunjung setia DuniaIlkom. Terutama bagi yang telah memberikan donasi untuk membeli buku saya sebelumnya: **HTML Uncover** dan **CSS Uncover**. Karena dari *feedback* dan dukungan rekan-rekan lah saya ‘pede’ lanjut menulis eBook **PHP Uncover** ini. Terimakasih :)

Padang Panjang, 2016

Penulis

Andre Pratama

[www.duniaIlkom.com](http://www.duniaIlkom.com)

# Tentang Penulis

## Andre Pratama



Saat ini memilih karir sebagai praktisi dan penulis di duniailkom.com. Menamatkan kuliah S1 Ilmu Komputer di **Universitas Sumatera Utara** pada tahun 2010. Sempat terjun ke dunia kerja melalui ODP (*Officer Development Program*) di Bank BUMN terbesar di Indonesia. Di sela-sela kesibukan, mendirikan situs duniailkom.com pada tahun 2012.

Karena kecintaan akan menulis dan programming, akhirnya memutuskan untuk keluar dari dunia perbankan dan fokus mengelola duniailkom di akhir tahun 2014. Berusaha untuk menjadikan duniailkom sebagai salah satu media belajar programming dan ilmu komputer terbaik di Indonesia.

Berdomisili di kota Padang Panjang, Sumatera Barat, Andre bisa dihubungi melalui email duniailkom di [duniailkom@gmail.com](mailto:duniailkom@gmail.com), facebook: [facebook.com/belajar.duniailkom](https://facebook.com/belajar.duniailkom)<sup>1</sup>, dan twitter: [twitter.com/duniailkom](https://twitter.com/duniailkom)<sup>2</sup>.

---

<sup>1</sup><https://www.facebook.com/belajar.duniailkom>

<sup>2</sup><https://twitter.com/duniailkom>

# Kata Pengantar

Nyaris tidak ada web programmer yang tidak mengenal **PHP**. **PHP** mungkin lebih populer daripada **HTML**. Selama mengelola dunia ilkom, mayoritas pertanyaan yang diajukan berkaitan dengan PHP, mulai dari cara pemrosesan form, menampilkan dan menyimpan data ke database (**MySQL**), hingga membuat aplikasi PHP seperti Sistem Informasi.

Jika saat ini anda sedang sekolah / kuliah dan belajar tentang web programming, hampir dipastikan PHP-lah yang akan dipelajari. Padahal masih banyak bahasa pemrograman web *server-side* lain seperti ASP, Python, Java, atau Ruby. Hal ini juga menunjukkan bahwa PHP menjadi salah satu bahasa pemrograman yang wajib dikuasai.

Berdasarkan situs [w3techs.com](http://w3techs.com)<sup>3</sup> yang saya akses di awal Agustus 2016, PHP digunakan lebih dari 82% web server di seluruh dunia. Oleh kerena itu, bisa dikatakan bahwa saat ini PHP merupakan bahasa paling populer untuk membuat web (*server-side programming languages*).

Dalam buku **PHP Uncover** ini, saya akan mengajak anda untuk masuk ke **dunia PHP**. Kita akan berkenalan dengan PHP, meng-install XAMPP, membahas cara kerja PHP, mempelajari dasar-dasar PHP, berbagai tipe data dan fungsi-fungsi PHP, hingga cara pemrosesan form, dan membuat koneksi ke database MySQL.

Seperti biasa, meskipun saya berusaha untuk menggunakan penjelasan yang sesederhana mungkin, PHP mungkin terasa sulit. Terutama jika anda belum pernah belajar bahasa pemrograman komputer sebelumnya. Oleh karena itu, jika terdapat penjelasan yang membuat bingung, apalagi membuat anda susah tidur, silahkan langsung layangkan email ke: duniailkom@gmail.com.

*Semoga buku PHP Uncover ini bisa menjadi buku pengantar terbaik dalam langkah anda menjadi seorang web programmer. Sampai jumpa di bab terakhir :)*

---

<sup>3</sup>[http://w3techs.com/technologies/overview/programming\\_language/all](http://w3techs.com/technologies/overview/programming_language/all)

## **Terimakasih untuk tidak memperbanyak / mengedarkan / mencopy eBook ini**

Menulis sebuah buku hingga ratusan halaman butuh waktu yang tidak sebentar. Belum lagi saya harus berjuang mempelajari referensi yang kebanyakan dalam bahasa inggris. Ini saya lakukan agar pembaca bisa mendapatkan materi yang detail, update, dan berkualitas.

Saya menyadari kekurangan sebuah ebook adalah mudah dicopy-paste dan disebarluaskan. Tapi dengan eBook, harga buku bisa ditekan. Selain tidak perlu mencetak, eBook DuniaIlkom ini bisa di dapat dengan murah, termasuk bagi teman-teman di daerah yang ongkos kirimnya lumayan mahal (jika berbentuk buku fisik).

Atas dasar itulah saya mohon kerjasamanya dari rekan-rekan semua **untuk tidak memperbanyak, menggandakan, atau mengupload ulang buku ini di forum, situs dan media lain**.

Saya juga berharap rekan-rekan tidak memposting materi apapun yang ada di dalam buku ini. Jika ingin sebagai bahan artikel untuk postingan blog/situs, silahkan ambil materi yang ada di website duniaIlkom (jangan yang dari buku).

Apabila rekan-rekan memperoleh buku ini **bukan** dari DuniaIlkom, saya mohon bantuan donasinya untuk membeli versi asli. Donasi pembelian buku ini adalah sumber mata pencarian saya untuk menafkahi keluarga. Lisensi atau hak guna buku ini hanya untuk 1 orang, yakni yang telah membeli langsung ke **duniaIlkom@gmail.com**.

Dengan kualitas yang ditawarkan, harga buku ini cukup terjangkau. Buku ini saya buat dengan waktu yang tidak sebentar, hingga berbulan-bulan, dan kadang sampai tengah malam. Bantuan donasi dari rekan-rekan yang membeli buku secara resmi sangat saya hargai, selain mendapat ilmu yang berkah, ini juga bisa menjadi penyemangat saya untuk terus berkarya dan menghadirkan ebook-ebook programming berkualitas lainnya.

Untuk yang membeli dari DuniaIlkom, saya ucapan banyak terimakasih :)

### **Anda diperbolehkan untuk:**

- Mencetak eBook ini untuk keperluan pribadi dan dibaca sendiri.
- Mencopy eBook ini ke laptop/smartphone/laptop milik sendiri.
- Membuat ringkasan buku untuk digunakan sebagai bahan ajar (bukan keseluruan isi buku).

### **Anda tidak dibolehkan untuk:**

- Mencetak eBook ini untuk dibaca oleh orang lain, walaupun gratis.
- Mencopy eBook ini untuk dijual ulang, maupun dibagikan kepada orang lain dengan gratis.
- Membeli buku ini untuk dibaca bersama-sama (lisensi buku ini hanya untuk 1 orang).
- Mengambil sebagian atau seluruh isi buku untuk di publish ke blog, situs, artikel, dan media publik lain.
- Membagikan eBook ini kepada murid/siswa/mahasiswa (jika digunakan untuk bahan pengajaran).

# Asumsi / Pengetahuan Dasar

Sesuai dengan judul buku ini, **PHP Uncover** fokus membahas PHP. Secara sederhana, PHP digunakan untuk men-generate atau menghasilkan kode-kode HTML. Oleh karena itu saya berasumsi rekan-rekan sudah paham dengan HTML dan bisa menjalankan file HTML di web browser.

Dalam buku ini, saya akan langsung masuk kepada PHP tanpa membahas ulang HTML. Jika perlu tutorial singkat tentang HTML, bisa mengunjungi situs duniaIlkom: [Tutorial HTML Dasar](#)<sup>4</sup>. Selain itu saya juga telah merilis eBook [HTML Uncover](#)<sup>5</sup> yang membahas HTML secara mendalam.

Pemahaman tentang CSS juga akan membantu, walaupun tidak diharuskan.

Contoh-contoh kode program dalam buku ini mungkin terlihat ‘apa-adanya’ tanpa warna, animasi, atau efek tampilan yang menarik. Ini semata-mata agar kode program yang ditulis lebih singkat dan simple. Untuk membuatnya lebih menarik, anda bisa menambahkan kode HTML dan CSS sendiri.

Bagi rekan-rekan yang sudah membaca eBook **HTML Uncover** dan **CSS Uncover**, eBook **PHP Uncover** ini merupakan lanjutan paling ‘pas’ sebagai langkah berikutnya untuk menguasai teknologi pembuatan website.

---

<sup>4</sup><http://www.duniaIlkom.com/tutorial-belajar-html-dasar-untuk-pemula/>

<sup>5</sup><http://www.duniaIlkom.com/html-uncover-panduan-belajar-html-lengkap-untuk-pemula/>

# Contoh Kode Program

Seluruh contoh kode program yang ada di buku **PHP Uncover** bisa di download dari folder sharing *Google Drive* yang saya kirim pada saat pembelian: **belajar\_php.zip**.

Sebagaimana yang sudah kita pahami, halaman web lengkap diawali dengan kode HTML seperti `<!DOCTYPE html>` serta tag-tag HTML lain seperti `<head>`, `<title>` dan `<body>`. Namun agar menghemat tempat, dalam buku ini kode pembuka HTML tersebut tidak selalu saya tulis. Di dalam file *belajar\_php.zip* saya kembali melengkapi tag-tag ini.

File kode program saya kelompokkan ke dalam folder sesuai dengan penomoran bab: `bab_06_variable`, `bab_07_tipe_data`, `bab_08_operator`, dst. Nama file juga akan disesuaikan berdasarkan nomor urut dan judul pembahasan.

Penomoran baris pada contoh kode program dalam buku ini (*line numbering*) berguna untuk memudahkan pembahasan. Jika anda ingin men-copy kode ini langsung dari eBook **pdf**, gunakan tombol **ALT + drag** agar nomor-nomor ini tidak ikut terseleksi.

Alternatif yang lebih saya sarankan adalah dengan mengetik ulang seluruh kode program. Tujuannya agar anda lebih cepat paham sekaligus bisa menghafal fungsi dari kode-kode tersebut.



Jika anda mencoba menulis ulang kode program yang ada di buku, dan ternyata tidak jalan, besar kemungkinan terdapat penulisan yang salah.

Di dalam programming, 1 karakter saja yang kurang (apakah itu berupa titik, tanda koma, atau tanda '>'), kode program tidak akan jalan sempurna. Jika ini yang terjadi, silahkan anda samakan dengan file-file dalam folder **belajar\_php.zip** ini.

# 1. Berkenalan Dengan PHP

Sebagai bab pembuka dalam buku **PHP Uncover DuniaIlkom**, saya akan mengajak anda berkenalan dengan PHP.

Kita akan mulai membahas pengertian PHP, memahami maksud dari PHP sebagai *server side programming language*, dan melihat bagaimana PHP bekerja.

## 1.1 Pengertian PHP

Dalam pengertian paling sederhana, PHP adalah *bahasa pemrograman web yang digunakan untuk men-generate atau menghasilkan kode HTML*.

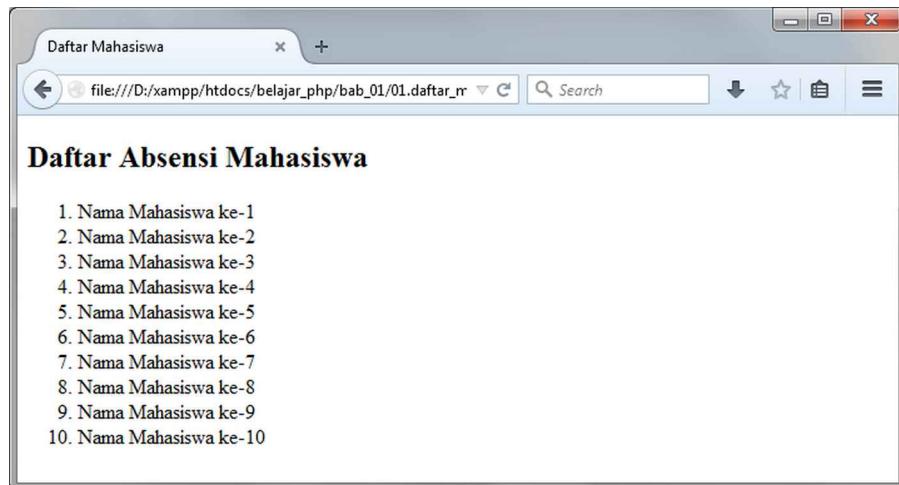
Sebagai contoh, misalkan saya ingin membuat sebuah halaman web yang menampilkan kalimat “Nama Mahasiswa” sebanyak 10 kali. Dengan HTML, saya bisa menggunakan kode berikut:

01.daftar\_mahasiswa.html

---

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Daftar Mahasiswa</title>
6 </head>
7 <body>
8 <h2>Daftar Absensi Mahasiswa</h2>
9   <ol>
10    <li>Nama Mahasiswa ke-1</li>
11    <li>Nama Mahasiswa ke-2</li>
12    <li>Nama Mahasiswa ke-3</li>
13    <li>Nama Mahasiswa ke-4</li>
14    <li>Nama Mahasiswa ke-5</li>
15    <li>Nama Mahasiswa ke-6</li>
16    <li>Nama Mahasiswa ke-7</li>
17    <li>Nama Mahasiswa ke-8</li>
18    <li>Nama Mahasiswa ke-9</li>
19    <li>Nama Mahasiswa ke-10</li>
20  </ol>
21 </body>
22 </html>
```

---



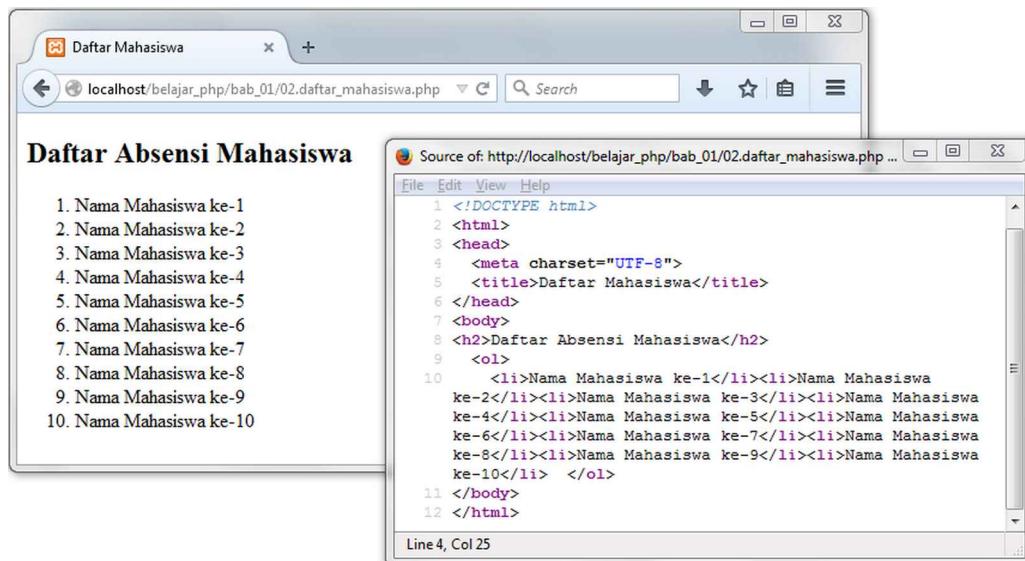
Gambar: Daftar mahasiswa dengan HTML

Menggunakan PHP, tampilan diatas bisa dihasilkan dengan kode berikut:

#### 02.daftar\_mahasiswa.php

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Daftar Mahasiswa</title>
6 </head>
7 <body>
8 <h2>Daftar Absensi Mahasiswa</h2>
9 <ol>
10   <?php
11     for ($i= 1; $i <= 10; $i++) {
12       echo "<li>Nama Mahasiswa ke-$i</li>";
13     }
14   ?>
15 </ol>
16 </body>
17 </html>
```

Baris program antara `<?php` dan `?>` adalah kode PHP. Yang ketika dijalankan, hasilnya sama persis dengan kode HTML sebelumnya. Sebagai pembuktian, saya akan tampilkan source code yang di proses oleh web browser:



Gambar: Daftar mahasiswa dengan PHP + tampilan kode HTML

Terlepas dari susunan tag `<li>` yang sedikit ‘berantakan’, kode yang dihasilkan dari PHP ini sama persis dengan kode HTML pertama.

Bagaimana jika ingin menampilkan 1000 nama mahasiswa? Menggunakan kode HTML “murni”, saya terpaksa men-copy-paste baris `<li>Nama Mahasiswa ke-1</li>` sebanyak 1.000 kali, kemudian mengganti satu-satu urutan angka yang ada.

Dengan bantuan PHP, hal ini sangat mudah dilakukan. Berikut revisi kode programnya:

### 03.daftar\_mahasiswa\_1000.php

---

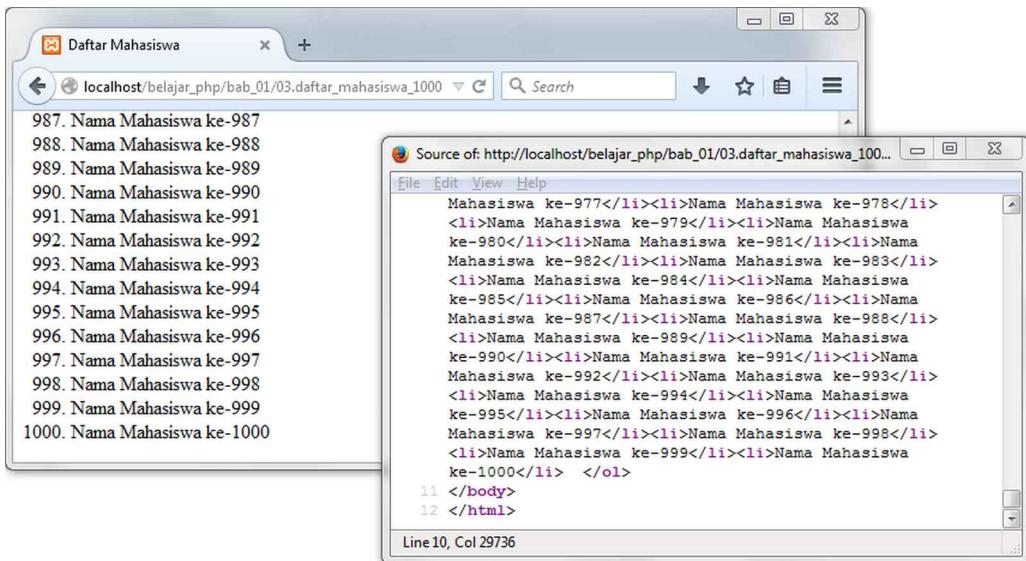
```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Daftar Mahasiswa</title>
6  </head>
7  <body>
8  <h2>Daftar Absensi Mahasiswa</h2>
9  <ol>
10     <?php
11         for ($i= 1; $i <= 1000; $i++) {
12             echo "<li>Nama Mahasiswa ke-$i</li>";
13         }
14     ?>
15     </ol>
16 </body>
17 </html>

```

---

Yup, cukup dengan menambahkan dua angka 0 pada baris ke 11 (dari 10 menjadi 1000), hasilnya adalah 1.000 nama mahasiswa!



Gambar: Daftar 1000 mahasiswa dengan PHP

Bagaimana jika 1.000.000 nama mahasiswa? cukup mengganti kode `$i <= 1000` dengan `$i <= 1000000`. Seperti inilah yang yang saya maksud bahwa **PHP digunakan untuk menghasilkan kode HTML**.

Sebenarnya PHP dapat melakukan lebih dari sekedar menghasilkan kode HTML. Kita bisa menggunakan PHP untuk pemrosesan form, mengakses database, management session dan cookie, membaca file teks, menangani file upload, membuat file pdf, membuat file excel, dan masih banyak lagi. Ini karena PHP adalah sebuah bahasa pemrograman web **server side** (*server side programming language*).

Apa itu *server side programming language*? Saya akan membahasnya dengan detail sesaat lagi. Singkatnya, *server side programming language* adalah bahasa pemrograman web yang berjalan di **server**, bukan di web browser seperti HTML, CSS maupun JavaScript.

Kembali ke pengertian PHP, saya belum menyinggung tentang kepanjangan PHP. PHP merupakan singkatan dari **PHP: Hypertext Preprocessor**. Singkatan ini disebut singkatan *rekursif*, yakni permainan kata dimana kepanjangannya juga terdiri dari singkatan PHP itu sendiri, yakni **PHP: Hypertext Preprocessor**.

**Hypertext Preprocessor** bisa diterjemahkan sebagai ‘*pemroses hypertext*’, atau ‘*pemroses HTML*’. Jadi tidak jauh berbeda dengan pengertian awal kita bahwa PHP adalah *bahasa pemrograman web yang digunakan untuk men-generate atau menghasilkan kode HTML*.

Agar lebih *formal*, saya ingin mengutip pengertian PHP dari dari [wikipedia](http://en.wikipedia.org/wiki/PHP)<sup>1</sup>:

**“PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language.”**

**“PHP adalah bahasa pemrograman script server-side yang didesain untuk pengembangan web, dan juga bisa digunakan sebagai bahasa pemrograman umum.”**

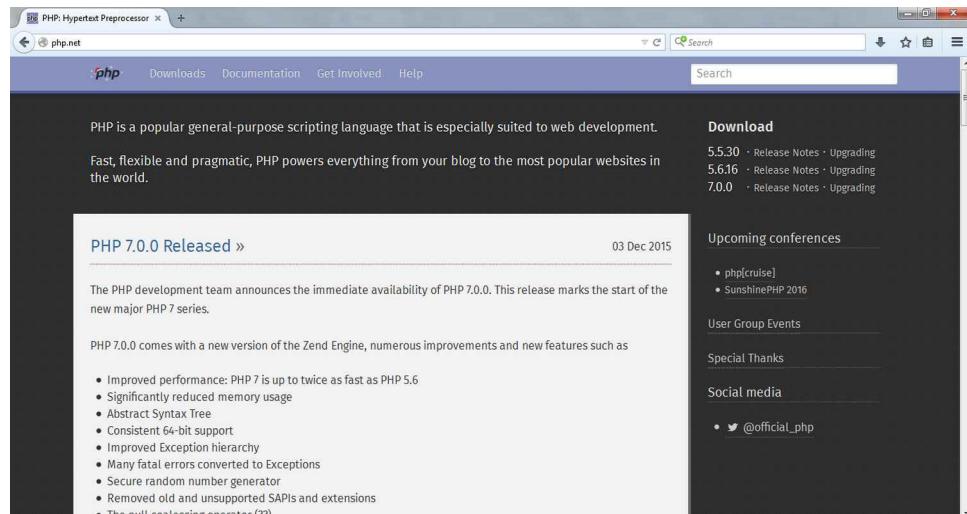
<sup>1</sup><http://en.wikipedia.org/wiki/PHP>

Dari pengertian ini, ternyata PHP juga bisa digunakan membuat aplikasi selain web (*general-purpose programming language*). Dan memang benar, terdapat modul seperti PHP-GTK untuk membuat aplikasi desktop dari bahasa PHP.

**i** Modul PHP-GTK sendiri sebenarnya sudah ‘mati-suri’ sejak tahun 2010 lalu (tidak dikembangkan lagi). Pada bulan Mei 2015, komunitas pengembang PHP-GTK memutuskan untuk menghidupkan kembali proyek ini.

Dalam buku ini saya hanya fokus kepada penggunaan PHP untuk membuat halaman web. Jika anda tertarik dengan PHP-GTK, bisa mengunjungi situs resminya di [gtk.php.net](http://gtk.php.net)<sup>2</sup>.

Saat ini PHP dikembangkan secara kolaborasi oleh berbagai programmer di seluruh dunia yang tergabung ke dalam **The PHP Group**. PHP bersifat open source dan bisa di download dari situs resminya: [php.net](http://php.net)<sup>3</sup>.



Gambar: Halaman awal situs PHP: [php.net](http://php.net)

## 1.2 PHP Sebagai Server Side Programming Language

Jika dibagi menurut tempat dimana kode program diproses, bahasa pemrograman web dapat dibagi menjadi 2 jenis, yakni: *client side programming language* dan *server side programming language*.

### Client-side Programming Language

**Client side programming language** (*bahasa pemrograman berbasis client*) adalah jenis bahasa pemrograman yang berjalan dan di proses di sisi **client**. Ketika berbicara tentang pemrograman

<sup>2</sup><http://gtk.php.net>

<sup>3</sup><http://php.net>

web, *client* mengacu kepada pengunjung website, atau lebih spesifik lagi kepada **web browser** yang digunakan pengunjung ketika mengakses sebuah website.

HTML dan CSS merupakan contoh dari bahasa pemrograman *client side* (walaupun menyebut keduanya sebagai ‘bahasa pemrograman’ sebenarnya kurang tepat). Ketika kita membuka website yang berisi kode HTML dan CSS, kode tersebut diproses di komputer pengunjung (oleh web browser-nya).

JavaScript adalah contoh lain dari bahasa pemrograman *client side*. Seluruh kode JavaScript yang terdapat pada sebuah website, diproses di dalam web browser pengunjung, yakni disisi **client**.

## Server-side Programming Language

Di lain pihak, **server side programming language** (*bahasa pemrograman berbasis server*) adalah kelompok bahasa pemrograman yang prosesnya di lakukan di dalam **server**, bukan di komputer pengunjung.

Karena termasuk *bahasa pemrograman berbasis server*, kita harus menjalankan kode-kode PHP dari sebuah **server**. Dengan kata lain, kode PHP tidak bisa dijalankan tanpa **server**.

### Mengenal Istilah Client - Server

Dalam dunia komputer, **server** adalah sebuah software dan/atau hardware yang berfungsi memproses ‘sesuatu’, untuk dikembalikan hasilnya kepada **client**. Oleh karena itulah terdapat istilah **client-server**.

Analoginya bisa disamakan dengan memesan makanan di restoran. Dapur tempat masakan disiapkan adalah **server**. Kita sebagai customer adalah **client**.

Setiap kali ada permintaan makanan dari kita (**client**), pramusaji akan meneruskannya ke dapur (**server**). Tergantung banyaknya permintaan, koki di dapur perlu beberapa saat untuk membuat masakan. Setelah selesai, pramusaji akan mengantar pesanan dari dapur (**server**) kembali ke kita (**client**).

Restoran juga memiliki keterbatasan seberapa banyak masakan yang bisa dilayani dalam satu waktu. Jika pesanan datang dalam jumlah banyak dan pada waktu yang bersamaan, koki di dapur bisa kewalahan, dan kita pun terpaksa menunggu lebih lama hingga makanan datang. Hal ini bisa diatasi dengan menambah koki atau memperbesar dapur (*upgrade server*).

Server tersedia dalam berbagai fungsi, tergantung layanan yang disediakan. Beberapa contohnya adalah:

- **File Server**: menyimpan dan berbagi pakai file (sharing).
- **Database Server**: menyimpan dan menampilkan data.
- **Email Server**: menyimpan dan mengatur lalu lintas email.
- **Web Server**: memproses dan menangani permintaan halaman web.

Secara fisik, server ini hanyalah komputer biasa dengan komponen ‘spesial’, misalnya menggunakan processor khusus server hingga 8 atau 16 core, memory RAM hingga 16GB atau 32GB,

dan harddisk hingga puluhan TeraByte (tergantung kebutuhan).

Untuk mengubah sebuah komputer menjadi server, kita tinggal menginstall aplikasi khusus server. Sebagai contoh, untuk **database server** tersedia aplikasi *MySQL*, *MariaDB*, *MS SQL*, *Oracle*, dll. Untuk **web server**, tersedia aplikasi *Apache*, *Nginx*, *IIS*, *LiteSpeed Web Server*, dll.

**Web server** inilah yang diperlukan untuk menjalankan kode program PHP. Dalam bab 3 nanti kita akan menginstall aplikasi **XAMPP** yang terdiri dari web server **Apache** dan database server **MySQL**.

## 1.3 Cara Kerja PHP dan Web Server

Pembahasan berikut ini mungkin terasa sedikit rumit, tapi sangat penting untuk bisa memahami “apa itu PHP”. Saya akan mengajak anda melihat bagaimana cara kerja PHP dalam menghasilkan kode HTML. Agar bisa memahaminya, kita juga harus membahas fungsi dari **web server**.

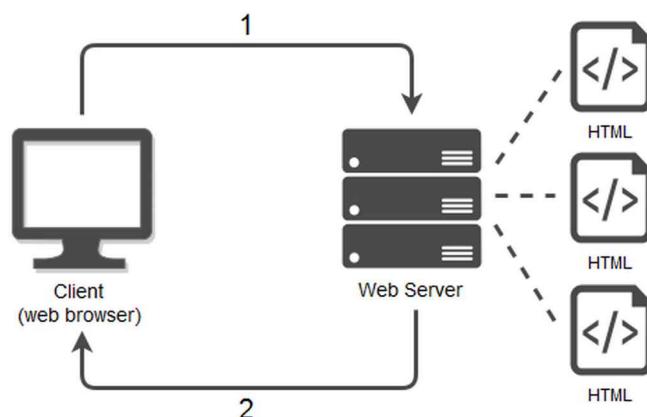
Memproses kode PHP hanya salah satu fungsi dari **web server**. Agar bisa diakses dari internet, file HTML, CSS, JavaScript, PHP dan berbagai file pembangun sebuah website, harus ditempatkan ke dalam **web server**. Disini web server berfungsi sebagai pengatur arus lalu lintas data dari dan ke web browser pengunjung.



Untuk situs online, web server ini berada di komputer perusahaan web hosting. Sebagai contoh, situs duniaIlkom dibangun dengan PHP (menggunakan Wordpress). Seluruh file PHP ini saya tempatkan di web hosting.

### Cara HTML di Proses Oleh Web Server

Pertama, mari kita lihat bagaimana proses menampilkan halaman web statis yang terdiri dari HTML saja (tanpa PHP). Perhatikan gambar berikut:



Gambar: Proses menampilkan halaman HTML oleh web server

Berikut penjelasan diagram diatas:

1. Pada saat kita membuka web browser (seperti *Google Chrome* atau *Mozilla Firefox*) dan mengetikkan alamat sebuah situs (misalkan *duniaikom.com/index.html*) lalu menekan tombol **Enter**, sebuah permintaan (*request*) dikirim dari **web browser** kepada **web server**.
2. Web server akan mencari file yang diminta oleh web browser (client). Dalam contoh ini, file yang ingin dicari adalah file *html*. Karena file ini terdiri dari kode HTML saja, web server akan langsung mengirimkan file tersebut kepada web browser tanpa melakukan pemrosesan.

Keitka file dari server sampai di web browser, web browser selanjutnya menerjemahkan kode HTML ini dan menampilkannya di komputer pengunjung.

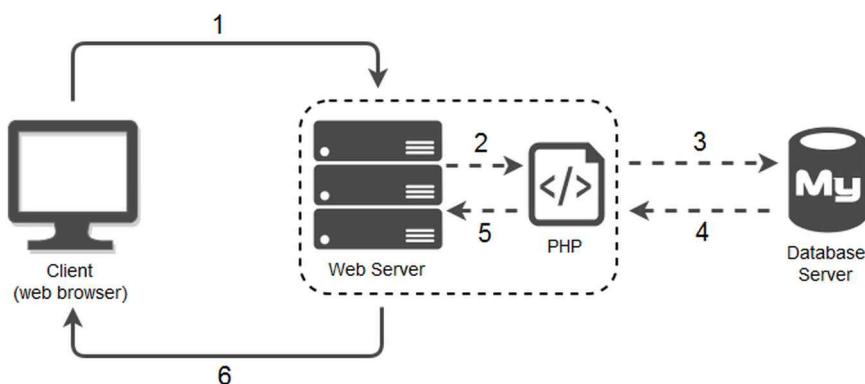
Dalam skenario diatas, file HTML yang dimaksud juga bisa terdiri dari kode CSS dan JavaScript. Semua kode ini termasuk kedalam bahasa pemrograman **client side**, sehingga tidak butuh diproses oleh web server.

Ini juga yang menjadi alasan pada saat menjalankan kode HTML dan CSS secara offline, kita tidak memerlukan web server (seperti seluruh file latihan dalam buku **HTML Uncover** dan **CSS Uncover**).

Menyambung analogi tentang restoran, permintaan file HTML ini bisa diibaratkan dengan menyajikan buah-buahan segar. Koki di dapur (**server**) hanya tinggal memotong buah-buahan dan langsung menyajikannya. Jika pun terpaksa, kita bisa membeli buah-buahan sendiri, yang sama artinya dengan menjalankan file HTML langsung ke browser (tidak perlu **web server**)

## Cara PHP di Proses Oleh Web Server

Bagaimana jika file yang akan dijalankan adalah adalah file PHP? Mari kita lihat diagram proses web dinamis yang menggunakan PHP:



Gambar: Proses menampilkan halaman PHP oleh web server

Penjelasan diagram diatas adalah sebagai berikut:

1. Web browser melakukan permintaan file kepada web server, misalkan yang diminta adalah file *php* pada situs *www.duniaikom.com*, sehingga alamat yang diketik di dalam web browser adalah: *www.duniaikom.com/index.php*.

2. Ketika user menekan tombol **Enter**, permintaan (*request*) akan dikirim kepada web server. Web server kemudian melihat file apa yang diminta. Karena file yang diminta berisi kode PHP, maka web server akan ‘memanggil’ **modul PHP** dan melakukan pemrosesan sesuai dengan aturan PHP.
3. Pada saat web server memproses kode PHP, file *php* bisa saja memiliki perintah untuk mengambil data dari database. Proses permintaan ini selanjutnya dikirim lagi ke **database server**, misalnya database **MySQL**.
4. **Database server** akan memproses permintaan dari PHP dan mengirimkan hasilnya. Proses ini bisa berupa pencarian data, mengisi data baru, mengubah data yang sudah ada, atau menghapus data yang ada di dalam database. Seluruh perintah ini ditulis dengan PHP.
5. **Modul PHP** yang terdapat di dalam web server kemudian menyerahkan hasil proses kepada web server. Web server kemudian mengenerate kode PHP menjadi HTML.
6. Kode HTML hasil dari pemrosesan selanjutnya dikirim ke web browser (client) untuk ditampilkan.

Dalam gambar diatas, saya menambahkan garis putus-putus yang melingkupi **web server** dan **PHP** untuk mempertegas bahwa **PHP merupakan bagian dari web server**. Untuk menjalankan PHP kita harus meletakkan file PHP di dalam web server.

Kembali ke analogi restoran, menjalankan file PHP bisa disamakan dengan restoran yang mampu menyajikan makanan apapun sesuai permintaan. Restoran seperti ini akan sangat fleksibel, karena bisa menyajikan nasi uduk, nasi padang, pizza hingga spaghetti. Jika bumbu masakan tidak tersedia, koki bisa menyuruh anggotanya untuk mencari ke pasar (ke database server).

Namun berbeda dengan restoran ‘impian’ yang mengharuskan kita menunggu sangat lama hingga makanan tiba, PHP diproses dengan sangat cepat. Selain itu kita juga bisa mengoptimalkan kode-kode PHP dan membuatnya dijalankan hampir sama dengan kode HTML ‘biasa’.

Jika ini adalah pertama kalinya anda belajar PHP, penjelasan diatas mungkin terasa membungkungkan. Dan ini sangat dapat dimaklumi.

Alur proses kerja ini akan semakin jelas ketika kita telah menjalankan beberapa kode program PHP. Oleh karena itu, jika anda belum begitu paham dengan penjelasan diatas, saya sarankan kembali kesini setelah menjalankan 1 atau 2 kode program PHP.

---

Dalam bab selanjutnya, saya akan mengajak anda untuk mendengar perjalanan panjang kisah PHP, mulai dari awal kemunculannya di tahun 1994 hingga PHP versi 7 yang baru saja rilis beberapa waktu yang lalu.

# 2. Sejarah dan Perkembangan PHP

Sebagaimana layaknya teknologi yang selalu berkembang, PHP juga melalui banyak perubahan dari tujuan awal. Bahkan PHP sebenarnya tidak direncanakan menjadi sebuah bahasa pemrograman.

Oleh karena itu, tidak ada salahnya kita melihat sejenak bagaimana perjalanan PHP sejak kemunculan pertamanya lebih dari 2 dekade yang lalu. Sejarah PHP ini saya banyak mengambil sumber dari [wikipedia<sup>1</sup>](https://en.wikipedia.org/wiki/PHP) dan [php.net<sup>2</sup>](http://php.net/manual/en/history.php.php)

## 2.1 Sejarah PHP

### PHP/FI : Personal Home Page/Forms Interpreter

Cikal bakal PHP berasal dari sebuah script dalam bahasa C yang dibuat tahun 1994 oleh **Rasmus Lerdorf**. *Rasmus Lerdorf* merupakan seorang programmer kelahiran Denmark yang saat ini berdomisili di Kanada.

Script yang dirancang *Lerdorf* ini bertujuan untuk mencatat jumlah user yang mengunjungi website pribadinya. Beberapa waktu kemudian, ia menambahkan fitur lain seperti penanganan form HTML dan menampilkan data dari database.

Rasmus Lerdorf menyebut kode program ini dengan sebutan **Personal Home Page/Forms Interpreter** atau **PHP/FI**. Inilah asal mula penamaan PHP. PHP/FI dapat digunakan untuk membuat aplikasi web dinamis sederhana.

Satu tahun berikutnya (1995), Rasmus Lerdorf merilis kode tersebut ke publik dengan nama **Personal Home Page Tools (PHP Tools) version 1.0**, yang kemudian dikenal sebagai **PHP 1**. Dapat dilihat bahwa pada awalnya PHP merupakan singkatan dari *Personal Home Page*.

Perilisan ini diumumkan pada 8 Juni 1995 di alamat *comp.infosystems.www.authoring.cgi*, sebuah group diskusi *Usenet*. Jika anda tertarik membaca pengumuman rilis PHP versi 1 ini bisa mengaksesnya di [groups.google.com<sup>3</sup>](https://groups.google.com/forum/#msg/comp.infosystems.www.authoring.cgi/PyJ25gZ6z7A/M9FkTUVDFcwJ).

**PHP/FI** sebenarnya tidak di tujuhan menjadi bahasa pemrograman sendiri. Namun dengan dirilisnya source code PHP/FI ke publik, perkembangan PHP/FI menjadi sangat pesat.

Saat itu Lerdorf [berkata<sup>4</sup>](#): *I don't know how to stop it, there was never any intent to write a programming language* (aku tidak tahu bagaimana cara menghentikannya, tidak pernah terpikir bagiku untuk membuat sebuah bahasa pemrograman baru).

---

<sup>1</sup><https://en.wikipedia.org/wiki/PHP>

<sup>2</sup><http://php.net/manual/en/history.php.php>

<sup>3</sup><https://groups.google.com/forum/#msg/comp.infosystems.www.authoring.cgi/PyJ25gZ6z7A/M9FkTUVDFcwJ>

<sup>4</sup>[http://web.archive.org/web/20130729213507id\\_/http://itc.conversationsnetwork.org/shows/detail58.html](http://web.archive.org/web/20130729213507id_/http://itc.conversationsnetwork.org/shows/detail58.html)

## PHP/FI : Personal Home Page/Forms Interpreter 2

PHP 2 atau lebih lengkapnya Personal Home Page/Forms Interpreter 2 dirilis Rasmus Lerdorf pada tahun 1996 dengan penambahan fitur seperti struktur logika IF ELSE, serta peningkatan performa dibandingkan versi PHP 1. PHP versi 2 ini dirancang Lerdorf pada saat mengerjakan sebuah proyek di **University of Toronto** yang membutuhkan pengolahan data dan tampilan web yang rumit.

PHP 2 cukup populer digunakan oleh programmer saat itu, tetapi memiliki masalah dengan kestabilan program yang kurang bisa diandalkan. Hal ini lebih dikarenakan Lerdorf hanya bekerja sendiri dalam mengembangkan PHP.

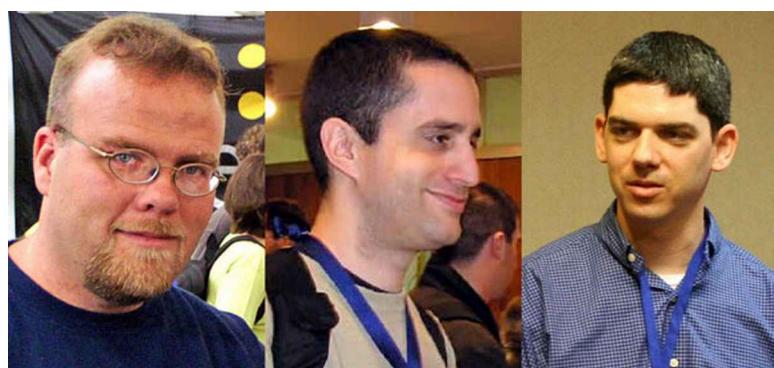
## PHP: Hypertext Preprocessor 3

Sekitar tahun 1997, **Zeev Suraski** dan **Andi Gutmans** ikut mengambil bagian dalam pengembangan PHP. Mereka membuat ulang *parsing engine* PHP agar lebih stabil. Proses membuat ulang *parsing engine* ini bisa diibaratkan dengan menulis ulang seluruh kode program PHP.

Ditambah dengan dukungan berbagai programmer lainnya, proyek PHP secara perlahan mulai beralih dari proyek satu orang menjadi proyek massal yang lebih akrab kita kenal sebagai *open-source project*. PHP selanjutnya dikembangkan oleh **The PHP Group** yang terdiri dari kumpulan programmer dari seluruh dunia.

Akhirnya pada tahun 1998 PHP versi 3 dirilis ke publik. Perilisan PHP 3 juga ditandai dengan perubahan singkatan PHP yang sebelumnya **PHP/FI: Personal Home Page Tools**, menjadi **PHP: Hypertext Preprocessor**. Perubahan ini juga menandakan bahwa PHP lebih dari sekedar *tool* (alat) untuk membuat halaman web pribadi.

Kebanyakan syntax atau perintah dasar PHP yang akan kita pelajari dalam buku ini berasal dari PHP 3. Pada puncaknya, PHP 3 digunakan oleh sekitar 10% web server di seluruh dunia.



Gambar: Pengembang utama PHP: Rasmus Lerdorf (kiri), Andi Gutmans (tengah) dan Zeev Suraski (kanan) (sumber: wikipedia)

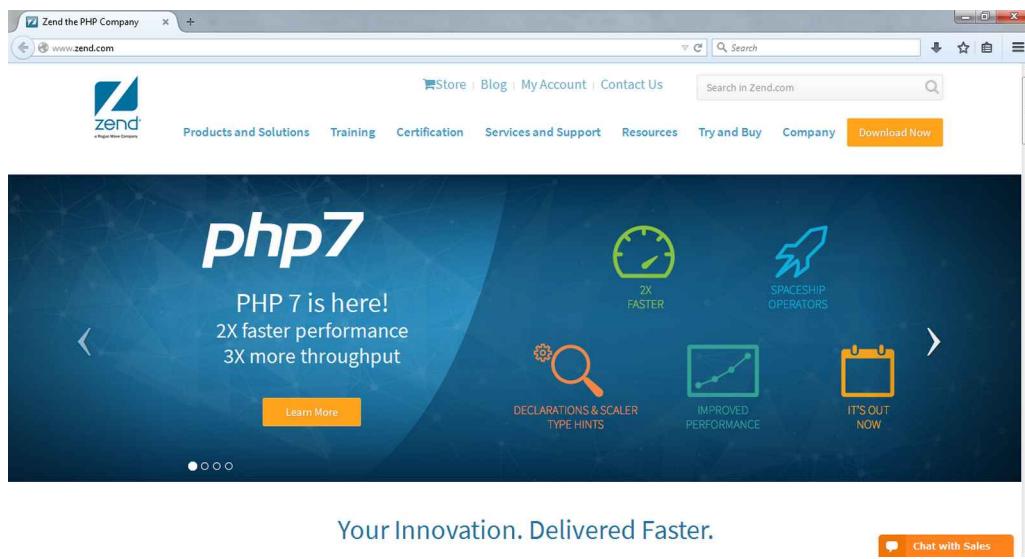
## PHP: Hypertext Preprocessor 4

Beberapa tahun setelahnya (tepatnya pada 22 Mei tahun 2000) PHP versi 4 dirilis. *Parsing engine* PHP 4 kembali dibuat ulang oleh **Zeev Suraski** dan **Andi Gutmans**, dan dinamakan **Zend**

**Engine** (gabungan dari nama Zeev dan Andi). Zend Engine memperbaiki performa PHP dari versi sebelumnya terutama untuk memproses data yang kompleks.

PHP versi 4 membawa banyak fitur baru, seperti *HTTP sessions*, *output buffering*, dan memperkenalkan *Object Oriented Programming* (OOP / pemrograman berbasis objek). Walaupun demikian, OOP pada PHP 4 masih belum sempurna.

- i** Zeev Suraski dan Andi Gutmans kemudian mendirikan **Zend Technologies**. Perusahaan ini mengembangkan berbagai produk terkait PHP, seperti *Zend Server*, *Zend Studio*, *Zend Framework*, dan *Zend Engine*. Situs resminya beralamat di: [www zend com](http://www zend com)<sup>5</sup>.



Gambar: Situs Zend Technologies, [www zend com](http://www zend com)

## PHP: Hypertext Preprocessor 5

Memperbaiki versi PHP sebelumnya, PHP 5 diluncurkan pada 13 Juli 2004. PHP 5 telah mendukung penuh pemrograman object (perbaikan dari PHP 4) serta peningkatan performa melalui *Zend Engine* versi 2.

PHP 5 juga menyertakan banyak fitur baru, seperti **PDO** (*PHP Data Objects*) untuk pengaksesan *database*, *closures*, *trait*, dan *namespaces*.

- i** Hingga saat ini, PHP 5 adalah versi PHP yang paling banyak digunakan. Ini karena sejak pertama kali dirilis pada 13 Juli 2004, hingga akhir November 2015 lalu, PHP tidak merilis versi 6 (penjelasannya akan saya bahas setelah ini). Selama 11 tahun tersebut, PHP hanya menggunakan penomoran minor, seperti 5.1, 5.2, hingga 5.6 (versi terakhir).

<sup>5</sup>[www zend com](http://www zend com)

## PHP: Hypertext Preprocessor 6

Versi lanjutan dari PHP, yakni PHP 6 sebenarnya telah lama dikembangkan, dimulai sejak tahun 2005 (satu tahun setelah PHP 5 dirilis). Fokus pengembangan PHP 6 terutama untuk mendukung *Unicode*, agar PHP bisa digunakan dengan berbagai jenis karakter bahasa non-latin.

Dalam perjalannya, performa PHP 6 ternyata tidak memuaskan. Dukungan untuk *unicode* membuat PHP berjalan lebih lambat. Selain itu fitur *unicode* sebenarnya tidak terlalu dibutuhkan. Ditambah dengan beberapa permasalahan lain, pengembangan PHP 6 menemui jalan buntu dan akhirnya dihentikan.

Fitur-fitur yang telah dirancang untuk PHP 6 akhirnya ditambahkan ke PHP 5. Terhentinya pengembangan PHP 6 membuat PHP seolah-olah ‘*stagnan*’ dan berhenti pada PHP 5.

## PHP: Hypertext Preprocessor 7

Pada tanggal 3 Desember 2015, PHP 7 resmi dirilis. Perubahan yang paling terlihat adalah peningkatan performa. Menggunakan **Zend Engine 3**, PHP 7 di-klaim berjalan 2 kali lebih cepat daripada PHP 5.6.

*Core engine* PHP 7 berasal dari proyek eksperimen **phpng** (*PHP next generation*), yang dikembangkan oleh **Dmitry Stogov**, **Xinchen Hui** dan **Nikita Popov**. Proyek ini menggunakan pendekatan modern agar PHP diproses dengan lebih cepat, seperti teknik *just-in-time (JIT) compiler*.

Selain performa yang meningkat, terdapat beberapa fitur baru di PHP 7, seperti *combined comparison operator* atau dikenal dengan *spacehip operator* “`<=>`”, *anonymous classes*, dan dukungan yang lebih stabil untuk server 64-bit.

Beberapa fitur yang sudah ‘usang’ (*deprecated*) juga dihapus, seperti penulisan PHP dengan ASP style `<% %>` dan `<script language=php> </script>`. Kedua cara ini sudah tidak bisa digunakan lagi. Penggunaan *mysql extension* juga dihapus, karena sudah digantikan dengan *mysqli extension*.

Terdapat hal unik dalam penamaan versi PHP. Sebelum PHP 7, versi terakhir dari PHP adalah PHP 5. Kemana PHP 6?

Setelah perdebatan yang cukup panjang, tim dibalik pengembangan PHP mengambil voting dan memutuskan tidak menamai PHP terbaru dengan PHP 6, tapi PHP 7. Tujuannya, agar menghindari kebingungan dengan buku PHP 6 yang sudah lama beredar. Versi PHP akan langsung ‘loncat’ dari PHP 5 menjadi PHP 7. Dengan kata lain, PHP 6 ‘tidak pernah dilahirkan’.

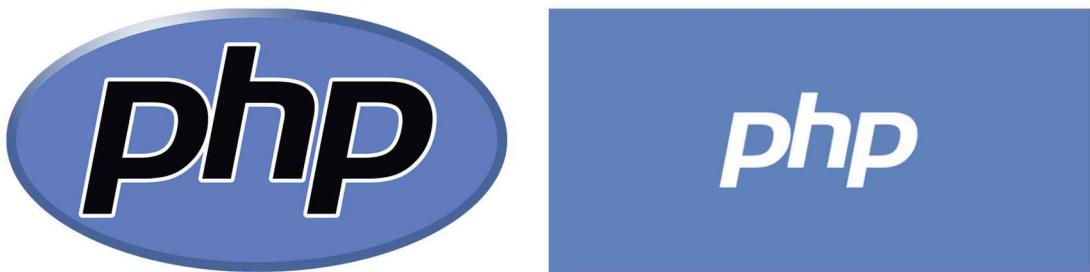


Karena masih sangat baru, penggunaan PHP 7 belum begitu banyak. Agar lebih universal, dalam buku ini saya akan menggunakan PHP versi 5.6.15.

Namun, hampir seluruh pembahasan yang ada juga tetap berjalan di PHP 7. Bahkan mungkin juga bisa berjalan di PHP 4. Ini karena walaupun PHP akan terus berkembang, konsep dasarnya tidak banyak berubah. Tipe data dasar seperti string, integer, boolean, struktur if dan looping tidak berubah sejak PHP 3. Perubahan yang ada dalam versi PHP 7 lebih banyak kepada fitur-fitur lanjutan.

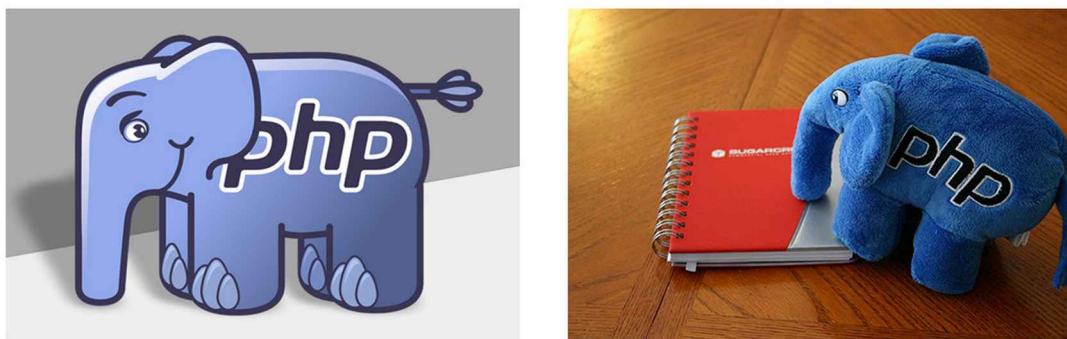
## 2.2 Logo dan Maskot PHP

PHP menggunakan logo resmi berupa tulisan PHP hitam di dalam lingkaran oval. Diakses dari wikipedia, PHP juga memiliki logo yang menggunakan font berwarna putih dan terkesan lebih modern.



Gambar: Logo resmi PHP dari situs php.net (kiri) Logo PHP dari wikipedia (kanan)

Selain itu, PHP juga memiliki maskot seekor gajah berwarna biru yang dikenal dengan **elePH-Pant**. Logo ini didesain oleh [Vincent Pontier](#)<sup>6</sup>.



Gambar: Maskot gajah PHP: elePHPant

## 2.3 Kepopuleran PHP

Ketika ingin mempelajari sebuah bahasa pemrograman baru, salah satu pertimbangan adalah seberapa populer bahasa tersebut. Apakah sudah terlalu ‘tua’ sehingga sudah mulai ditinggalkan, atau terlalu baru dan belum jelas masa depannya.

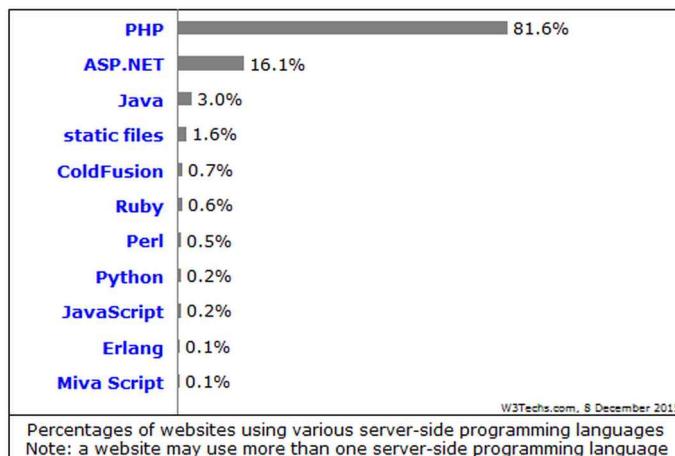
Berbeda dengan **HTML**, **CSS** dan **JavaScript** yang ‘memonopoli’ bidangnya masing-masing, **PHP** memiliki banyak pesaing. Untuk merancang tampilan web, kita harus menggunakan **HTML** dan **CSS**, belum ada bahasa pemrograman lain yang bisa menggantikannya. Tetapi kita bisa menggunakan bahasa pemrograman **ASP** atau **Python** untuk menggantikannya **PHP**.

Saat ini, bahasa pemrograman web berbasis server (*server side programming language*), terdiri

<sup>6</sup><http://www.elroubio.net>

dari beberapa pilihan: **PHP**, **ASP.NET**, **JAVA**, **ColdFusion**, **Ruby**, **Perl**, **Python**, dll. Bagaimana tingkat popularitas penggunaan bahasa pemrograman ini?

Berikut grafik tingkat penggunaan *server side programming language* yang disurvei oleh situs [w3techs.com](http://w3techs.com)<sup>7</sup> :

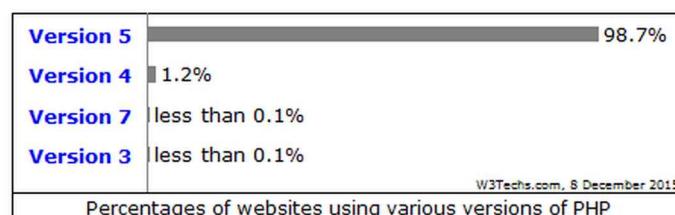


Gambar: Persentase penggunaan server side programming language dari survei situs [w3techs.com](http://w3techs.com) (diakses 8 Desember 2015)

Grafik diatas memperlihatkan tingkat penggunaan *server side programming language*. PHP digunakan oleh 80% lebih web server diseluruh dunia. Pesaing terdekat PHP adalah bahasa pemrograman ASP.NET buatan Microsoft yang dipakai oleh 16% web server. Hal ini memperlihatkan, walaupun PHP sudah berumur 20 tahun lebih, tapi masih sangat populer dan menjadi bahasa pemrograman yang wajib dikuasai saat ini.

Jika anda berencana membuat website ‘online’ dan menyewa web hosting, Majoritas web hosting menyediakan server dengan bahasa PHP secara *default* (baik perusahaan hosting dalam negeri maupun luar negeri).

Dari 81,6% web server yang menggunakan PHP, versi PHP berapa yang paling banyak digunakan? berikut grafiknya:



Gambar: Versi PHP yang banyak digunakan)

Dari grafik terlihat versi PHP yang paling banyak digunakan adalah versi 5. Ini tidak mengejutkan, karena PHP 5 sudah beredar selama 11 tahun. PHP 7 sebagai penerus versi PHP juga baru saja dirilis.

Butuh waktu yang tidak sebentar untuk peralihan dari PHP 5 ke PHP 7. Prediksi saya, baru 1

<sup>7</sup>[http://w3techs.com/technologies/overview/programming\\_language/all](http://w3techs.com/technologies/overview/programming_language/all)

atau 2 tahun lagi PHP 7 bisa menggantikan posisi PHP 5 sebagai versi PHP yang paling banyak digunakan. Itupun dengan catatan framework PHP dan CMS Populer seperti wordpress juga ber-migrasi ke PHP 7.

## 2.4 Framework PHP dan CMS

Dalam buku **PHP Uncover** ini kita akan mempelajari dasar-dasar PHP, dan bagaimana cara menggunakan PHP untuk membuat website dinamis. Walaupun begitu, saya ingin membicarakan sedikit tentang **Framework PHP** dan **CMS (Content Management System)**.

Saat ini, proses pembuatan website menjadi semakin kompleks dari waktu ke waktu. Hal ini seiring dengan perkembangan teknologi yang menuntut perubahan terus menerus. Bagi kita sebagai web programmer, ini menjadi tantangan yang harus dihadapi.

Agar tidak ketinggalan, kita juga harus kreatif dan efisien. Daripada membuat segala sesuatunya dari nol, kita bisa memanfaatkan kode program orang lain yang sudah jadi. Istilahnya: “*don’t reinvent the wheel*” (maksudnya, jangan membuat sesuatu yang memang sudah ‘tersedia’). Dalam dunia web programming, **Framework** dan **CMS** adalah bagian dari “*don’t reinvent the wheel*”.

Secara sederhana, **Framework** dan **CMS** adalah kumpulan kode program siap pakai yang bisa digunakan untuk membangun website dengan cepat. Khusus untuk CMS, kita bisa membuat website tanpa mengerti coding sama sekali. Sebagai contoh, situs duniaIlkom saya bangun dengan menggunakan **CMS Wordpress**.

Sebagian besar Framework dan CMS populer dibuat dari PHP, seperti: **Code Igniter**, **Laravel**, **Symfony**, **Zend Framework**, **Wordpress**, **Joomla** dan **Drupal**.

Kenapa saya ingin menyinggung tentang framework dan CMS? Karena jika anda ingin serius berkarir sebagai web programmer, Framework dan CMS ini wajib dikuasai. Buktinya, mayoritas lowongan kerja programmer PHP, mewajibkan anda untuk menguasai salah satu Framework PHP.

Akan tetapi, sebelum sampai ke framework, anda harus terlebih dahulu paham dasar-dasar PHP. Ingin membuat theme Wordpress atau template Joomla? Juga wajib menggunakan PHP.

Dengan sekian banyak alasan mempelajari PHP, dalam bab berikutnya, kita akan mulai mempersiapkan perangkat yang dibutuhkan dengan menginstall XAMPP dan Text Editor.

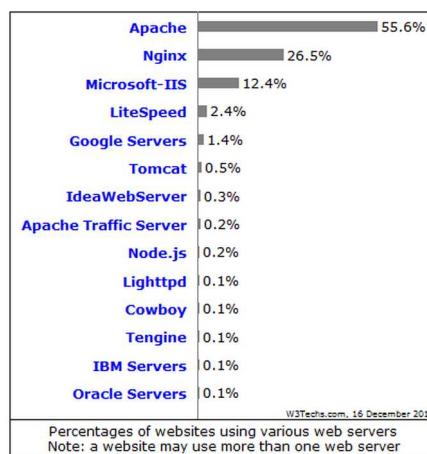
# 3. Instalasi XAMPP, Web Browser dan Text Editor

Dalam bab ini kita akan mempersiapkan perangkat dan aplikasi yang di butuhkan untuk menjalankan kode PHP. Aplikasi tersebut adalah XAMPP, Web Browser dan Text Editor.

## 3.1 Mengenal Aplikasi XAMPP

Pada bab pertama, saya telah membahas sekilas bagaimana cara PHP berkerja. PHP membutuhkan aplikasi **web server** untuk menerjemahkan kode-kode PHP menjadi HTML. Di dalam web server inilah nantinya kode PHP akan ditempatkan.

Terdapat berbagai aplikasi web server, seperti: *Apache*, *Nginx*, *Microsoft IIS*, dan *LiteSpeed*. Saat ini *Apache web server* menjadi web server yang paling banyak digunakan. Selain *Apache*, *Nginx* juga sering dipilih, karena dianggap lebih ringan dan membutuhkan sumber daya (processor dan memory RAM) yang lebih sedikit jika dibandingkan dengan *Apache*.



Gambar: Tren Penggunaan Web Server dari situs w3techs.com

Perlu menjadi catatan bahwa **web server** dan **PHP** adalah 2 aplikasi yang berbeda dan saling terpisah. Memproses kode PHP merupakan salah satu fungsi dari web server. Sebuah web server bisa mendukung bahasa *PHP*, *ASP*, *Python*, dan bahasa pemrograman web lainnya.

Untuk membuat proses instalasi semakin rumit, kita harus menginstall web server dan PHP secara terpisah. Kemudian, mengatur agar keduanya bisa saling ‘berkomunikasi’. Jika nanti butuh aplikasi **database server** seperti MySQL, kita juga harus menghubungkannya dengan web server. Bagi pemula, menginstall satu-satu aplikasi ini (dan mengatur konfigurasinya) bisa membuat sakit kepala.

Untungnya, terdapat aplikasi yang ‘membundle’ paket **web server + PHP + MySQL** ke dalam 1 kali proses instalasi. Dengan menginstall aplikasi ‘paket’ ini, kita sudah mendapatkan 3 aplikasi lengkap (dan settingannya) sehingga sudah siap pakai.

Salah satu aplikasi ‘bundle’ ini adalah **XAMPP**. Nama aplikasi XAMPP terdiri dari paket yang ada di dalamnya. X (berarti cross-platform, maksudnya tersedia dalam berbagai sistem operasi), Apache Web Server, MySQL Database Server, PHP dan Perl. Selain aplikasi ini, XAMPP juga menyertakan aplikasi pelengkap seperti phpMyAdmin, File Zilla FTP Server, Mercury Mail Server, dll.

XAMPP tersedia gratis dan bisa di download melalui situs resminya di: [apachefriends.org](https://www.apachefriends.org/)<sup>1</sup>.



Gambar: Situs apachefriend.org

## Mengenal AMP Stack

XAMPP adalah salah satu aplikasi yang dikenal sebagai **AMP Stack**. AMP merupakan singkatan dari Apache Web Server, MySQL Database Server, dan PHP. Dalam pengembangan web, ketiga aplikasi ini sangat populer digunakan.

Jika AMP Stack dijalankan di sistem operasi Windows, namanya menjadi **WAMP** (Windows-Apache-MySQL-PHP). Jika dijalankan di OS Linux, menjadi **LAMP** (Linux-Apache-MySQL-PHP). Untuk Mac OS dikenal sebagai **MAMP** (Mac-Apache-MySQL-PHP). Anda akan sering menemukan istilah-istilah ini ketika membahas tentang instalasi PHP.

Selain XAMPP, aplikasi AMP Stack populer lain adalah **WAMP Server**<sup>2</sup>, **AMPPS**<sup>3</sup>, dan **BitNami WAMPStack**<sup>4</sup>.

## MySQL vs MariaDB

Jika anda sudah membuka situs XAMPP di [apachefriends.org](https://www.apachefriends.org/), ada sebuah pengumuman penting. Mulai dari versi 5.5.30 keatas, XAMPP tidak lagi menggunakan MySQL, tetapi **MariaDB**. Kemana MySQL?

<sup>1</sup><http://apachefriends.org>

<sup>2</sup><http://www.wampserver.com/en>

<sup>3</sup><http://www.ampps.com>

<sup>4</sup><https://bitnami.com/stack/wamp>

Tidak dapat dipungkiri bahwa MySQL adalah aplikasi database paling terkenal dan paling banyak digunakan untuk pengembangan web. Jika digabung menjadi **PHP-MySQL**, keduanya merupakan aplikasi yang sangat powerful dan digunakan pada jutaan situs di internet. Ini tidak lepas dari lisensi MySQL yang open source dan bisa digunakan secara gratis.

Pada tahun 2008, perusahaan dibalik MySQL, yakni **MySQL AB** dibeli oleh **Sun Microsystems** (perusahaan yang mengembangkan bahasa pemrograman JAVA). Satu tahun setelahnya, giliran **Sun Microsystems** yang dibeli oleh raksasa industri database: **Oracle**.

Banyak kalangan yang menyayangkan hal ini, karena besar kemungkinan Oracle akan menganaktirikan MySQL. Ini terbukti sejak berada di tangan Oracle, pengembangan MySQL sedikit melambat.

Untuk menghindari suatu saat MySQL ‘dihapus’ oleh Oracle, kalangan programmer bersatu untuk membuat database server ‘cloningan’ MySQL, dan lahirlah **MariaDB**. Salah satu programmer inti dibalik **MariaDB** tidak lain adalah **Monty Widenius**, yang dulunya juga ‘melahirkan’ MySQL.

### **Jadi, apa beda MySQL dengan MariaDB?**

Karena menggunakan source code yang sama dengan MySQL, MariaDB sangat mirip dengan MySQL. Setiap kode program yang ditulis untuk MySQL, hampir dipastikan juga berjalan di MariaDB, termasuk fungsi **mysqli\_connect()** yang menghubungkan PHP dan MySQL. Selain itu, MariaDB memiliki fitur-fitur tambahan yang tidak dimiliki oleh MySQL.

Tetap diakui bahwa nama MariaDB masih relatif asing. MySQL jauh lebih populer. XAMPP versi terbaru sepertinya ‘memaksa’ kita untuk beralih dari MySQL ke MariaDB. Namun seperti yang saya tulis sebelumnya, secara teknis MariaDB nyaris sama dengan MySQL. Anda tidak akan menyadari bahwa XAMPP versi baru menggunakan MariaDB, bukan MySQL.

Bagi kita sebagai web programmer, perubahan seperti ini akan selalu hadir. Pertanyaannya, apakah kita siap (dan mampu) mengikuti perubahan. Untuk kasus MySQL vs MariaDB, hal ini tidak terlalu masalah, karena tidak ada perubahan dari segi kode program.

Jika anda ingin menggunakan MySQL ‘asli’ bisa menggunakan XAMPP versi 5.5.24 kebawah.



Karena kode PHP yang digunakan untuk mengakses MariaDB sama dengan kode program untuk MySQL, selain dari bab ini saya tetap menggunakan istilah ‘MySQL’, walaupun yang kita gunakan adalah MariaDB.

## **3.2 Instalasi XAMPP**

Pada saat buku ini ditulis, versi terakhir dari XAMPP adalah **XAMPP 7.0.2**. Penamaan versi XAMPP saat ini mengikuti versi PHP yang terdapat di dalamnya. Dengan kata lain, XAMPP 7.0.2 berisi **PHP 7.0.2**. Selain itu, XAMPP 7.0.2 terdiri dari web server **Apache 2.4.18** dan database server **MariaDB 10.1.10**.

Jika komputer/laptop yang anda gunakan sudah relatif baru (Windows 7, 8 dan 10), silahkan download **XAMPP 7.0.2**, atau versi diatasnya. Dalam beberapa kasus (seperti yang saya alami),

XAMPP 7.0.2 tidak bisa berjalan. Jika anda mengalami hal ini, silahkan coba install **XAMPP 5.6.15**. Pembahasan dalam buku ini dapat dijalankan baik di XAMPP 7.0.2 keatas, maupun XAMPP 5.6.15.

Saya sendiri juga akan menggunakan **XAMPP 5.6.15** (PHP 5.6.15) karena lebih universal dan mayoritas web hosting masih menggunakan PHP 5. Jika anda menggunakan **Windows XP**, terpaksa harus menggunakan **XAMPP 1.8.2** yang berisi PHP 5.5.36. Versi XAMPP diatas itu tidak bisa berjalan di Windows XP.

Untuk memulai proses instalasi, silahkan download XAMPP dari [apachefriends.org](http://apachefriends.org)<sup>5</sup>. Atau langsung download dari salah satu link berikut:

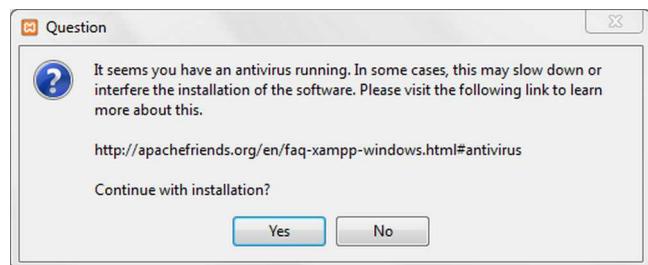
- [XAMPP 7.0.2](https://www.apachefriends.org/xampp-files/7.0.2/xampp-win32-7.0.2-1-VC14-installer.exe)<sup>6</sup> (117 MB), dengan PHP versi 7.0.2
- [XAMPP 5.6.15](https://www.apachefriends.org/xampp-files/5.6.15/xampp-win32-5.6.15-1-VC11-installer.exe)<sup>7</sup> (108 MB), dengan PHP versi 5.6.15
- [XAMPP 1.8.2](http://sourceforge.net/projects/xampp/files/XAMPP%20Windows/1.8.2/xampp-win32-1.8.2-6-VC9-installer.exe/download)<sup>8</sup> (115 MB), dengan PHP versi 5.5.36



Apabila di dalam komputer anda sudah terinstall XAMPP yang masih relatif baru, anda tetap bisa menggunakan aplikasi tersebut. Kode program PHP yang akan kita pelajari sebagian besar masih jalan di PHP 5.4 keatas.

Setelah file aplikasi XAMPP tersedia, kita sudah bisa mulai proses instalasi. Silahkan double klik file XAMPP. Pada contoh ini, file XAMPP yang saya gunakan adalah: **xampp-win32-5.6.15-1-VC11-installer.exe**.

Jika anda menggunakan anti-virus, akan tampil jendela berikut:



Gambar: Saran untuk mematikan anti virus)

Peringatan ini berisi penjelasan bahwa saat ini program anti virus sedang berjalan. Program anti virus bisa jadi akan membuat proses instalasi berjalan lambat atau mengganggu proses instalasi XAMPP. Anda boleh mematikan anti virus untuk sementara (sekitar 10 menit), atau klik saja tombol Yes.

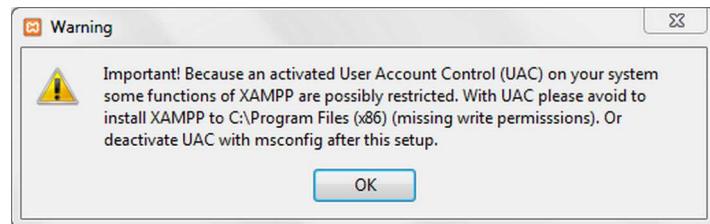
Jendela peringatan berikutnya adalah tentang **UAC (User Account Control)**:

<sup>5</sup><http://apachefriends.org>

<sup>6</sup><https://www.apachefriends.org/xampp-files/7.0.2/xampp-win32-7.0.2-1-VC14-installer.exe>

<sup>7</sup><https://www.apachefriends.org/xampp-files/5.6.15/xampp-win32-5.6.15-1-VC11-installer.exe>

<sup>8</sup><http://sourceforge.net/projects/xampp/files/XAMPP%20Windows/1.8.2/xampp-win32-1.8.2-6-VC9-installer.exe/download>



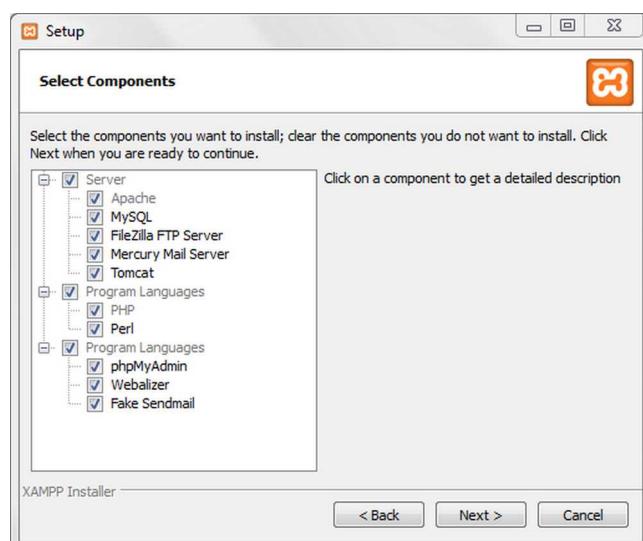
Gambar: Peringatan UAC jika XAMPP diinstall di C:Program Files)

Peringatan ini berkaitan dengan proteksi pada Windows. Dimana jika XAMPP diinstall pada folder **C:Program Files (x86)** mungkin akan terjadi pembatasan hak akses yang menyebabkan XAMPP tidak berjalan dengan normal. Karena alasan ini, saya akan memindahkan file instalasi XAMPP ke folder lain. Silahkan klik tombol **OK** untuk melanjutkan.



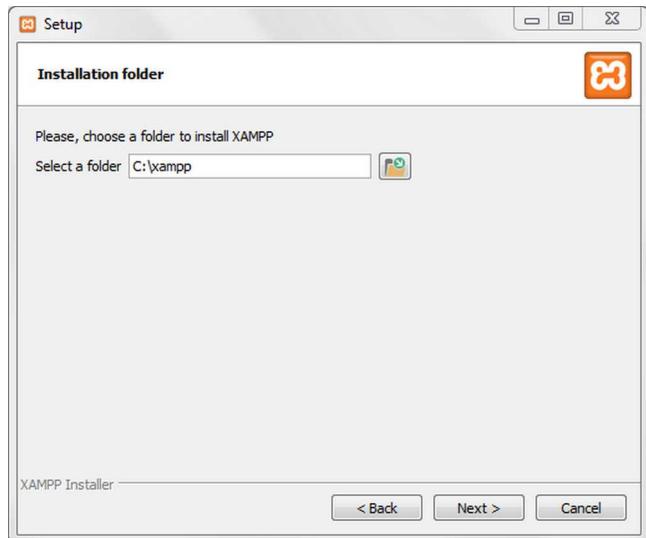
Gambar: Jendela awal instalasi XAMPP)

Jendela awal instalasi akan muncul, Klik **Next**.



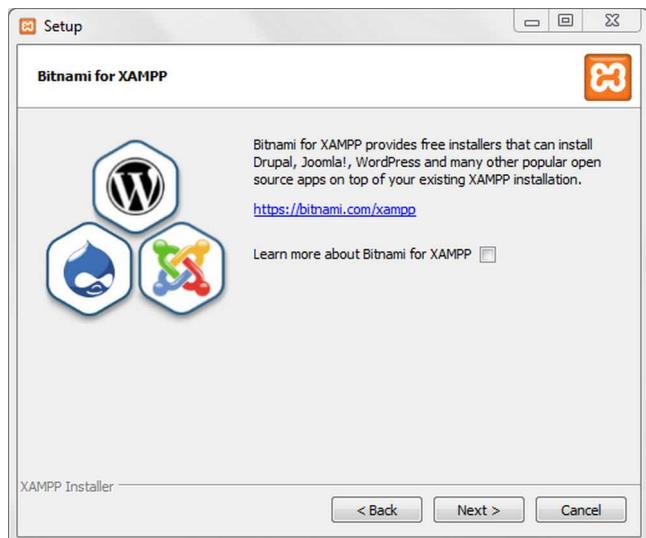
Gambar: Jendela Select Component

Jendela berikutnya adalah “*Select Component*”. Pada bagian ini kita bisa memilih aplikasi apa saja yang akan diinstall. Dalam tahap ini saya akan membiarkan semua pilihan. Klik **Next** untuk melanjutkan.



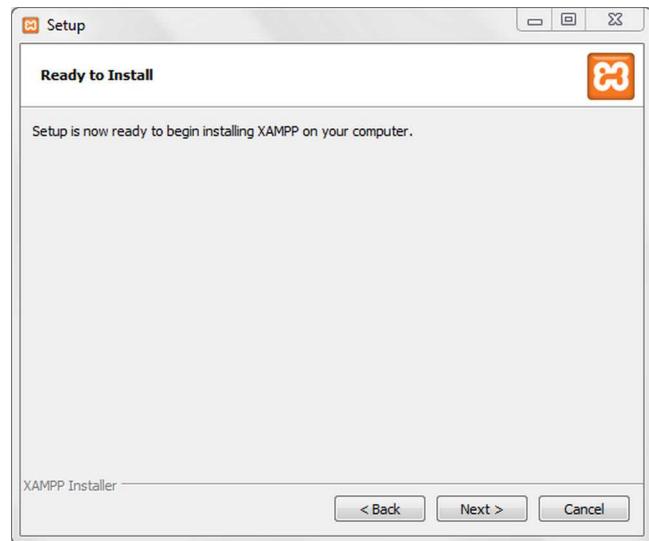
Gambar: Jendela Installation Folder

Jendela selanjutnya adalah “*Installation Folder*”. Dalam bagian ini, kita bisa mengubah lokasi dimana file-file XAMPP akan diinstall. Anda bebas menentukan lokasi manapun. Sebagai contoh, saya akan meletakkan file XAMPP di C:\xampp. Lanjutkan dengan men-klik tombol **Next**.



Gambar: Jendela Bitnami for XAMPP

Tampilan berikutnya adalah jendela “*Bitnami for XAMPP*”. Bitnami adalah aplikasi **AMP stack** yang juga menjadi sponsor XAMPP. Hapus pilihan “*learn more about Bitnami for XAMPP*”, kemudian klik **Next**.



Gambar: Jendela Ready to Install

Jendela berikutnya adalah konfirmasi untuk mulai menginstall XAMPP, klik **Next**, dan XAMPP akan memulai proses instalasi beberapa dalam beberapa saat.



Gambar: XAMPP sedang dalam proses instalasi



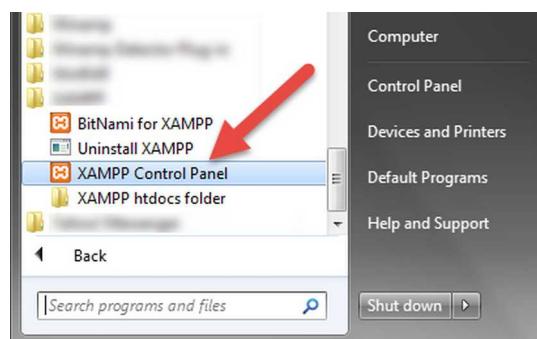
Gambar: XAMPP telah selesai diinstall

Jika jendela “*Completing the XAMPP Setup Wizard*” telah tampil, berarti proses instalasi XAMPP sudah selesai. Pada bagian ini kita akan langsung mencoba aplikasi XAMPP, sehingga biarkan pilihan check list “*Do you want to start the Control Panel now?*”, kemudian klik **Finish**.

### 3.3 XAMPP Control Panel

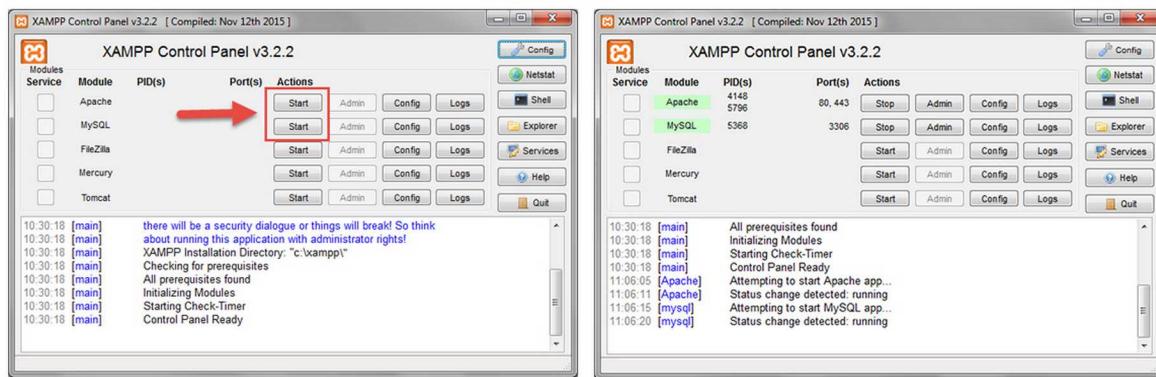
Jika anda membiarkan pilihan “*Do you want to start the Control Panel now?*” pada jendela terakhir proses instalasi XAMPP, maka akan tampil jendela XAMPP Control Panel.

Sesuai dengan namanya, jendela **XAMPP Control Panel** adalah jendela yang digunakan untuk mengontrol apa saja modul XAMPP yang akan atau sedang dijalankan. Jika jendela ini tidak tampil, anda bisa mengaksesnya dari menu **START->All Programs->XAMPP->XAMPP Control Panel**.



Gambar: Menjalankan XAMPP Control Panel dari Menu Windows

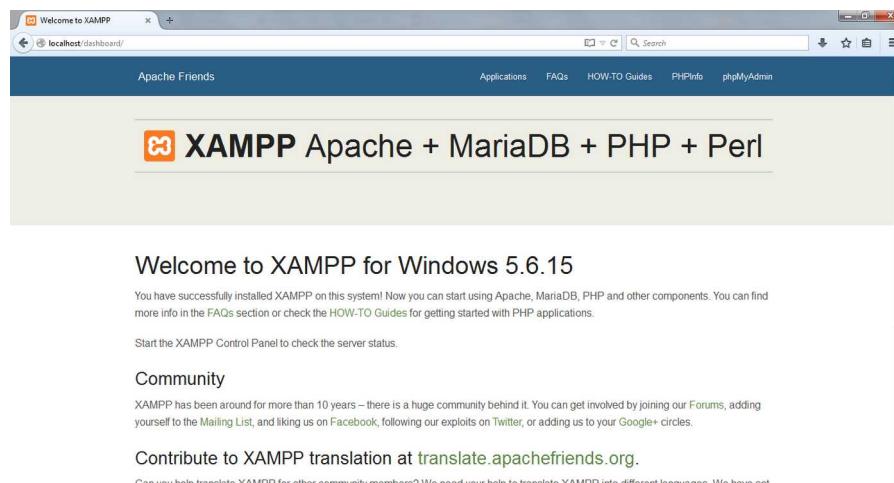
Untuk menguji instalasi XAMPP, silahkan klik tombol **START** pada modul **Apache** dan **MySQL**. Jika tidak ada masalah, akan tampil warna hijau pada kolom **module** ini, seperti tampilan dibawah:



Gambar: Cara menjalankan Apache web server dan MySQL Server

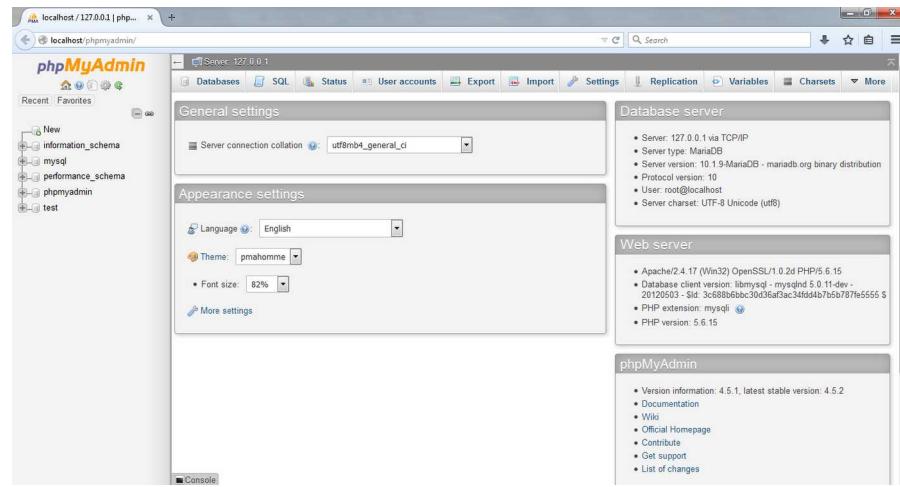
Tombol START akan berubah menjadi tombol STOP. Tombol stop digunakan untuk mematikan Apache web server dan MySQL database server.

Selanjutnya, buka web browser dan ketikkan <http://localhost> pada address bar, kemudian tekan enter. Jika tampil jendela pembuka XAMPP, maka semuanya telah terinstall dengan baik.



Gambar: Tampilan halaman localhost XAMPP

Untuk menguji apakah database MySQL (atau tepatnya MariaDB) sudah terkoneksi dan bisa diakses, klik menu **phpMyAdmin** di sudut kanan atas, atau ketik alamat <http://localhost/phpmyadmin>. Jika tampil jendela berikut, berarti Database sudah terhubung dengan Apache Web Server dan PHP:



Gambar: Tampilan halaman phpMyAdmin XAMPP

## 3.4 XAMPP Troubleshooting

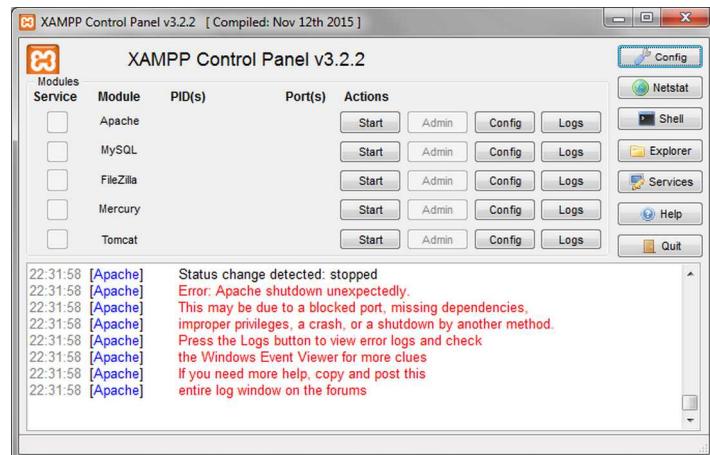


Bagian ini khusus bagi anda yang gagal menginstall XAMPP. Jika halaman home localhost sebelumnya sudah tampil sukses, anda bisa melewati pembahasan ini.

Tutorial cara menginstall XAMPP sebelum ini juga sudah ada di DuniaIlkom, mayoritas komentar yang ada terkait gagalnya proses instalasi XAMPP. Oleh karena itu saya akan membahas sedikit masalah yang mungkin terjadi dan solusinya.

### Terdapat Aplikasi Lain yang Menggunakan Port Apache dan MySQL.

Ini adalah sumber masalah yang sangat sering terjadi, dimana aplikasi XAMPP (yakni web server Apache dan MySQL) bentrok dengan aplikasi sejenis. Apakah sebelumnya anda pernah menginstall XAMPP atau aplikasi AMP stack lain? Jika pernah, sedapat mungkin cari lokasi instalasinya dan **uninstall**.



Gambar: XAMPP error karena port 80 sudah dipakai oleh aplikasi lain

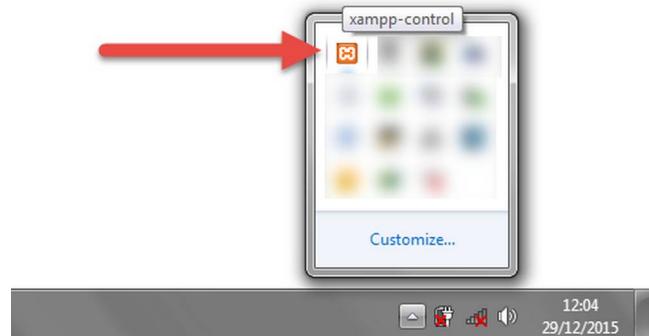
Aplikasi lain yang bisa bentrok dengan apache adalah **Skype** dan **TeamViewer**. Kedua aplikasi ini menggunakan port 80 yang juga merupakan port default Apache. Jika dikomputer anda terinstall aplikasi ini, matikan terlebih dahulu selama menjalankan XAMPP.

Apabila anda ragu atau tidak menemukan aplikasi ini, bisa dicari dari system Windows. Tutorialnya sudah saya tulis di duniaIlkom: [Cara mengatasi error XAMPP: Port 80 in use by “Unable to open process”](#).

## Aplikasi XAMPP Lain Sudah Berjalan

Ketika anda men-klik icon close (tanda silang disudut kanan atas) *XAMPP Control Panel*, ini bukan berarti menutup *XAMPP Control Panel*, tetapi sebenarnya akan tersimpan ke *tray icon* (icon di kanan bawah jendela windows, dekat penunjuk waktu dan tanggal).

Jika anda sudah menjalankan **Apache server** dan **MySQL server**, keduanya juga masih tetap aktif. Dengan men-klik menu tray icon ini, XAMPP Control Panel juga akan muncul kembali.



Gambar: XAMPP Control Panel sebenarnya masih aktif di tray icon

Sangat mungkin anda menyangka bahwa XAMPP Control Panel telah tertutup dan men-klik kembali shortcut icon *XAMPP Control Panel* di desktop atau Start Menu. Hal mengakibatkan ada 2 atau 3 XAMPP Control Panel yang jalan bersamaan.

<sup>9</sup><http://www.duniaIlkom.com/cara-mengatasi-error-xampp-port-80-in-use-by-unable-to-open-process/>

Ketika anda menjalankan Apache (klik tombol Start), akan tampil error. Karena (tentu saja) Apache sudah berjalan yang berasal dari XAMPP Control Panel sebelumnya. Silahkan periksa tray icon untuk memastikan.

Atau bisa juga test mengunjungi halaman <http://localhost/> Jika tampilan halaman welcome dari XAMPP, berarti web server apache sebenarnya sudah aktif.



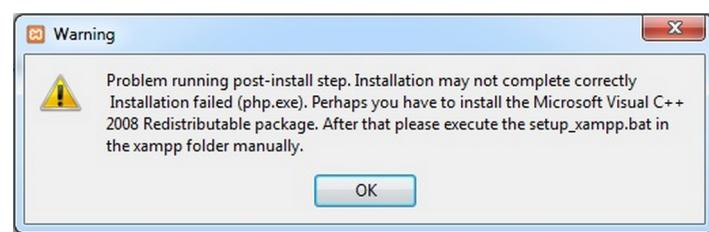
Gambar: Ada 3 XAMPP yang jalan bersamaan !

## Mengubah Port Default Apache dan MySQL

Jika 2 solusi diatas tidak berhasil, anda juga bisa menukar port yang digunakan oleh Apache dan MySQL. Solusi ini saya sarankan bagi yang anda yang tidak keberatan dengan utak-atik konfigurasi XAMPP: [Tutorial Cara Mengganti Port Apache dan MySQL di XAMPP<sup>10</sup>](#)

## Microsoft .NET Framework Belum Terinstall

Apabila anda tidak menemukan program yang menggunakan port yang sama dengan Apache dan MySQL, bisa jadi error ini disebabkan Microsoft .NET Framework belum terinstall. Jika ini penyebabnya, pada saat instalasi XAMPP biasanya juga ada pesan error seperti ini:



Gambar: Error XAMPP karena Net Framework tidak terdeteksi / belum diinstall

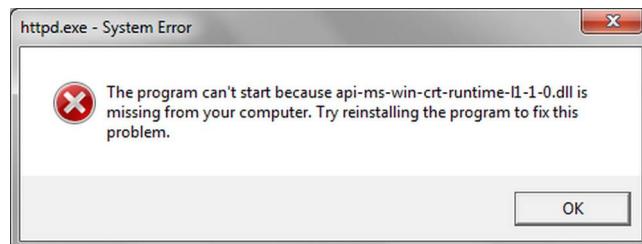
Solusinya, download dan install *Visual C++ Redistributable Packages for Visual Studio 2013* dari situs [microsoft.com<sup>11</sup>](https://microsoft.com). Setelah itu, jalankan `setup_xampp.bat` di dalam folder instalasi XAMPP.

<sup>10</sup><http://www.dunialkom.com/tutorial-cara-mengganti-port-apache-dan-mysql-di-xampp>

<sup>11</sup><https://www.microsoft.com/en-us/download/details.aspx?id=40784>

## Microsoft .NET Framework 2015 untuk XAMPP 7.0.2

Ini adalah error yang saya alami ketika mencoba menginstall XAMPP 7.0.2 di Windows 7. Pada saat proses instalasi semuanya berjalan lancar, tetapi ketika menjalankan web server Apache, tampil jendela error:



Gambar: Error api-ms-win-crt-runtime-1-1-0-dll

Solusinya adalah dengan menginstall [Visual C++ Redistributable for Visual Studio 2015](#)<sup>12</sup>.

Masalahnya, “*Visual C++ Redistributable for Visual Studio 2015*” juga menolak ketika diinstall. Dari [sumber](#)<sup>13</sup> yang saya temukan, kita harus update Windows dulu, lalu install kembali. Apabila cara ini tidak memungkinkan, silahkan install **XAMPP 5.6.15**.

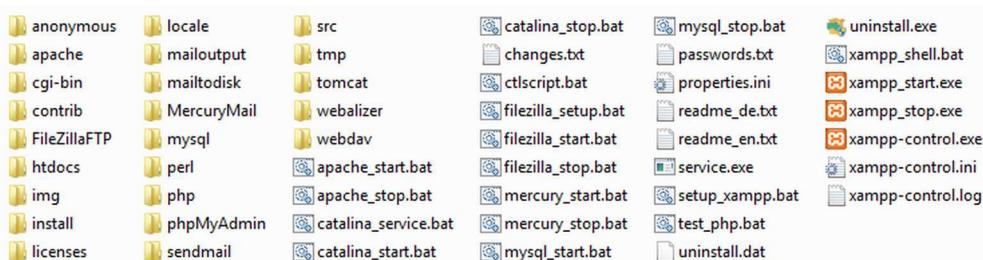
## Install Ulang Windows

Ini adalah solusi paling extreme sebagai harapan terakhir. Lakukan hanya jika langkah-langkah diatas gagal dan anda tidak bisa menemukan solusinya di Google :)

## 3.5 Melihat Folder Instalasi XAMPP

Seperti yang telah kita bahas, XAMPP pada dasarnya ‘mengumpulkan’ berbagai aplikasi yang diperlukan untuk menjalankan PHP. Aplikasi ini ditempatkan dalam folder instalasi XAMPP. Mari kita lihat beberapa diantaranya.

Silahkan buka folder instalasi XAMPP. Dalam contoh sebelumnya, saya menginstall XAMPP di C:\xampp\. Di dalam folder inilah berbagai file dan folder penyusun XAMPP berada. Sebagian besar diantaranya tidak perlu kita ‘ganggu’.



Gambar: Isi folder Instalasi XAMPP

<sup>12</sup><https://www.microsoft.com/en-us/download/details.aspx?id=48145>

<sup>13</sup><http://stackoverflow.com/questions/33265663/api-ms-win-crt-runtime-l1-1-0-dll-is-missing-when-open-office-file?lq=1>

Dalam folder utama terdapat file **xampp-control.exe** ini tidak lain adalah program yang digunakan untuk menjalankan XAMPP Control Panel. File **xampp\_start.exe** dan **xampp\_stop.exe** digunakan untuk menjalankan XAMPP dari cmd (command prompt) Windows. Selama kita bisa mengakses XAMPP Control Panel, kedua file ini tidak perlu dijalankan.

## Folder htdocs

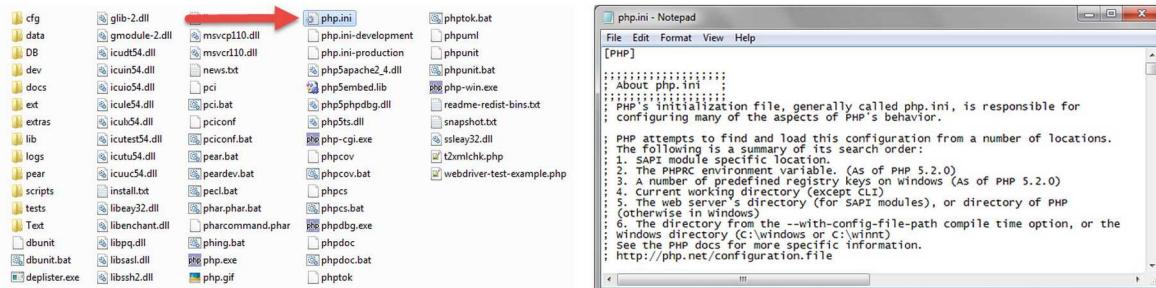
Folder **htdocs** bisa jadi merupakan folder yang paling penting dari seluruh folder lain di dalam XAMPP. Karena, disinilah file PHP harus ditempatkan. Jika file PHP disimpan di dalam folder lain (walaupun masih di dalam folder instalasi xampp), kode program tersebut TIDAK AKAN BERJALAN.

Saya akan kembali membahas penggunaan folder **htdocs** dalam bab berikutnya.

## Folder php

Folder **php** adalah tempat dimana seluruh file inti PHP berada. Jika anda mendownload aplikasi PHP secara manual dari [php.net](http://php.net)<sup>14</sup>, isi dari aplikasi PHP itu sama dengan folder **C:\xampp\php**.

Di dalam folder **php**, terdapat 1 file penting: **php.ini**. File **php.ini** berisi seluruh settingan terkait PHP. File **php.ini** hanyalah file teks biasa yang bisa dibuka dan diedit menggunakan Notepad++.



Gambar: File **php.ini** yang berisi seluruh konfigurasi PHP

## Folder apache

Folder **apache** berisi seluruh file apache web server. Sama seperti **php**, jika anda mendownload aplikasi apache dari situs resminya <http://httpd.apache.org><sup>15</sup>, isinya akan sama dengan folder **C:\xampp\apache**. File konfigurasi apache, **httpd.conf** berada di folder **C:\xampp\apache\conf**.

## Folder mysql

Folder terakhir yang perlu diketahui adalah **mysql**. Seperti yang sudah bisa anda tebak, folder ini berisi seluruh file-file MySQL Server, termasuk database yang akan kita buat. File konfigurasi MySQL terdapat di file **my.ini** dalam folder **C:\xampp\apache\mysql\bin**.

<sup>14</sup>[php.net](http://php.net)

<sup>15</sup>[https://httpd.apache.org](http://httpd.apache.org)

Jika anda teliti lebih jauh, seluruh file di dalam folder **mysql** memiliki nama **mysql**, padahal sebenarnya ini adalah file **MariaDB**. Pada dasarnya, MariaDB menggunakan source code yang sama dengan mysql, oleh karena itu pula nama file dan foldernya tetap bernama mysql. Hal ini sangat memudahkan kita untuk beralih dari MySQL ke MariaDB.

## 3.6 Instalasi Web Browser

Aplikasi kedua yang dibutuhkan untuk menjalankan kode PHP, tentu saja: **web browser**. Tidak ada ketentuan khusus untuk web browser ini. Anda bisa menggunakan web browser apapun.

Dalam buku ini saya akan menggunakan web browser **Mozilla Firefox**, **Google Chrome**, dan **Opera** secara bergantian. Jika memungkinkan, update web browser anda ke versi yang paling baru.

## 3.7 Instalasi Text Editor

File PHP pada dasarnya hanyalah file teks biasa dengan aturan penulisan khusus. Untuk menulis kode PHP, kita bisa menggunakan aplikasi teks editor apapun. Bahkan, aplikasi **Notepad** bawaan Windows juga sudah mencukupi.

Dalam buku **HTML Uncover**, saya menggunakan aplikasi **Notepad++**. Sedangkan dalam buku **CSS Uncover**, saya memilih **Komodo Edit**. Kedua teks editor ini sudah lebih dari cukup untuk mengetik kode PHP.

Aplikasi **Notepad++** bisa di download dari <https://notepad-plus-plus.org/download><sup>16</sup>. Sedangkan **Komodo Edit** bisa di download dari <http://komodoide.com/komodo-edit><sup>17</sup>. Proses instalasi keduanya relatif mudah. Anda tinggal klik tombol **Next** beberapa kali, dan keduanya siap digunakan. **Notepad++** dan **Komodo Edit** sama-sama bersifat open source dan bisa digunakan dengan gratis.

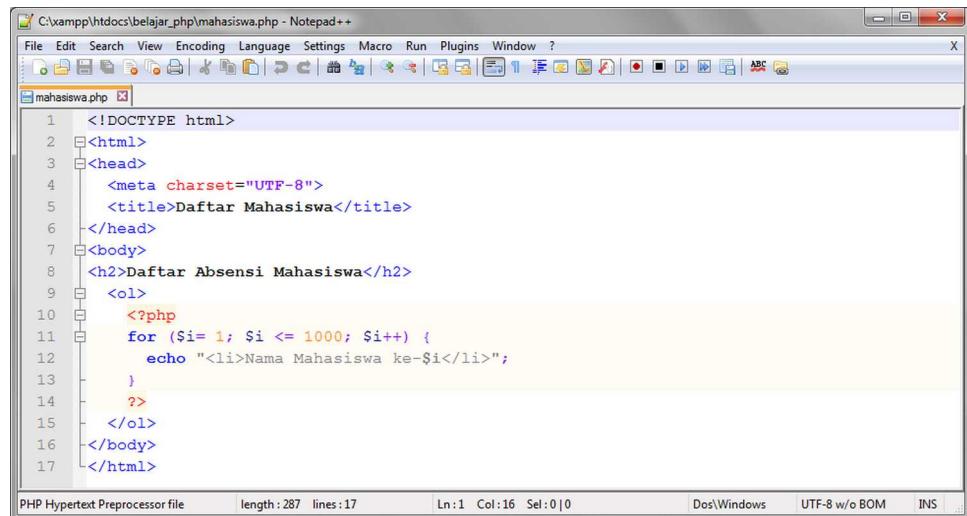
### Notepad++

**Notepad++** ringan dan cepat (hanya berukuran 20MB setelah diinstall). Teks editor ini sangat populer di kalangan web developer. Fitur-fitur penting juga sudah tersedia di dalam Notepad++, seperti penomoran baris (*line numbering*), pewarnaan kode program (*syntax hilighting*), serta *code completion*.

---

<sup>16</sup><https://notepad-plus-plus.org/download>

<sup>17</sup><http://komodoide.com/komodo-edit>



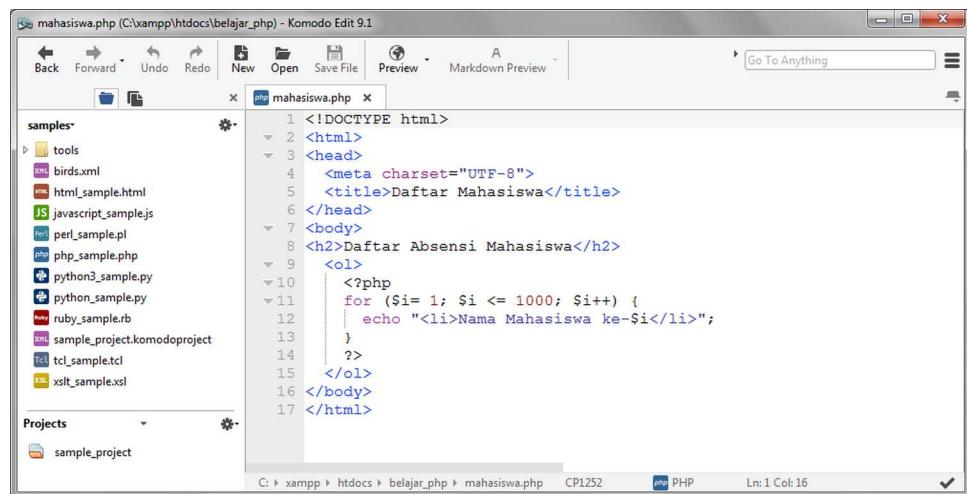
Gambar: Tampilan Aplikasi Text Editor Notepad++

Salah satu kelemahan dari **Notepad++** adalah fitur pewarnaan kode program (*syntax highlighting*) hanya mendukung 1 bahasa dalam 1 file. Ketika dalam file yang sama terdapat kode program HTML, CSS dan PHP, Notepad++ tidak bisa ‘mewarnai’ semua bahasa.

## Komodo Edit

Dibandingkan Notepad++, **Komodo Edit** memang belum sepopuler Notepad++. Selain itu, aplikasi ini juga cukup berat (berukuran sekitar 200MB setelah diinstall).

Namun Komodo Edit memiliki fitur yang lebih baik dan tampilan yang lebih rapi. Ketika kita menulis berbagai bahasa pemrograman dalam 1 file, semuanya diwarnai dengan baik.



Gambar: Tampilan Aplikasi Text Editor Komodo Edit

Selain itu, Komodo Edit memiliki fitur *live debugging*. Fitur ini akan menandai kode program yang salah sebelum kode tersebut dijalankan. Hal ini sangat membantu walaupun kadang-kadang sedikit menyebalkan. Pesan kesalahan sudah tampil duluan sebelum kita selesai menulis kode program :)

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Daftar Mahasiswa</title>
6 </head>
7 <body>
8 <h2>Daftar Absensi Mahasiswa</h2>
9 <ol>
10 <?php
11 for ($i = 1; $i <= 1000 $i++); // Line 11
12 echo "<li>Nama Mahasiswa ke-$i</li>"; // Line 12
13 }
14 ?>
15 </ol>
16 </body>
17 </html>

```

Gambar: Fitur Live Debugging Komodo Edit

Untuk mengedit kode program PHP dengan cepat, saya biasanya menggunakan Notepad++. Tapi jika butuh membuat aplikasi PHP yang cukup panjang dan lebih dari 1 file, saya lebih suka menggunakan Komodo Edit.

## 3.8 Text Editor Alternatif

Komodo Edit dan Notepad++, hanyalah *'salah dua'* teks editor gratis yang bisa digunakan. Sebagai pilihan lain (yang juga gratis) anda bisa mencoba Bracket<sup>18</sup>, Atom<sup>19</sup>, Aptana Studio<sup>20</sup>, Eclipse<sup>21</sup> atau Net Beans<sup>22</sup> .

Anda bebas ingin menggunakan editor apa saja, tidak harus Notepad++ atau Komodo Edit. Saya juga menyarankan untuk mencoba semua aplikasi teks editor diatas, dan pilih yang menurut anda paling nyaman.

### Teks Editor Premium

Selain teks editor gratisan diatas, juga terdapat teks editor berbayar dengan fitur melimpah. Yang cukup populer adalah Sublime Text<sup>23</sup> dan PHPStorm<sup>24</sup>.

### Bagaimana dengan Adobe Dreamweaver?

Salah satu aplikasi populer yang cukup ‘melegenda’ terutama bagi pemula web programming adalah Adobe Dreamweaver. Dreamweaver merupakan aplikasi canggih dengan fitur melimpah untuk pembuatan web.

Dreamweaver termasuk kelompok aplikasi yang dikenal dengan sebutan WYSIWYG (What You See Is What You Get), dimana kita bisa merancang tampilan website dengan cara ‘*drag and drop*’, yakni menggambar tampilan web secara visual tanpa harus mengetahui kode program dibalik itu (walaupun dreamweaver juga menyediakan fitur coding yang sangat lengkap).

<sup>18</sup><http://brackets.io>

<sup>19</sup><https://atom.io>

<sup>20</sup><http://www.aptana.com>

<sup>21</sup><https://eclipse.org>

<sup>22</sup><https://netbeans.org>

<sup>23</sup><http://www.sublimetext.com>

<sup>24</sup><https://www.jetbrains.com/phpstorm/>

Selain keunggulannya, menurut saya Dreamweaver tidak cocok untuk proses belajar. Aplikasi ini cenderung ‘berat’ dan berharga jutaan rupiah untuk versi legalnya. Dreamweaver lebih pas digunakan jika anda telah memahami kode-kode program yang ada dan mampu membeli versi aslinya.

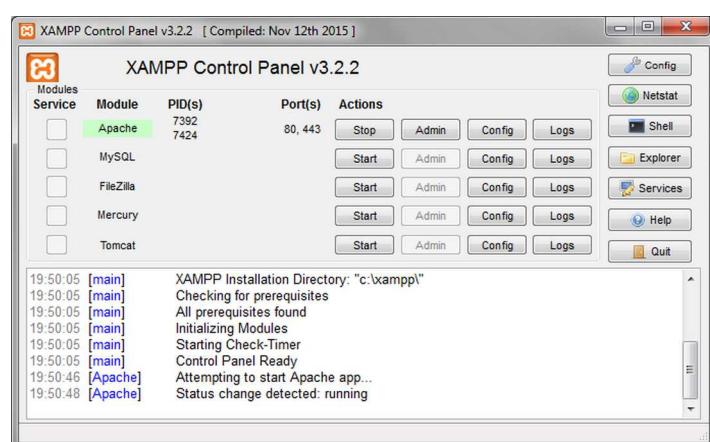
---

Dalam bab ini kita telah mempersiapkan ‘perlengkapan perang’ yang terdiri dari aplikasi XAMPP, web browser, dan teks editor. Selanjutnya, kita akan langsung praktik untuk menjalankan kode PHP.

# 4. Menjalankan File PHP

Setelah semua aplikasi yang dibutuhkan tersedia, dalam bab ini saya akan memandu anda untuk mulai membuat kode PHP. Kita akan membahas cara menjalankan PHP dan melihat bagaimana PHP diproses untuk menghasilkan halaman web.

Hal pertama yang wajib dilakukan sebelum menjalankan kode PHP adalah memastikan **Apache web server** sudah berjalan. Silahkan buka **XAMPP Control Panel** dan klik tombol **Start** pada kolom **Action**.



Gambar: Jalankan Apache web server dengan klik tombol Start

Ketika pertama kali di-klik, akan berwarna kuning beberapa saat, baru berubah menjadi hijau. Jika berwarna merah berarti web server apache gagal berjalan. Pastikan latar belakang tulisan apache sudah berwarna **hijau**.



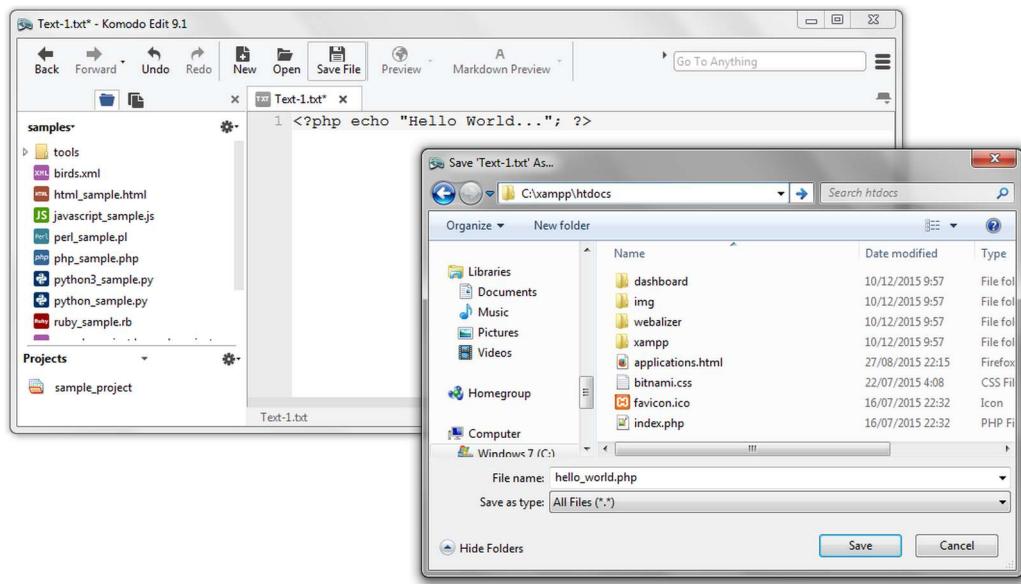
Dalam beberapa bab kedepan, kita belum membutuhkan MySQL. Karena itu tidak perlu menjalankan MySQL server, cukup Apache saja.

## 4.1 Menampilkan Teks Hello World dengan PHP

Sudah menjadi tradisi di dunia pemrograman untuk menampilkan teks **“Hello World”** ketika mempelajari bahasa pemrograman baru. Dan inilah yang akan kita lakukan. Silahkan buka aplikasi teks editor **Notepad++** atau **Komodo Edit** (atau aplikasi text editor lain). Lalu ketik kode berikut ini:

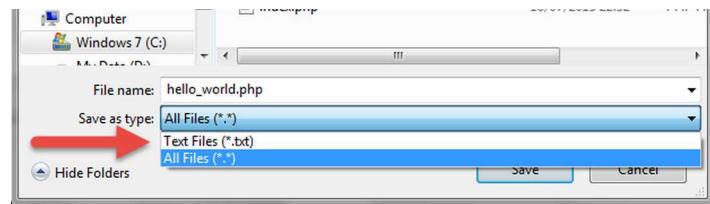
```
<?php echo "Hello World..."; ?>
```

Kemudian tekan kombinasi tombol **Ctrl+S** atau pilih menu **File -> Save** untuk menyimpan file. Dari jendela **Save As**, carilah folder **htdocs** yang berada di **C:\xampp**, kemudian save sebagai **hello\_world.php** seperti yang terlihat pada gambar dibawah ini:



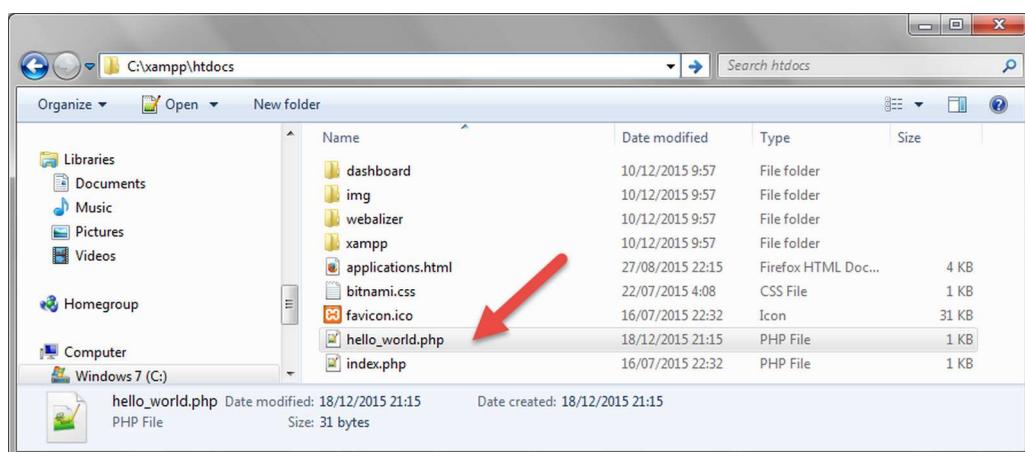
Gambar: Simpan file hello\_world.php pada folder C:\xampp\htdocs\

Untuk beberapa teks editor, ubah pilihan “Save as type” dari “Text Files (\*.txt)” menjadi “All Files (\*.\*)”. Ini untuk menghindari file tersimpan sebagai hello\_world.php.txt. Jika ini terjadi, hapus akhiran .txt dari nama file.



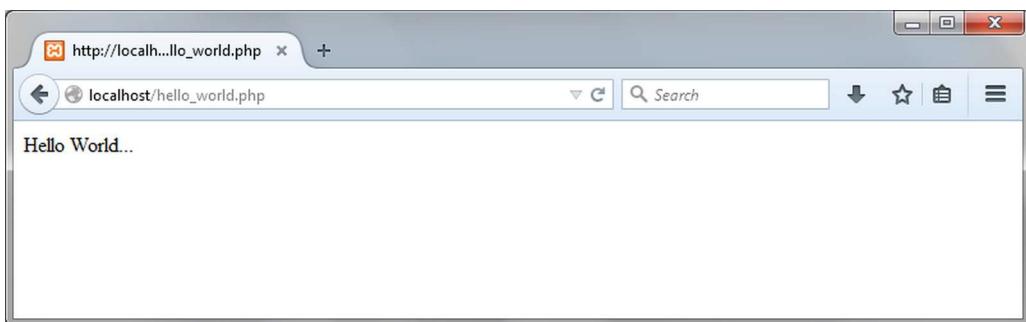
Gambar: Tukar ‘Save as type’ menjadi ‘All Files (.)’

Untuk lebih memastikan, silahkan buka folder C:\xampp\htdocs\ . Seharusnya akan terdapat file hello\_world.php di dalam folder ini.



Gambar: Pastikan file hello\_world.php sudah berada di C:\xampp\htdocs

Kemudian buka web browser, ketikkan alamat: **localhost/hello\_world.php** dan tekan Enter. Jika semuanya sesuai maka akan tampil text “Hello World...” di dalam web browser.



Gambar: Hello World...

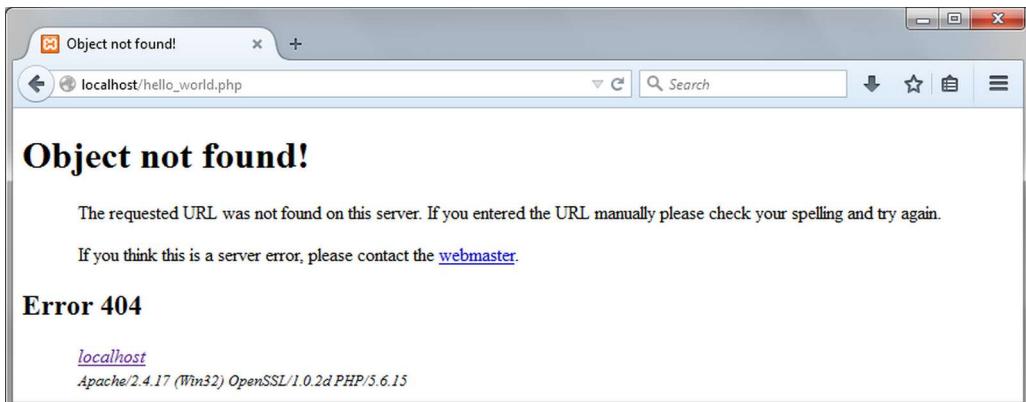
**Selamat!** Anda telah berhasil menjalankan kode PHP pertama Anda!

Dari contoh file **hello\_world.php** ini, kita dapat melihat bagaimana cara menulis dan menjalankan sebuah file PHP:

- Setiap file PHP harus disimpan dengan extension **\*.php**.
- PHP diawali dengan kode `<?php` dan ditutup dengan kode `?>`.
- Perintah **echo** digunakan untuk menampilkan teks ke dalam web browser.
- Seluruh file PHP harus berada di dalam folder **htdocs**.
- Untuk menjalankan file PHP, dapat diakses dengan alamat **localhost/nama\_file.php**.

Saya akan membahas lebih jauh tentang cara kerja PHP di dalam bab ini.

## Bagaimana jika hasilnya seperti ini?



Gambar: Error, Object not found!

Sesuai dengan pesan error: “**Object not found!**”, artinya web server apache tidak bisa menemukan file tersebut.

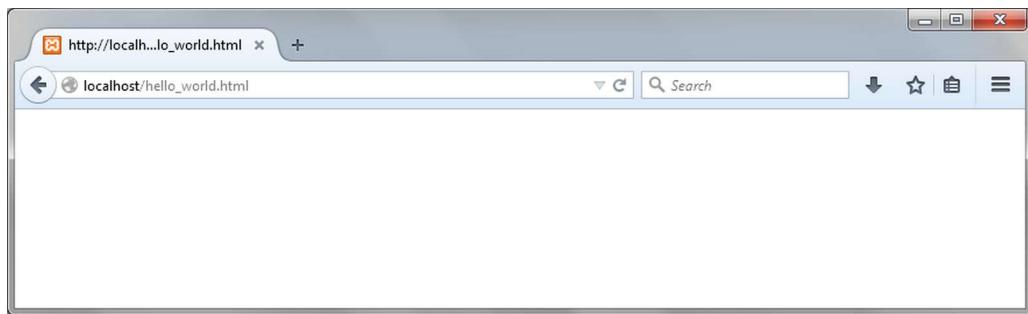
Terdapat beberapa kemungkinan, yang pertama: *salah tulis nama file*. Dalam contoh kita, saya membuat nama file sebagai **hello\_world.php**. Jika anda men-savenva dengan nama **helloworld.php**, maka alamat file menjadi tidak cocok, dan akan tampil error “**object not found**”.

Kemungkinan kedua: *salah tulis alamat file*. Jika file aslinya adalah **hello\_world.php**, namun diakses dengan **http://localhost/helloworld.php**, tentu juga tidak akan ditemukan.

Kemungkinan ketiga: *salah lokasi penyimpanan*. Kalau yang ini cukup sering saya alami. Dalam contoh ini saya menempatkan file `hello_world.php` langsung di dalam folder `htdocs`. Kadang tidak sengaja saya menyimpannya di dalam subfolder seperti `htdocs\folder_-php\hello_world.php`.

Kemungkinan keempat: *salah folder xampp*. Nama file sudah cocok, nama folder pun sudah pas, kenapa masih error? Ternyata saya menempatkan file tersebut di folder `htdocs` XAMPP lain. Ini sering terjadi jika anda menginstall lebih dari 1 aplikasi XAMPP pada komputer yang sama.

## Bagaimana jika tampilannya seperti berikut ini?



Gambar: Tidak ada error, tapi kenapa kosong?

Halaman diatas terlihat kosong tanpa kode error apapun. Dalam beberapa kasus, yang tampil adalah kode PHP yang kita buat (termasuk tulisan “`echo`”).

Hasil seperti ini cukup sering ditanyakan di dunia ilkom. Dapatkah anda melihat dimana letak kesalahannya? Fokuskan ke dalam alamat file di web browser. Yup, nama file adalah `hello_-world.html`. Agar kode program PHP dapat diproses, kita **harus** menyimpannya ke dalam file berakhiran `.php`, bukan `.html`. File diatas seharusnya disimpan sebagai `hello_world.php`, bukan `hello_world.html`.

Kesalahan seperti ini juga bisa terjadi jika anda mengakses `hello_world.php` bukan dari alamat `localhost`, tapi dari alamat foldernya, seperti file: `:///C:/xampp/htdocs/hello_world.php`.

## Apa itu Localhost?

Dalam percobaan menjalankan kode PHP diatas, kita mengetik alamat `http://localhost/hello_-world.php`. Kenapa harus ditulis dengan **localhost**?

Dalam jaringan komputer, **localhost** berarti “komputer itu sendiri”. Karena kita menjalankan web server di komputer lokal, maka harus diakses menggunakan alamat **localhost**.

Selain itu, **localhost** juga bisa diganti menjadi alamat IP: **127.0.0.1**. File `hello_world.php` sebelumnya, juga bisa diakses dengan mengetik alamat: `http://127.0.0.1/belajar_php/hello_-world.php`.

Dalam internal web server Apache XAMPP, **localhost** digunakan untuk mengakses isi folder `htdocs`. Misalkan saya punya file `semangat.php` di alamat `C:\xampp\htdocs\satu\dua\semangat.php`. Untuk mengakses file ini, ganti bagian `C:\xampp\htdocs` dengan **localhost**, sehingga alamat di web browser adalah: `http://localhost/satu/dua/semangat.php`.

## 4.2 PHP Untuk Menghasilkan HTML

Seperti yang telah kita pelajari, web browser hanya dapat memproses dan menampilkan kode HTML, CSS atau JavaScript (kelompok *client side programming language*). Kode yang dibuat dengan PHP akan diterjemahkan oleh web server menjadi salah satu dari ketiga bahasa tersebut. Agar lebih mudah dipahami, mari masuk ke dalam contoh berikutnya.

Kali ini buatlah sebuah file baru, kemudian ketikkan kode program berikut:

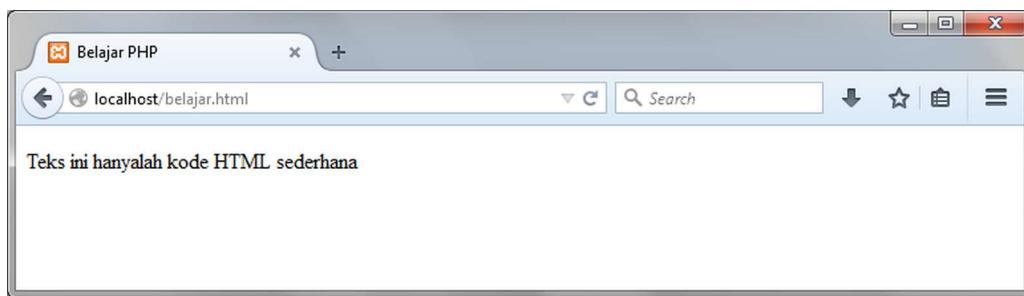
belajar.html

---

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar PHP</title>
6 </head>
7 <body>
8   <p>Teks ini hanyalah kode HTML sederhana</p>
9 </body>
10 </html>
```

---

Savelah kode diatas sebagai **belajar.html** di dalam folder **htdocs**. Untuk mengaksesnya, anda bisa mengetik alamat: **localhost/belajar.html** seperti yang terlihat dalam gambar berikut:



Gambar: Tampilan kode program belajar.html

Kode program yang kita ketik diatas hanyalah file HTML sederhana tanpa ada kode PHP sama sekali. Perhatikan extension dari file tersebut adalah **.html**. Web server Apache tidak hanya bisa menjalankan file dengan extension **.php** saja, tetapi juga extension lain seperti **.html**.

Mari kita tambahkan sedikit kode program PHP kedalam HTML diatas.

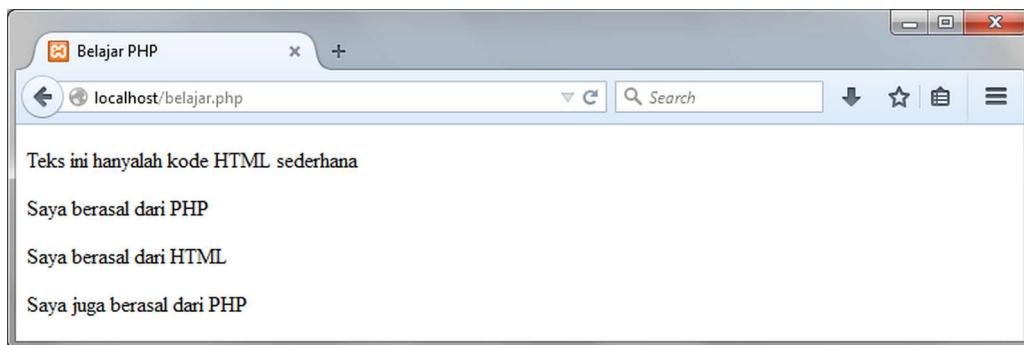
**belajar.php**

---

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar PHP</title>
6 </head>
7 <body>
8   <p>Teks ini hanyalah kode HTML sederhana</p>
9   <?php echo "<p>Saya berasal dari PHP</p>"; ?>
10  <p>Saya berasal dari HTML</p>
11  <?php echo "<p>Saya juga berasal dari PHP</p>"; ?>
12 </body>
13 </html>
```

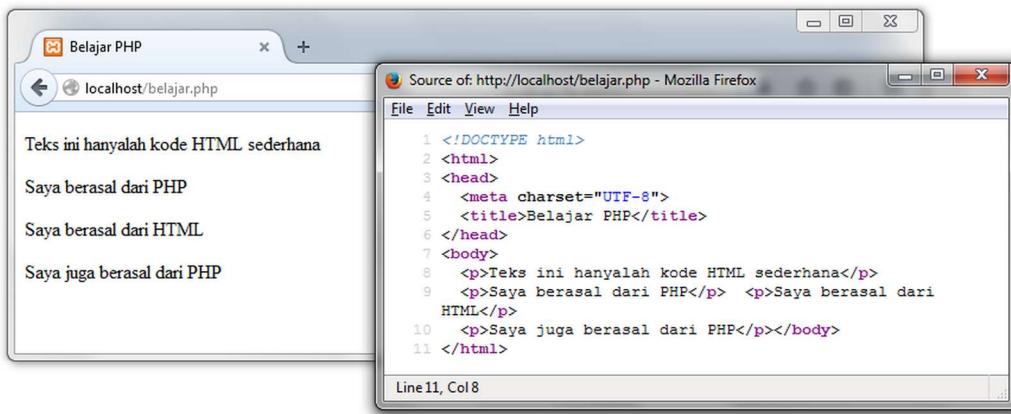
---

Kali ini, simpan kembali (Save As) kode program diatas sebagai **belajar.php** di **htdocs**, kemudian akses dengan alamat **localhost/belajar.php**. Hasilnya akan terlihat seperti gambar dibawah:



Gambar: Tampilan kode program belajar.php

Dari kode program diatas, anda dapat melihat bahwa saya menyisipkan kode PHP kedalam kode HTML. Perintah **echo** dari PHP akan menghasilkan kode HTML `<p>Saya berasal dari PHP</p>`. Jika Anda melihat kode HTML yang dihasilkan oleh web server, akan tampak seperti gambar berikut:



Gambar: Kode HTML yang dihasilkan dari belajar.php

Dari gambar diatas, terlihat bahwa seluruh kode PHP yang kita buat, ‘diterjemahkan’ menjadi kode HTML oleh web server, kemudian dikirim ke dalam web browser untuk ditampilkan. Bagaimana cara kerjanya? Mari kita bahas dengan lebih detail.

### 4.3 Cara PHP diproses di dalam Web Server

Ketika permintaan sebuah halaman dari web browser sampai kepada ke web server (Apache), web server akan mengambil keputusan apakah akan memprosesnya terlebih dahulu, atau tinggal mencari file tersebut dan langsung mengirimnya ke web browser. Ini tergantung dari jenis file yang diminta.

Menggunakan contoh sebelumnya, ketika saya mengetik alamat: **localhost/belajar.html**, web browser akan mengirim permintaan (*request*) file **belajar.html** kepada Apache web server.

Saat web server menerima *request* ini, ia akan mencari file **belajar.html** di dalam folder **htdocs**. Karena file yang diminta berupa file **html**, web server tidak memerlukan proses lanjutan dan tinggal mengirim balik file tersebut kepada web browser untuk ditampilkan.

Tetapi bagaimana dengan **localhost/belajar.php**? Karena saya ‘meminta’ file PHP, web server akan melakukan proses terlebih dahulu (perhatikan bahwa file yang diminta berakhiran “.php”)

Setelah mendapatkan file **belajar.php** di dalam folder **htdocs**, web server akan memproses file tersebut dengan mencari seluruh tag pembuka dan penutup PHP (karakter `<?php` dan `?>`). Proses ini dilakukan dari baris paling atas, hingga baris paling bawah secara berurutan.

Ketika karakter `<?php` ditemukan, web server akan beralih ke “**PHP mode**”, kemudian mengeksusi seluruh kode program menggunakan aturan PHP, sampai ditemukan tag penutup PHP `?>`.

Apabila di dalam halaman tersebut terdapat lebih dari 1 karakter `<?php` dan `?>` (yang berarti kode PHP dibuat secara terpisah), web server juga akan memprosesnya. Hal ini terus dilakukan hingga tidak ada lagi perintah PHP yang belum dijalankan. Hasil dari proses inilah yang selanjutnya dikirim ke web browser.



Penjelasan ini mungkin terasa rumit, tapi sangat penting untuk dipahami. Jika anda masih ragu, silahkan dibaca dengan perlahan :)

## 4.4 Menghasilkan HTML Dinamis dengan PHP

Kode PHP yang telah kita buat (`belajar.php`) dapat menggambarkan bagaimana PHP diproses untuk menghasilkan kode HTML. Tetapi kode itu hanya menampilkan sebuah teks yang sebenarnya bisa dibuat dengan lebih mudah tanpa menggunakan PHP sama sekali. Kali ini kita akan lihat bagaimana PHP dapat mempermudah pembuatan HTML.

Sebagai contoh, Berikut kode HTML untuk menampilkan data 10 orang peserta kegiatan dalam bentuk tabel:

`peserta.html`

---

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar PHP</title>
6 </head>
7 <body>
8 <h2>Daftar Nama Peserta</h2>
9 <table border="1" cellspacing="0" cellpadding="4">
10  <tr><th>No.</th><th>Nama Peserta</th></tr>
11  <tr><td>1</td><td>Nama Peserta 1</td></tr>
12  <tr><td>2</td><td>Nama Peserta 2</td></tr>
13  <tr><td>3</td><td>Nama Peserta 3</td></tr>
14  <tr><td>4</td><td>Nama Peserta 4</td></tr>
15  <tr><td>5</td><td>Nama Peserta 5</td></tr>
16  <tr><td>6</td><td>Nama Peserta 6</td></tr>
17  <tr><td>7</td><td>Nama Peserta 7</td></tr>
18  <tr><td>8</td><td>Nama Peserta 8</td></tr>
19  <tr><td>9</td><td>Nama Peserta 9</td></tr>
20  <tr><td>10</td><td>Nama Peserta 10</td></tr>
21 </table>
22 </body>
23 </html>
```

---

Savelah kode tersebut sebagai `peserta.html` di dalam folder `htdocs`. Anda bisa menjalankannya dengan alamat: `localhost/peserta.html`.

Kode diatas adalah kode HTML sederhana untuk menampilkan sebuah tabel dengan 10 baris. Dapat dilihat bahwa saya mengulang tag `<td>` sebanyak 10 kali untuk membuat struktur tabel. Dengan menggunakan PHP, kode tersebut dapat ditulis dengan lebih efisien:

**peserta.php**

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6  </head>
7  <body>
8  <h2> Daftar Nama Peserta </h2>
9  <table border="1" cellspacing="0" cellpadding="4">
10 <tr><th>No.</th><th>Nama Peserta</th></tr>
11 <?php
12 for ($i=1;$i<=10;$i++)
13 {
14     echo "<tr><td>$i</td><td>Nama Peserta $i</td></tr>";
15 }
16 ?>
17 </table>
18 </body>
19 </html>

```

Savelah kode diatas sebagai **peserta.php** di dalam folder **htdocs**, dan dapat diakses dari alamat: **localhost/peserta.php**.

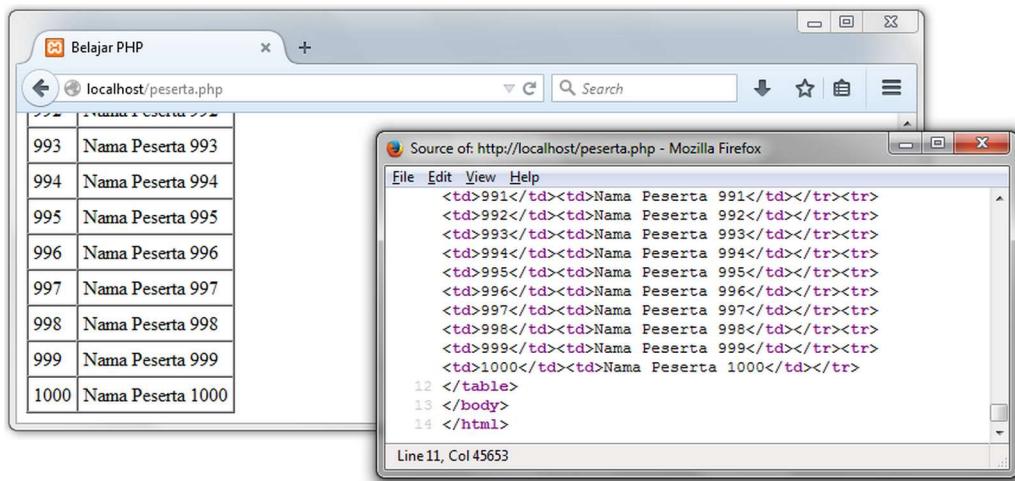
Jika anda menjalankan baik **peserta.html** maupun **peserta.php**, hasilnya akan sama persis seperti yang terlihat pada gambar berikut:

No.	Nama Peserta
1	Nama Peserta 1
2	Nama Peserta 2
3	Nama Peserta 3
4	Nama Peserta 4
5	Nama Peserta 5
6	Nama Peserta 6
7	Nama Peserta 7
8	Nama Peserta 8
9	Nama Peserta 9
10	Nama Peserta 10

Gambar: Tampilan file peserta.php

Walaupun kode PHP yang ada pada contoh diatas belum kita bahas, tetapi anda bisa melihat ‘ide’-nya. Daripada menulis tag **<td>** sebanyak 10 kali, kita tinggal menggunakan perulangan PHP untuk menghasilkan tag **<td>**.

Lebih jauh lagi, jika butuh untuk 1000 baris tabel, saya tinggal mengganti `$i<=10`, menjadi `$i<=1000`, dan tabel HTML dengan 1000 baris langsung dapat ditampilkan, jauh lebih praktis dan efisien.



Gambar: 1000 daftar peserta yang dihasilkan dari file peserta.php



Kolom "Nama Peserta" dalam tabel diatas seharusnya berisi nama asli yang berasal dari database. ini dapat dilakukan dengan mudah menggunakan PHP. Kita akan membahas cara menampilkan data dari database MySQL dalam bab tersendiri.

## 4.5 PHP di dalam HTML, atau HTML di dalam PHP?

Dari beberapa file latihan yang telah kita buat, sedikit banyak anda telah memahami cara menulis kode PHP. Namun jika diperhatikan, saya menulis kode PHP di dalam HTML. Mak-sudnya, saya menyisipkan tag `<?php` dan `?>` diantara kode-kode HTML. Tetapi kita juga bisa melakukan hal sebaliknya, yakni seluruh kode HTML ditulis dengan PHP.

Perhatikan contoh kode PHP berikut ini:

`masih_belajar.php`

---

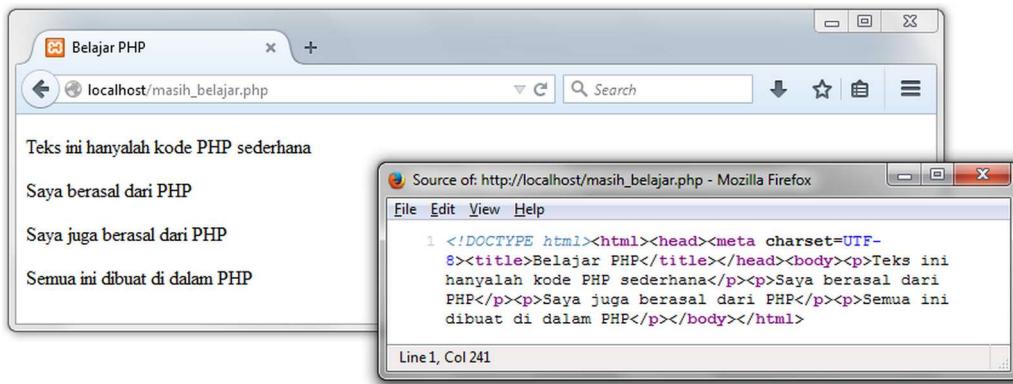
```

1 <?php
2 echo "<!DOCTYPE html>";
3 echo "<html>";
4 echo "<head>";
5 echo "<meta charset=UTF-8>";
6 echo "<title>Belajar PHP</title>";
7 echo "</head>";
8 echo "<body>";
9 echo "<p>Teks ini hanyalah kode PHP sederhana</p>";
10 echo "<p>Saya berasal dari PHP</p>";
11 echo "<p>Saya juga berasal dari PHP</p>";
12 echo "<p>Semua ini dibuat di dalam PHP</p>";

```

```
13 echo "</body>";  
14 echo "</html>";  
15 ?>
```

Savelah sebagai **masih\_belajar.php** di dalam folder **htdocs**, dan akses di alamat: **localhost/-masih\_belajar.php**. Jika anda menjalankan kode diatas, hasilnya adalah sebagai berikut:



Gambar: Tampilan file **masih\_belajar.php**

Inilah yang saya maksud dengan **HTML di dalam PHP**.

Jadi, mana yang sebaiknya digunakan? apakah PHP di dalam HTML (seperti file **belajar.php**), atau HTML di dalam PHP (seperti file **masih\_belajar.php**)?

Menurut saya, yang terbaik adalah PHP di dalam HTML, seperti file **belajar.php**. Kita hanya perlu masuk ke dalam **PHP mode** pada saat dibutuhkan saja. Selain lebih efisien, cara ini membuat penulisan kode menjadi lebih rapi.

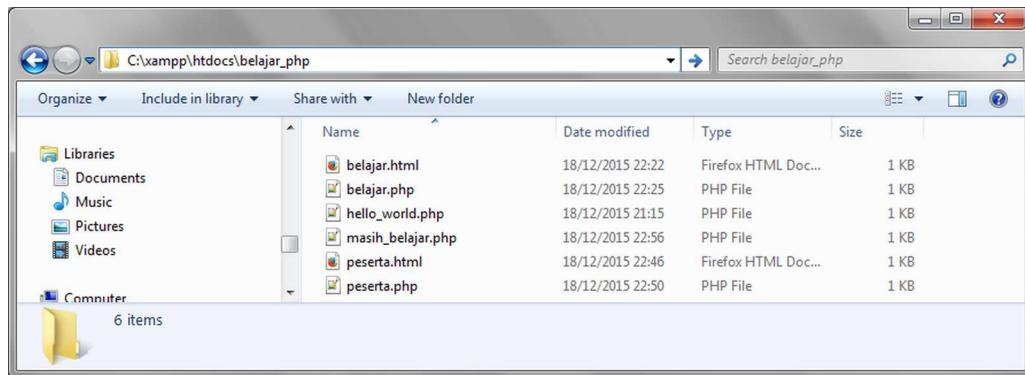
Walaupun demikian, jangan sampai aturan ini menghalangi Anda untuk berkreasi. Apabila sebuah kode program akan lebih mudah dimengerti jika ditulis di dalam PHP, ini pun tidak salah. PHP diproses dengan sangat cepat. Anda mungkin tidak akan melihat perbedaan kecepatan antara kode PHP dengan kode HTML saat diproses oleh web server, kecuali website yang dibangun dikunjungi oleh jutaan pengunjung dan terdiri dari ratusan ribu baris program.

## 4.6 Mempersiapkan Folder **belajar\_php**

Sampai disini saya yakin Anda telah paham cara menjalankan file PHP dan cara menampilkan-nya ke dalam web browser.

Di dalam bab ini kita telah membuat beberapa file PHP, tetapi file itu ‘bergabung’ dengan file-file bawaan XAMPP. Agar lebih rapi, buatlah sebuah folder baru di dalam **htdocs** dengan nama: **belajar\_php**. Kemudian pindahkan seluruh file yang telah kita buat kedalam folder ini.

Karena file tersebut sekarang berada di dalam folder **belajar\_php**, maka untuk mengaksesnya kita harus menambahkan kata “**belajar\_php**” setelah penulisan localhost, menjadi: **localhost/-belajar\_php/belajar.php**.



Gambar: Folder belajar\_php di dalam htdocs XAMPP

Demi menghemat tempat, mulai dari bab selanjutnya saya tidak akan membuat lengkap kode HTML di dalam contoh program, tetapi langsung masuk dengan kode PHP.

Caranya, buatlah sebuah file di dalam folder `htdocs/belajar_php` dengan nama: `index.php`. File `index.php` akan menjadi file ‘sandbox’ atau ‘dummy’ yang bisa anda gunakan sebagai file latihan. File ini dapat diakses dari alamat: `localhost/belajar_php/index.php` atau `localhost/belajar_php/`.



**Index.php** atau **index.html** adalah nama file khusus di dalam web server. Sebuah file dengan nama **index** akan menjadi halaman *default* ketika diakses hanya dengan memanggil nama folder saja (tanpa mencantumkan nama file).

File `index.php` yang berada di dalam folder `htdocs\belajar_php` bisa diakses dengan alamat `localhost/belajar_php/`, tanpa perlu menuliskan secara lengkap menjadi `localhost/belajar_php/belajar.php`.

Sebagai kode template untuk file `index.php` anda bisa menggunakan kode HTML berikut ini:

**index.php**

---

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6  </head>
7  <body>
8      <?php
9          //... kode PHP disini
10         //... kode PHP disini
11         //... kode PHP disini
12     ?>
13     </body>
14 </html>

```

---

Diantara bagian <?php dan ?> inilah nantinya kita akan membuat kode program PHP. Anda tinggal menyisipkan kode PHP, dan menimpanya dengan kode baru.



Seluruh kode program yang ada di dalam buku ini tersedia untuk di download dari folder sharing Google Drive. Silahkan anda extract file **belajar\_php.zip** dan tempatkan di dalam folder **htdocs**.

---

Dalam bab ini kita telah mempelajari cara menjalankan file PHP. Selain itu, saya juga membahas bagaimana cara PHP diproses di dalam web server apache. Berikutnya, kita akan masuk ke Aturan Dasar Penulisan Kode PHP.

# 5. Aturan Dasar Penulisan Kode PHP

Dalam bab ini kita akan mempelajari tentang aturan dasar penulisan kode PHP. Saya akan mulai dengan membahas extension file PHP, PHP tag, perintah echo dan print, PHP statement, case sensitivity, whitespace, membuat komentar, serta cara mengatasi error PHP.

## 5.1 Extension File PHP

Aturan pertama yang akan kita bahas adalah **extension file PHP**. Dengan bahasa yang lebih sederhana hal yang saya maksud adalah: '*file PHP harus disimpan dalam akhiran apa?*'

Setiap halaman yang memiliki kode PHP, **HARUS** di simpan dalam akhiran **.php**. Walaupun di dalam file tersebut terdiri dari 1000 baris kode HTML dan 1 baris kode PHP, tetap harus memiliki akhiran file **.php**, seperti *index.php*, *homepage.php*, atau *register.php*.

Walaupun tidak harus, anda juga bisa menggunakan akhiran **.php** untuk sebuah file HTML yang tidak memiliki kode PHP sama sekali. Karena, mungkin saja di lain waktu kita ingin menambahkan kode PHP ke dalam file tersebut.

Sama seperti akhiran **.html**, nama file PHP tidak boleh mengandung spasi atau karakter khusus lain seperti '?', '\*' atau '&'. Untuk nama file yang terdiri dari 2 suku kata, bisa dipisahkan dengan tanda spasi, seperti: *daftar\_mahasiswa.php* atau *form\_login.php*.



Akhiran **.php** ini sebenarnya bisa diubah dari settingan web server Apache. Misalkan kita ingin agar file PHP bisa disimpan dalam **.php5**, **.php7**, atau bahkan **.pehape**. Namun hal ini tidak umum dilakukan dan perlu pengetahuan teknis tentang konfigurasi Apache.

## 5.2 Mengenal PHP Tag

**PHP tag** adalah sebutan untuk tag yang berfungsi sebagai penanda kode PHP. Seperti yang telah kita pelajari dalam bab sebelumnya, sebuah file PHP dapat terdiri dari kode PHP itu sendiri serta kode HTML, CSS, dan JavaScript. Oleh karena itu, web server membutuhkan suatu cara untuk menandai bagian mana yang merupakan kode PHP, dan mana HTML.

### Standar PHP tag

PHP tag yang paling umum dan paling disarankan adalah dengan karakter **<?php** sebagai tag pembuka dan karakter **?>** sebagai tag penutup. Karakter **<?php** dan **?>** secara formal disebut juga sebagai **Standar PHP tag** atau **XML style tag**.

Diantara kedua tag inilah kita menulis kode PHP. Jika perintah PHP berada di luar tag ini, kode tersebut tidak akan berjalan dan dianggap sebagai kode HTML biasa. Berikut contoh penulisannya:

```
1 <?php
2 echo "<p>Kalimat ini berasal dari Standard PHP tag </p>";
```

```
3 ?>
```

## Short PHP tag

Short PHP tag atau disebut juga dengan *SMGL style tag*, menggunakan tag `<?` sebagai pembuka, dan tag `?>` sebagai penutup. Perhatikan bahwa di dalam tag pembuka, tidak menggunakan kata “php”. Berikut contohnya:

```
1 <?
2 echo "<p>Kalimat ini berasal dari Short PHP tag </p>";
```

```
3 ?>
```

## ASP Style tag

Cara ini disebut **ASP style** karena meniru tag yang digunakan oleh bahasa ASP buatan Microsoft. Tag ini menggunakan karakter `<%` sebagai tag pembuka dan `%>` sebagai tag penutup. Berikut contohnya:

```
1 <%
2 echo "<p>Kalimat ini berasal dari ASP style tag </p>";
```

```
3 %>
```

## HTML script tag

HTML script tag menggunakan tag HTML `<script>` untuk menginput kode PHP, namun agar berbeda dengan JavaScript, tag pembuka ditulis dengan `<script language="php">`, sedangkan tag penutup menggunakan `</script>`. Berikut contohnya:

```
1 <script language="php">
2 echo "<p>Kalimat ini berasal dari HTML script tag </p>";
```

```
3 </script>
```

Selain **Standar PHP tag**, ketiga PHP Tag lain tidak bisa gunakan langsung, tapi harus diaktifkan terlebih dahulu dari file **php.ini**. Khusus untuk PHP 7, **ASP style tag** dan **HTML script tag** sudah sepenuhnya dihapus dan tidak bisa digunakan lagi. Oleh karena itu sebaiknya kita tetap menggunakan tag standar `<?php` dan `?>` untuk masuk ke PHP mode.



Untuk file PHP yang seluruhnya terdiri dari kode PHP (tanpa ada kode HTML), kita bisa mengabaikan tag penutup `?>` di akhir file. Beberapa referensi malah menyarankan hal ini, termasuk manual resmi PHP. Ini untuk menghindari error yang terjadi karena *output buffering* (saya akan membahasnya pada bab tersendiri)

## 5.3 Perintah echo dan print

Hal paling dasar yang bisa kita lakukan dari PHP adalah menulis sesuatu untuk ditampilkan. Kode untuk melakukan perintah ini adalah **echo**. Selain **echo**, di dalam PHP juga tersedia perintah **print** yang juga digunakan untuk hal yang sama, berikut contohnya:

```

1 <?php
2   echo "<p>Saya lahir di Jakarta</p>";
3   print "<p>Saya lahir di Jakarta</p>";
4 ?>

```

Kedua perintah diatas akan menampilkan teks yang sama dan tidak ada perbedaan sedikitpun. Jadi, perintah mana yang sebaiknya digunakan? apakah **echo** atau **print**?

Pertanyaan ini banyak diajukan di forum-forum PHP. Tetapi kesimpulannya kembali kepada kesukaan kita masing-masing. Beberapa sumber mengatakan perintah **print** sedikit lebih lambat daripada **echo** karena **print** mengembalikan sebuah nilai, namun anda tidak akan merasakan perbedaan ini.

Jika melihat dari berbagai kode program PHP populer seperti wordpress, perintah **echo** lah yang paling banyak dipakai. Oleh karena itu, tidak ada salahnya kita juga menggunakan **echo**.

## Gabungan PHP Tag dan echo

Di dalam PHP, terdapat sebuah PHP Tag khusus yang langsung digabung dengan **echo**, yakni dengan menggunakan tag `<?=` dan tag penutup `?>`. Sebagai contoh, kedua perintah dibawah ini akan menampilkan hasil yang sama:

```

1 <p>Saya Sedang Belajar PHP dari buku <?php echo "DuniaIlkom"?></p>
2 <p>Saya Sedang Belajar PHP dari buku <?= "DuniaIlkom"?></p>

```

Karena tag `<?=` secara tidak langsung berisi perintah **echo**, tag ini hanya cocok untuk menampilkan kode yang singkat dan terdiri dari 1 baris seperti dalam pembuatan form:

```
<input type="text" name="nama_user" value="<?= "$nama"; ?>">
```

Dalam versi PHP 5.4 kebawah, tag `<?=` tidak selalu tersedia (harus diaktifkan terlebih dahulu dari **php.ini** karena sepaket dengan *Short PHP tag* ). Untuk PHP 5.4 keatas, tag ini sudah bisa digunakan langsung.

Beberapa sumber<sup>1</sup> tidak menyarankan penggunaan tag ini, karena kita tidak tahu apakah web server sudah mendukungnya atau belum.

<sup>1</sup><http://programmers.stackexchange.com/questions/151661/is-it-bad-practice-to-use-tag-in-php>

## 5.4 Statement PHP

**Statement** adalah sebuah perintah yang menginstruksikan PHP untuk melakukan sesuatu. Di dalam PHP, suatu *statement* bisa terdiri dari perintah singkat (seperti **echo** untuk menampilkan text di layar) atau sesuatu yang lebih rumit dan terdiri dari beberapa baris (seperti logika IF atau perulangan).

Mirip dengan bahasa pemrograman lain, *statement* di dalam PHP diakhiri dengan tanda **semicolon** (titik koma) “;”. Berikut adalah contoh penulisan statement di dalam PHP:

```

1 <?php
2 echo "Hello world";
3 $a = 3; $b = 4;
4 $nama = "andi";
5 $c = $a / 25.0;
6 if ($a != $b) {
7     echo "Tampilkan Data";
8 }
9 ?>

```

Sebagian besar contoh kode program diatas belum kita bahas, tetapi anda cukup perhatikan tanda “;” di akhir penulisan setiap *statement* atau baris perintah.



Lupa menambahkan tanda “;” di akhir *statement* adalah kesalahan/error yang paling sering terjadi.

Pada beberapa kasus, tanda titik koma “;” ini boleh tidak ditulis. Sebagai contoh, tanda titik koma boleh tidak ditulis pada baris terakhir sebelum tag penutup **?>**. Berikut contohnya:

```

1 <?php
2 $a = 5 + 7;
3 echo $a;
4 echo "Hello world"
5 ?>

```

Kode program diatas tidak akan error dan berjalan sempurna. Namun akan lebih baik tetap menulis tanda titik koma. Mungkin saja nanti kita ingin menambahkan baris baru.

Selain itu, tanda titik koma juga tidak boleh ditulis setelah tanda kurung kurawal ‘{’ dan ‘}’. Kedua tanda kurung ini akan kita bahas dalam bab tentang struktur kode program PHP.

## 5.5 Case Sensitivity

**Case Sensitivity** adalah istilah yang membahas apakah sebuah bahasa pemrograman membedakan penulisan huruf kecil dan huruf besar.

PHP tidak membedakan huruf besar dan kecil untuk penamaan fungsi (*function*), nama class, maupun keyword bawaan PHP seperti *echo*, *while*, dan *class* (bersifat **case insensitive**).

Ketiga baris berikut akan dianggap sama oleh PHP:

```
1 <?php
2     echo "Hello World";
3     ECHO "Hello World";
4     EcHo "Hello World";
5 ?>
```

Akan tetapi, PHP membedakan penulisan huruf besar dan huruf kecil (**case sensitive**) untuk penamaan variabel. Variabel *\$nama*, *\$Nama* dan *\_ \$NAMA\_* akan dianggap sebagai 3 variabel yang berbeda. Sering kali terjadi error karena kita salah menulis nama variabel. Yang seharusnya menggunakan huruf kecil, ditulis dengan huruf besar, seperti contoh berikut:

```
1 <?php
2     $andi="Andi";
3     echo $Andi; // Notice: Undefined variable: Andi
4 ?>
```

Untuk mengatasi hal ini, saya menyarankan menggunakan huruf kecil untuk seluruh kode PHP, termasuk penulisan variabel, fungsi maupun class. Dengan demikian, kita akan terhindar dari kesalahan seperti diatas.

## 5.6 Whitespace

**Whitespace** adalah istilah pemrograman yang merujuk kepada karakter khusus seperti spasi, tab, enter, dan karakter lain yang ‘tidak tampak’ di dalam kode program.

Secara umum, whitespace akan diabaikan pada saat eksekusi kode program PHP. Kita boleh memecah sebuah *statement* menjadi beberapa baris, atau menyatukan beberapa *statement* dalam sebuah baris yang panjang. Seperti contoh berikut:

```
1 <?php
2     echo "Ini kalimat pertama"; echo "Ini kalimat kedua"; $nama="andi";
3 ?>
```

Baris perintah diatas sama artinya dengan:

```
1 <?php
2 echo "Ini kalimat pertama";
3 echo "Ini kalimat kedua";
4 $nama="andi";
5 ?>
```

Walaupun cara penulisan ini akan menghasilkan ukuran file yang sedikit lebih besar (beberapa byte) tetapi sangat disarankan agar kode program lebih mudah dibaca.

**Indenting (indent)** adalah istilah yang berarti menjorokkan penulisan kode program menggunakan tab atau karakter spasi. Hal ini digunakan agar kode program lebih rapi dan lebih mudah dibaca. Jika anda menggunakan text editor khusus pemrograman seperti **Notepad++**, proses indenting ini bisa dilakukan secara otomatis.

## 5.7 Baris Komentar

Dalam merancang kode program, kadang kita perlu menambahkan keterangan, catatan atau komentar yang berisi penjelasan untuk apa kode tersebut dibuat. Baris komentar ini tidak akan diproses dan diabaikan oleh web server.

Walaupun kode program kita tidak akan dibaca oleh orang lain, tetap disarankan untuk memberikan penjelasan tentang apa tujuan kode tersebut, terutama untuk kode yang cukup rumit. Jika anda membaca kode program yang sama 3 bulan mendatang, belum tentu anda masih ingat apa yang dikerjakan oleh kode tersebut.

PHP mendukung 3 jenis cara penulisan komentar: **Unix Shell style comment**, **C style comment**, dan **C++ style comment**.

### Unix Shell style comment

Disebut sebagai **Unix Shell**, karena cara penulisan komentar ini berasal dari sistem Unix. Metode ini menggunakan karakter **tanda pagar** atau **hash mark** (#). PHP akan mengabaikan seluruh text yang terdapat setelah tanda pagar sampai akhir baris atau tag penutup PHP (mana yang terlebih dahulu didapatkan). Berikut contoh penggunaannya:

```
1 <?php
2 $nilai = $p * exp($r * $t); # menghitung bunga majemuk
3 ?>
```

Komentar ini berlaku hanya untuk 1 baris saja. Jika kita ingin memberikan komentar lebih dari 1 baris, harus membuatnya lagi:

```
1 <?php
2  # kode program berikut digunakan untuk
3  # menghitung bunga majemuk
4  $nilai = $p * exp($r * $t);
5 ?>
```

## C++ style comment

Metode komentar ini meminjam cara membuat komentar dari bahasa pemrograman C++. Hampir sama dengan komentar *Unix Shell style*, metode komentar C++ berlaku hanya untuk satu baris saja atau sampai tag penutup PHP ditemukan. Kali ini karakter yang digunakan adalah dua kali garis miring (**two slashes**), yakni “//”. Berikut contoh penggunaannya:

```
1 <?php
2  // kode program berikut digunakan untuk
3  // menghitung bunga majemuk
4  $nilai = $p * exp($r * $t);
5 ?>
```

## C style comment

Jika metode komentar *Unix Shell style* dan *C++ style* digunakan untuk membuat komentar pendek, untuk membuat komentar panjang PHP meminjamnya dari bahasa C. Metode komentar ini disebut juga **tipe komentar blok** karena bisa membuat komentar dalam 1 blok (tidak per baris seperti cara penulisan komentar sebelumnya).

Awal komentar menggunakan tanda “/\*”, dan akhir komentar menggunakan tanda “\*/”. Seluruh baris diantara kedua karakter ini dianggap sebagai komentar. Berikut contoh penggunaannya:

```
1 <?php
2  /* kode program berikut digunakan untuk
3  menghitung bunga majemuk */
4  $nilai = $p * exp($r * $t);
5 ?>
```

Selain untuk memberikan keterangan, baris komentar juga sering digunakan untuk menandai awal dari sebuah blok program, atau untuk menghentikan baris program supaya tidak dijalankan oleh PHP. Metoda ini sangat berguna jika Anda mencoba kode program baru, tetapi tidak ingin menghapus kode program yang sudah ada.

```

1 <?php
2 /////////////////
3 // Validasi Form Register
4 /////////////////
5
6 // $nama = "duniailkom";
7 $nama = "andi";
8 echo $nama;
9 ?>

```



Untuk menghemat tempat, dalam beberapa contoh kode program saya akan menampilkan hasil di dalam tanda komentar.

## 5.8 Fungsi phpinfo()

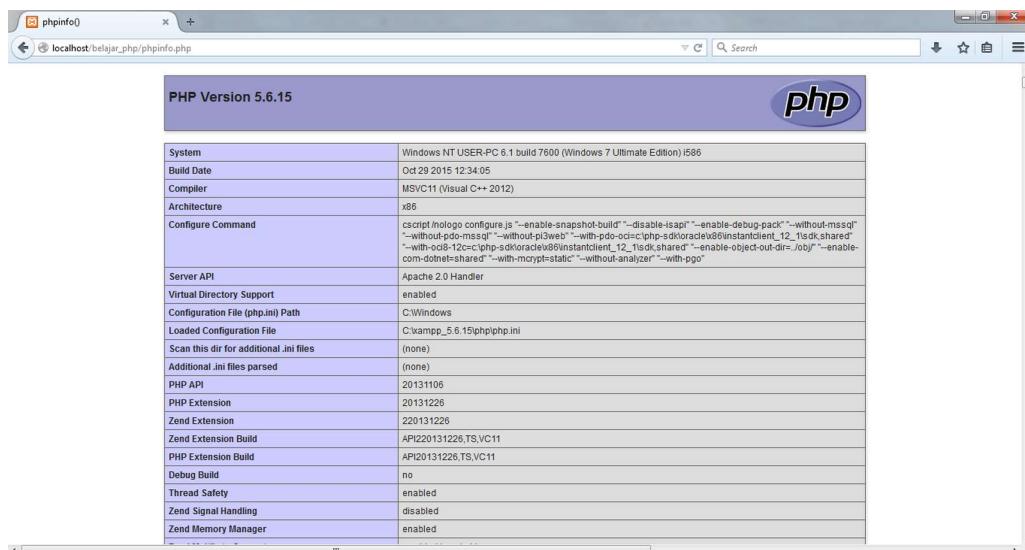
PHP memiliki berbagai macam fitur dan konfigurasi. Untuk melihat informasi seperti versi PHP yang digunakan, modul-modul PHP yang terinstall, dll. Kita bisa menggunakan fungsi `phpinfo()`. Walaupun saya belum membahas tentang fungsi (*function*), tapi penggunaannya tidak terlalu sulit.

Buatlah sebuah file php dengan nama file sembarang, kemudian ketikkan kode berikut:

```

1 <?php
2   phpinfo();
3 ?>

```



Gambar: Tampilan phpinfo()

Jalankan halaman tersebut, dan anda akan dapat melihat seluruh informasi tentang php yang saat ini sedang berjalan. Pada bagian paling atas, dapat terlihat versi PHP yang digunakan.



Tampilan `phpinfo()` seperti ini juga bisa diakses dari halaman awal XAMPP. Silahkan buka localhost, kemudian pilih menu `phpinfo()` pada menu sebelah kanan atas.

## 5.9 Mengatasi Error

Sepanjang proses pembuatan kode program, sangat susah untuk menghindari error. Mulai dari yang paling dasar seperti lupa menutup statement dengan tanda ‘;’ hingga kesalahan logika yang susah di telusuri.

Jika PHP adalah bahasa pemrograman pertama anda (HTML dan CSS sebenarnya bukanlah ‘bahasa pemrograman’), menemukan kode program yang error tampak seperti sebuah mimpi buruk. Tapi, anda tidak perlu takut. Error adalah hal yang lumrah dan akan jadi ‘makanan sehari-hari’ dalam pembuatan kode program. Programmer paling ahli sekalipun juga tetap mendapati error, terutama untuk kode program yang cukup rumit.

Semakin sering anda berlatih menjalankan kode program (dan melakukan kesalahan), akan semakin mudah untuk memahami kode error yang terjadi.

Secara garis besar, terdapat beberapa level error di dalam PHP, yakni **Error**, **Warning**, dan **Notice**:

### Error

**Error** adalah pesan yang berarti ada sesuatu yang salah di dalam kode program. Error ini harus diperbaiki karena PHP berhenti memproses kode program.

Lupa menutup sebuah statement dengan tanda ‘;’ akan menghasilkan error:

```
1 <?php
2   $a = 5 + 7 // Parse error: syntax error
3   echo $a;
4 ?>
```

### Warning

**Warning** adalah pesan yang menyatakan ada sesuatu yang salah, tapi PHP akan mencoba melanjutkan kode program. Contohnya, ketika meng-include file PHP yang tidak bisa diakses:

```
1 <?php
2  include("tidak_ada.php");
3  // Warning: include(tidak_ada.php):
4  // failed to open stream: No such file or directory
5 ?>
```

## Notice

**Notice** adalah pesan yang berisi pemberitahuan. Kode program kita pada dasarnya tidak ada yang salah, tetapi mungkin menggunakan fungsi atau perintah yang tidak disarankan.

Sebagai contoh, mengakses sebuah variabel yang belum diisi nilai akan menghasilkan notice:

```
1 <?php
2  $andi="Andi";
3  echo $Andi; // Notice: Undefined variable: Andi
4 ?>
```

Dari ketiga jenis error ini, **Error** adalah yang akan paling kita temui. Berikut beberapa tips yang bisa memandu anda untuk mengatasinya:

### Jalankan kode program sesering mungkin

Jika akan membuat 100 baris kode program, dianjurkan untuk menjalankannya setiap beberapa baris. Misalkan setelah membuat 10 baris pertama, jalankan PHP. Jika tidak ditemukan error, baru tambah 10 baris lagi. Kemudian jalankan kembali. Lakukan hal ini hingga seluruh program selesai.

Dengan menguji kode program secara bertahap, kita bisa mendekripsi error dengan lebih mudah.

### Jangan timpa kode program yang sudah berjalan

Kadang kita ingin mencoba sesuatu yang baru terhadap kode program yang sebenarnya sudah jalan. Apakah itu untuk mencari cara yang lebih efisien, atau agar kode program jadi lebih rapi.

Untuk situasi seperti ini, JANGAN timpa kode program lama anda, tetapi copy kode tersebut, dan ubah kode lama menjadi komentar. Dengan cara ini, lebih gampang untuk mengoreksi kode program jika kode yang baru ternyata tidak berjalan.

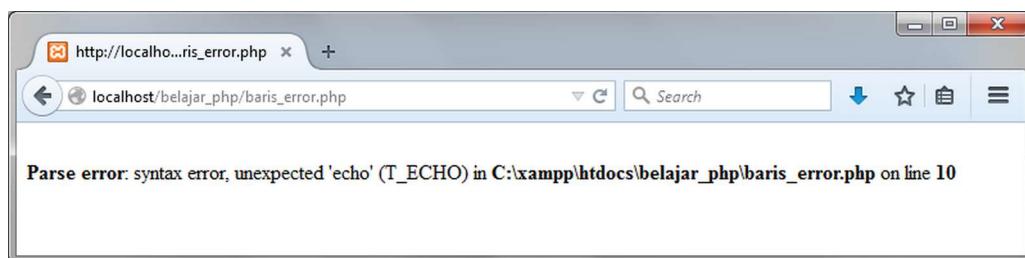
Berikut contohnya:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Daftar Mahasiswa</title>
6  </head>
7  <body>
8  <h2>Daftar Absensi Mahasiswa</h2>
9  <ol>
10     <?php
11         /* Kode program lama
12
13         for ($i= 1; $i <= 10; $i++) {
14             echo "<li>Nama Mahasiswa ke-$i</li>";
15         }
16     */
17
18         for ($i= 1; $i <= 1000; $i++) {
19             echo "<li>Nama Mahasiswa ke-$i</li>";
20         }
21     ?>
22     </ol>
23 </body>
24 </html>
```

### Perhatikan nomor baris dari error yang terjadi

Ketika terjadi error, PHP akan memperlihatkan urutan baris kode program yang menyebabkan error tersebut. Urutan baris ini menjadi petunjuk yang paling berharga. Jika pada baris tersebut anda yakin sudah tidak error, lihat baris sebelumnya.

Sebagai contoh, berikut pesan error yang saya dapat:



Gambar: Syntax error pada baris ke-10

Berikut kode programnya:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6  </head>
7  <body>
8      <?php
9      $a = 5 + 7
10     echo $a;
11     ?>
12  </body>
13  </html>
```

Seperti yang terlihat, kesalahan sebenarnya bukan pada baris ke 10, tapi di baris ke-9, dimana saya lupa menutup statement dengan tanda ‘;’.

### **Jika terjadi beberapa error sekaligus, fokuslah para error pertama**

Ketika anda menemukan banyak error yang terjadi (mungkin bisa lebih dari 10 error), fokuslah pada error yang pertama. Sering kali error inilah yang menyebabkan error lainnya.

### **Copy pesan error dan paste ke Google**

Mungkin inilah tips yang paling ampuh untuk mengatasi error. Dengan jutaan programmer di seluruh dunia, besar kemungkinan ada orang lain yang telah mengalaminya :)

---

Dalam bab ini kita telah membahas beberapa aturan penulisan dasar kode program PHP, termasuk tips menangani error. Berikutnya, kita akan membahas tentang Variabel dan Konstanta PHP.

# 6. Variabel dan Konstanta

Dalam bab ini saya akan membahas salah satu inti dari PHP (dan bahasa pemrograman pada umumnya). Kita akan mempelajari tentang pengertian variabel, cara penulisan variabel serta perbedaannya dengan konstanta.

## 6.1 Pengertian Variabel

Mengutip dari [wikipedia](#)<sup>1</sup>, variabel adalah *suatu lokasi penyimpanan di dalam memori komputer yang berisi data atau informasi yang nilainya telah diketahui maupun belum diketahui*. Variabel memiliki **nama** atau **identifier** yang digunakan untuk mengakses nilai ini.

*In computer programming, a **variable** or scalar is a storage location paired with an associated symbolic name (an identifier), which contains some known or unknown quantity or information referred to as a value -wikipedia.*

Dalam defenisi sederhana, variabel adalah *kode program yang digunakan untuk menampung nilai dari sebuah data*. Nilai ini bisa berupa angka, teks, objek, dan lain-lain. Sebuah variabel memiliki **nama** yang digunakan untuk mengakses nilai dari variabel tersebut.

Sepanjang program, nilai dari variabel dapat diubah isinya dengan nilai lain. Jika anda memiliki pengetahuan dasar tentang bahasa pemograman, tentunya tidak asing dengan istilah variabel.

## 6.2 Aturan Penulisan Variabel PHP

Untuk membuat variabel, PHP memiliki aturan penulisan sebagai berikut:

- Sebuah variabel harus diawali dengan tanda **dollar** (\$)
- Setelah tanda dollar, karakter pertama setelahnya harus berupa huruf atau underscore (\_).
- Karakter kedua dan seterusnya bisa berupa huruf, angka, atau underscore (\_).
- Nama variabel bersifat **case sensitif** (huruf besar dan huruf kecil dianggap berbeda).
- Untuk memberikan nilai kepada sebuah variabel, PHP menggunakan karakter sama dengan (=).
- Variabel dalam PHP tidak memerlukan deklarasi terlebih dahulu.

Sebagai contoh, berikut adalah cara penulisan variabel yang benar di dalam PHP:

---

<sup>1</sup>[http://en.wikipedia.org/wiki/Variable\\_%28computer\\_science%29](http://en.wikipedia.org/wiki/Variable_%28computer_science%29)

```
1 <?php
2   $i;
3   $nama;
4   $Umur;
5   $_lokasi_memori;
6   $ANGKA_MAKSIMUM;
7 ?>
```

Berikut adalah contoh penulisan variabel yang salah:

```
1 <?php
2   $4ever;           //variabel tidak boleh diawali dengan angka
3   $_ salah satu; //varibel tidak boleh mengandung spasi
4   $nama*^;         //variabel tidak boleh mengandung karakter khusus: * dan ^
5 ?>
```

Tanda sama dengan (=) digunakan untuk memberikan nilai ke dalam variabel:

```
1 <?php
2   $nama = "andi";
3   $umur = 17;
4   $pesan = "Saya sedang belajar PHP";
5 ?>
```

Perintah pemberian nilai kepada sebuah variabel disebut dengan **assignment**. Jika variabel tersebut belum pernah digunakan, dan langsung diberikan nilai awal, maka disebut juga dengan proses **inisialisasi**, seperti pada kode diatas.

Dapat anda perhatikan bahwa di dalam PHP sebuah variabel tidak perlu ditulis akan berisi data angka (**integer**) atau teks (**string**). Setiap variabel di dalam PHP bisa berisi data apa saja dan dapat diubah isinya kapan saja, seperti contoh berikut:

```
1 <?php
2   $foo = 17;      // nilai variabel $foo berisi angka (integer)
3   $foo = "aku";  // nilai variabel $foo diubah menjadi kata (string)
4   $foo = 17.42;  // nilai variabel $foo diubah menjadi desimal (float)
5 ?>
```

Variabel akan sangat sering kita gunakan sepanjang pembuatan program dengan PHP.

## Tips Penulisan Variabel

Beberapa programmer PHP cenderung mengadopsi cara penulisan variabel yang dinamakan **snake\_case**, yakni menggunakan huruf kecil untuk setiap variabel dan menggunakan underscore sebagai pemisah kata, contohnya: `$jumlah_data`, atau `$koneksi_database`.

Sebagian lagi menggunakan penulisan **camelCase**, yakni membuat nama variabel dengan menggunakan huruf kecil pada kata pertama, kemudian menggunakan huruf besar pada karakter pertama kata kedua, ketiga dan seterusnya. Berikut contoh penulisannya: `$jumlahData`, `$koneksiDatabase` atau `$namaPanjang`.

Anda bebas ingin menggunakan gaya penulisan yang mana, apakah **snake\_case** atau **camelCase**.



Walaupun demikian, hampir semua variabel dan fungsi bawaan PHP menggunakan gaya penulisan **snake\_case**. Dalam buku ini saya juga akan menggunakan gaya penulisan **snake\_case**.

Selain itu, sangat disarankan untuk membuat nama variabel sesuai dengan fungsinya. Membuat variabel dengan 1 atau 2 karakter memang lebih cepat, tetapi akan mempersulit pembacaan program. Daripada membuat variabel `$a`, `$b` atau `$aa` lebih baik menggantinya menjadi `$nama`, `$nama_lengkap` atau `$alamat_rumah`.

## 6.3 Pengertian Konstanta

Masih mengutip dari [wikipedia<sup>2</sup>](#), konstanta (*constant*) adalah *lokasi penyimpanan di dalam memori komputer yang berisi data atau informasi yang nilainya bersifat tetap dan tidak bisa diubah*. Sama seperti variabel, konstanta juga menggunakan **nama** atau **identifier** yang digunakan untuk mengakses nilai dari konstanta tersebut.

*In computer programming, a constant is an identifier with an associated value which cannot be altered by the program during normal execution (the value is constant) - wikipedia.*

Berbeda dengan variabel yang isi-nilainya dapat diubah bahkan dihapus selama program berjalan, sebuah konstanta jika telah diberikan nilai, tidak dapat diubah lagi. Hal ini sesuai dengan namanya, yakni konstan (tetap).

## 6.4 Aturan Penulisan Konstanta PHP

Untuk membuat konstanta, PHP memiliki aturan penulisan sebagai berikut:

- Konstanta dibuat menggunakan keyword **const** atau dengan fungsi **define()**.

<sup>2</sup>[http://en.wikipedia.org/wiki/Constant\\_%28programming%29](http://en.wikipedia.org/wiki/Constant_%28programming%29)

- Karakter pertama untuk sebuah konstanta harus diawali dengan huruf atau underscore (\_).
- Karakter kedua dan seterusnya bisa berupa huruf, angka, atau underscore (\_).
- Nama konstanta bersifat **case sensitif** (huruf besar dan huruf kecil dianggap berbeda).
- Setelah dibuat, nilai di dalam konstanta tidak bisa diubah.

Berikut adalah contoh cara pembuatan konstanta di dalam PHP:

```
1 <?php
2 // menggunakan keyword const
3 const kota1 = "Jakarta";
4
5 // menggunakan fungsi define
6 define("kota2", "Bandung");
7 ?>
```

Sesuai dengan namanya, jika sebuah konstanta telah dibuat, maka kita tidak akan bisa mengubah nilai konstanta tersebut:

```
1 <?php
2 define("GAJI", 5000000);
3 define("GAJI", 50000);
4 // Notice: Constant GAJI already defined
5 ?>
```

Sepanjang pembuatan program, umumnya kita tidak akan terlalu sering menggunakan konstanta.

## Tips Penulisan Konstanta

Jika variabel disarankan menggunakan **snake\_case** dengan huruf kecil, untuk konstanta PHP banyak yang menggunakan **snake\_case** dengan HURUF BESAR. Konstanta bawaan PHP juga ditulis dengan huruf besar. Ini dilakukan agar mudah dibedakan dengan kode program lain.

## 6.5 Predefined Variable dan Predefined Constant

PHP memiliki beberapa variabel dan konstanta yang digunakan secara internal oleh PHP itu sendiri. Oleh karena itu kita sebaiknya tidak menggunakan nama variabel dan nama konstanta yang sama.

Variabel yang digunakan secara internal oleh PHP dikenal dengan sebutan **Predefined Variable** atau **Reserved Variable**. Daftar lengkap predefined variable PHP dapat dilihat di: [PHP reserved variables<sup>3</sup>](http://php.net/manual/en/reserved.variables.php). Beberapa diantaranya adalah:

---

<sup>3</sup><http://php.net/manual/en/reserved.variables.php>

```
$GLOBALS, $_SERVER, $_GET, $_POST, $_FILES, $_COOKIE, $_SESSION,  
$_REQUEST, $_ENV, $php_errormsg, $HTTP_RAW_POST_DATA,  
$http_response_header, $argc, $argv, $this
```

Sedangkan **Predefined Constant** atau **Reserved Constant** adalah nama-nama konstanta yang digunakan secara internal oleh PHP. Daftar lengkap predefined constant dapat dilihat di: [PHP reserved constants](#)<sup>4</sup>. Beberapa diantaranya adalah:

```
PHP_VERSION, PHP_MAJOR_VERSION, PHP_MINOR_VERSION, PHP_RELEASE_VERSION,  
PHP_VERSION_ID, PHP_EXTRA_VERSION, __LINE__, __FILE__, __DIR__,  
__FUNCTION__, __CLASS__, __TRAIT__, __METHOD__, __NAMESPACE__
```

## 6.6 Cara Menampilkan Variabel dan Konstanta

Setelah membuat variabel atau konstanta, tentunya kita ingin mengakses nilai yang tersimpan di dalamnya. PHP menyediakan berbagai cara untuk menampilkan nilai ini.

Cara yang paling dasar adalah menggunakan perintah **echo**, kemudian diikuti dengan nama variabel atau konstanta tersebut. Berikut contohnya:

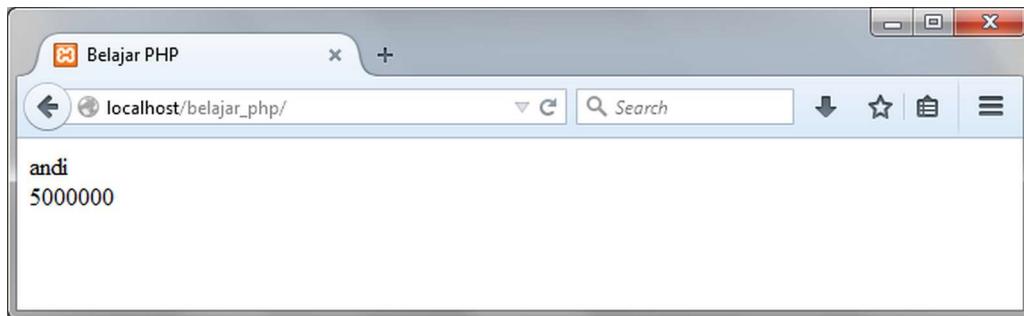
### 01.menampilkan\_variabel\_dan\_konstanta.php

---

```
1  <!DOCTYPE html>  
2  <html>  
3  <head>  
4  <meta charset="UTF-8">  
5  <title>Belajar PHP</title>  
6  </head>  
7  <body>  
8  <?php  
9  $nama="andi";  
10 define("GAJI", 5000000);  
11  
12 echo $nama;  
13 echo "<br>";  
14 echo GAJI;  
15 ?>  
16 </body>  
17 </html>
```

---

<sup>4</sup><http://php.net/manual/en/reserved.constants.php>



Gambar: Menampilkan nilai variabel dan konstanta dengan perintah echo

Selain menampilkan variabel secara langsung, kita juga bisa menampilkan variabel di dalam sebuah **string** (saya akan membahas tentang tipe data string dalam bab selanjutnya). Berikut contoh pengaksesan variabel yang berada di dalam teks (*string*):

```
1 <?php
2   $nama="andi";
3   echo "Selamat pagi $nama"; // hasil: Selamat pagi andi
4 ?>
```

Agar hasil tersebut dapat berjalan, kalimat tersebut harus berada di antara tanda kutip dua ("").

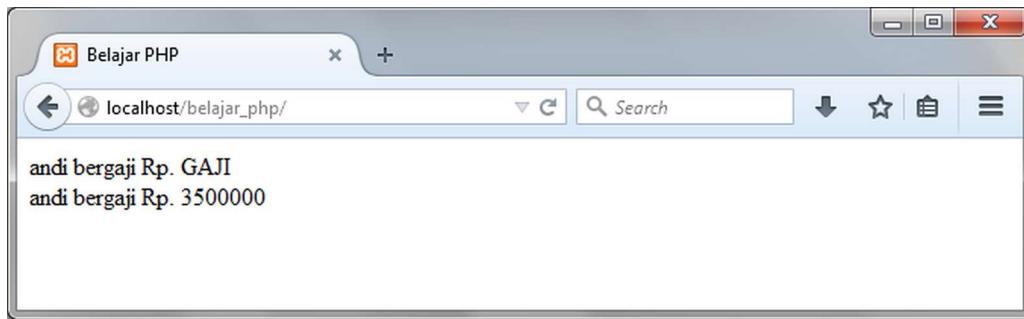
Untuk konstanta, teks tersebut harus diputus dan disambung menggunakan operator penyambungan string, yakni tanda titik (.). Berikut contohnya:

#### 02.konstanta\_string.php

---

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar PHP</title>
6 </head>
7 <body>
8 <?php
9   $nama="andi";
10  define("GAJI", 3500000);
11
12  echo "$nama bergaji Rp. GAJI"; // hasil: andi bergaji Rp. GAJI
13  echo "<br>";
14  echo "$nama bergaji Rp. ".GAJI; // hasil: andi bergaji Rp. 3500000
15 ?>
16 </body>
17 </html>
```

---



Gambar: Menampilkan konstanta di dalam String

---

Variabel dan konstanta yang kita pelajari disini hampir selalu ada pada setiap program PHP. Penggunaan variabel inilah yang membuat PHP bisa menghasilkan website dinamis.

Variabel dan konstanta baru memiliki makna ketika diisi dengan data. Dalam bab selanjutnya, saya akan membahas lebih lanjut tentang tipe data di dalam PHP.

# 7. Tipe Data PHP

Variabel dan **konstanta** yang telah kita bahas dalam bab sebelumnya merupakan tempat atau ‘container’ dari data. Agar dapat diproses, PHP membagi kelompok data sesuai dengan jenisnya masing-masing. Kelompok data inilah yang dikenal dengan **tipe data**. Dalam bab ini akan dibahas secara rinci tipe-tipe data PHP.



Bagi anda yang belum pernah belajar PHP atau bahasa pemrograman komputer, mungkin akan bertanya ‘apa fungsi dari yang saya pelajari disini?’. Dalam setengah buku PHP Uncover, kita masih mempelajari dasar-dasar PHP yang sepintas belum jelas apa tujuannya.

Dasar-dasar pemrograman ini baru berperan ketika kita masuk ke bagian ‘aplikasi’ seperti pemrosesan form atau mengambil data dari database. Jadi, buat sementara pelajari dulu apa yang ada :)

## 7.1 Jenis-Jenis Tipe Data di dalam PHP

Di dalam PHP, terdapat 8 jenis tipe data yang bisa dikelompokkan menjadi 3 bagian:

- Tipe data dasar / tipe data *primitive* / tipe data *scalar*:
  1. **Integer** (angka bulat)
  2. **Float** (angka pecahan)
  3. **Boolean** (logika true atau false)
  4. **String** (teks)
- Tipe data *composite* / tipe data *compound*:
  1. **Array**
  2. **Object**
- Tipe data khusus:
  1. **Resource**
  2. **Null**

### Tipe data dasar / tipe data *primitive* / tipe data *scalar*

Tipe data dasar adalah tipe data yang umum tersedia di setiap bahasa pemrograman.

Tipe data **integer** terdiri dari angka bulat seperti 3, 4, 900 atau 1000. Tipe data **float** terdiri dari kumpulan angka pecahan seperti 4.3, 9.99 atau 164.55. Tipe data **boolean** hanya memiliki 2 nilai, yakni *true* atau *false*. Tipe data **string** berisi data teks seperti “A”, “xyz”, “Andi” atau “Sedang belajar PHP”.

Tipe data dasar ini disebut juga dengan tipe data **scalar** karena hanya bisa berisi 1 nilai (seperti pengertian **scalar** dalam *aljabar linear*). Tipe data **scalar** berbeda dengan tipe data *composite*/*compound* yang bisa terdiri dari beberapa nilai.

## Tipe data **composite** / tipe data **compound**

Tipe data *composite* / tipe data *compound* terdiri dari 2 jenis: **array** dan **object**.

**Array** merupakan tipe data bentukan yang terdiri dari berbagai tipe data dasar. Tipe data **object** juga dibangun dari beberapa tipe data dasar, sekaligus memiliki fungsi (*function*) sendiri. Tipe data **object** akan dipelajari secara khusus dalam pemrograman berbasis object (*object oriented programming* – **OOP**).

## Tipe data khusus

Di dalam PHP, terdapat 2 tipe data khusus: **resources** dan **null**.

Tipe data **resources** digunakan untuk menyimpan link atau referensi untuk ke aplikasi lain di luar PHP, contohnya: hasil koneksi dari database.

Tipe data **null** adalah tipe data khusus yang hanya bisa diisi 1 nilai, yakni: ‘**null**’. Dalam PHP, null berarti ‘tidak ada data’.

Kita akan membahas lebih dalam tentang masing-masing tipe data ini.

## 7.2 PHP sebagai Typeless Programming Language

Seluruh tipe data PHP yang dijelaskan sebelumnya, akan disimpan ke dalam variabel atau konstanta.

PHP termasuk kelompok bahasa pemrograman yang dikenal dengan **typeless programming language**, yakni bahasa pemrograman yang variabelnya tidak harus bertipe data tertentu. Sebuah variabel PHP bisa diisi dengan tipe data apapun dan diubah nilainya kapanpun.

Ini sedikit berbeda dibandingkan bahasa pemrograman berjenis **typed programming languages** seperti Pascal, C, C++, atau Java. Dalam bahasa ini, seluruh variabel harus dideklarasikan untuk 1 jenis tipe data, dan tidak bisa diisi dengan tipe data lain.

Sebagai contoh, berikut adalah cara pendefenisian variabel di dalam bahasa pascal:

```
var
  var1:integer;
  var2:real;
  var3:string
begin
  var1:= 12;
  var2:= 99.99;
  var3:= "dunia1kom";
  var3:= 14 // error !!!
end.
```

Seperti yang terlihat, di dalam Pascal seluruh variabel (**var1**, **var2** dan **var3**) harus dideklarasikan akan berjenis tipe data apa. Kita juga tidak bisa mengisi variabel ini dengan jenis tipe data yang berbeda.

Di dalam PHP, sebuah variabel bisa diisi tipe data apapun dan kapanpun:

```

1 <?php
2   $var1 = 12;
3   $var2 = 99.99;
4   $var3 = "duniailkom";
5   $var3 = 14;
6   $var1 = "Sedang belajar PHP";
7 ?>

```

Konsep **typeless programming language** membuat PHP ‘relatif’ lebih mudah ditulis dan dipelajari. Akan tetapi ini juga mendatangkan masalah, karena sebuah tipe data bisa ‘berubah’ menjadi tipe data lain tanpa kita instruksikan (saya akan membahas hal ini di akhir bab).

## 7.3 Cara Memeriksa Tipe Data dengan var\_dump()

Karena PHP tidak memerlukan deklarasi tipe data, setiap variabel bisa bertipe data apa saja, tergantung nilai yang di input ke dalam variabel tersebut. Perhatikan contoh dibawah ini:

```

1 <?php
2   $var = 12;           // $var bertipe integer
3   $var = 99.99;        // $var berubah menjadi float
4   $var = "duniailkom"; // $var menjadi string
5 ?>

```

Untuk dapat mengetahui jenis tipe data suatu variabel, kita bisa menggunakan fungsi **var\_dump()**. Berikut contoh penggunaannya:

```

1 <?php
2   $var=12;
3   var_dump($var);    // int(12)
4   echo "<br/>";
5
6   $var=99.99;
7   var_dump($var);    // float(99.99)
8   echo "<br/>";
9
10  $var="duniailkom";
11  var_dump($var);    // string(10) "duniailkom"
12 ?>

```

Dapat dilihat bahwa tipe data sebuah variabel berubah-ubah tergantung isinya. Fungsi **var\_dump()** akan memperlihatkan tipe data sebuah variabel. Fungsi ini sangat berguna selama proses pembuatan program.



Dalam bab ini saya akan sering menggunakan **var\_dump()** daripada **echo**. Ini agar kita bisa melihat tipe data yang digunakan untuk setiap variabel.

## 7.4 Tipe Data Integer

Tipe data pertama yang akan kita bahas adalah **integer**. Integer merupakan tipe data yang terdiri dari angka bulat, seperti 1, 2, 6, atau 9999. Tipe data ini cocok digunakan untuk menampung nilai yang *pasti* bulat, seperti jumlah orang, jumlah peserta, jumlah hari, dll.

Nilai integer dapat bernilai positif (+) maupun negatif (-). Jika tidak diberi tanda, maka diasumsikan nilai tersebut adalah positif.

Berikut contoh penggunaan tipe data integer:

```

1 <?php
2   $umur=21;
3   $harga=15000;
4   $keuntungan=-500000;
5
6   var_dump($umur);      // int(21)
7   echo "<br />";
8   var_dump($harga);      // int(15000)
9   echo "<br />";
10  var_dump($keuntungan); // int(-500000)
11 ?>

```

Jangkauan angka integer bergantung kepada kemampuan komputasi komputer. Biasanya dimulai dari -2,147,483,648 sampai dengan +2,147,483,647, atau 32bit.

Untuk mengetahui nilai maksimal tipe data integer pada komputer, PHP menyediakan konstanta **PHP\_INT\_MAX**. Berikut adalah hasil nilai **PHP\_INT\_MAX** yang saya jalankan:

```

1 <?php
2   print PHP_INT_MAX; // 2147483647
3 ?>

```

Bagaimana jika angka yang digunakan diluar dari rentang ini? Mari kita coba:

```

1 <?php
2   $var=9999999999;
3   var_dump($var); // float(9999999999)
4 ?>

```

Dari hasil yang didapat, PHP secara otomatis akan menkonversi nilainya menjadi tipe data **float**.

Selain angka bulat ‘normal’, dalam komputasi komputer juga terdapat beberapa jenis sistem bilangan yang umum digunakan selain bilangan desimal. Bilangan tersebut adalah **heksadesimal**, **oktal**, dan **biner**.

## Bilangan Integer Heksadesimal

Berbeda dengan bilangan desimal yang menggunakan 10 digit (0-9), bilangan **heksadesimal** menggunakan 16 digit, yakni angka 0-9 dan huruf A-F. Penggunaan bilangan heksadesimal yang paling sering adalah untuk kode warna CSS. Jika anda telah mempelajari HTML atau CSS, mungkin tidak asing dengan kode warna #FF00AA, atau #08BAFC.

Tipe data integer di dalam PHP juga mendukung penulisan heksadesimal, yakni dengan menambahkan tanda “0x” (angka nol dan huruf ‘x’) sebelum penulisan angka.

Berikut contoh penggunaan angka heksadesimal di dalam PHP:

```

1 <?php
2   $angka_desimal= 43981;
3   $angka_heksadesimal=0xABCD;      // ABCD heksadesimal = 43981 desimal
4
5   var_dump($angka_desimal);        // int(43981)
6   echo "<br />";
7   var_dump($angka_heksadesimal); // int(43981)
8 ?>

```

Pada contoh diatas, variabel **\$angka\_desimal** dan **\$angka\_heksadesimal** sama-sama ditampilkan dengan nilai 43981 (PHP secara tidak langsung mengkonversi nilai **\$angka\_heksadesimal** menjadi nilai desimal).

## Bilangan Integer Oktal

Bilangan **oktal** adalah sistem bilangan yang terdiri dari 8 karakter (angka 0-7). Untuk membuat angka oktal di dalam PHP, kita menambahkan karakter “0” (angka nol), kemudian diikuti dengan penulisan angka.

Berikut contoh penggunaan angka oktal di dalam PHP:

```

1 <?php
2   $angka_desimal= 511;
3   $angka_oktal=0777;        // 777 heksadesimal = 511 desimal
4
5   var_dump($angka_desimal); // int(511)
6   echo "<br />";
7   var_dump($angka_oktal);  // int(511)
8 ?>

```

Bilangan oktal tidak terlalu sering digunakan di dalam web programming. Anda mungkin akan menggunakannya jika membuat program yang berhubungan dengan karakter dan byte (1 byte = 8bit).

Penulisan 0 didepan sebuah angka sering kita tulis dalam fungsi matematika atau catatan sehari-hari. Karena di dalam PHP angka 0 ini berfungsi sebagai penanda bilangan oktal, sedapat mungkin hindari kebiasaan ini.

Jika yang dimaksud agar angka menjadi cantik, misalkan untuk pengurutan nomor : 01, 02, 03.. dst. PHP juga menyediakan fungsi khusus untuk keperluan ini (akan kita pelajari nantinya).

## Bilangan Integer Biner

Bilangan **biner** (atau *binary*) adalah sistem bilangan yang hanya terdiri dari 2 karakter saja, yakni 1 dan 0. Bilangan biner kadang disebut juga dengan bilangan logika, yakni logika benar (ditandai dengan angka 1), dan logika salah (ditandai dengan angka 0). Untuk membuat angka biner di dalam PHP, kita menambahkan karakter “**0b**” (angka nol dan huruf b).

Berikut contoh penggunaan angka biner di dalam PHP:

```
1 <?php
2     $angka_desimal= 170;
3     $angka_biner=0b10101010; // 10101010 biner = 170 desimal
4
5     var_dump($angka_desimal); // int(170)
6     echo "<br />";
7     var_dump($angka_biner); // int(170)
8 ?>
```

Ketika ditampilkan, variabel yang berisi angka biner akan langsung terkonversi menjadi angka desimal, seperti variabel **\$angka\_biner** pada contoh diatas.

## 7.5 Tipe Data Float

Tipe data **float** adalah tipe data yang terdiri dari angka pecahan, seperti 1.33, 7.99, atau -0.0005. Tipe data ini dikenal juga dengan tipe data **floating point**, **double** atau **real**.

Tipe data float cocok digunakan untuk data angka yang ‘tidak penuh’, seperti nilai IPK, hasil pembagian, atau angka yang diluar jangkauan integer. Tipe data float bisa bernilai positif maupun negatif.

Sama seperti *integer*, jangkauan angka **float** bergantung kepada komputasi prosessor yang digunakan. Pada umumnya *range* angka float mulai dari  $1.7 \times 10^{-308}$  sampai dengan  $1.8 \times 10^{308}$  dengan keakuratan 14 digit.

Berikut contoh penggunaan tipe data float:

```
1 <?php
2   $IPK=3.19;
3   $nilai_tukar=13235.50;
4   $keuntungan=5000000000;
5
6   var_dump($IPK);           // float(3.19)
7   echo "<br />";
8   var_dump($nilai_tukar); // float(13235.5)
9   echo "<br />";
10  var_dump($keuntungan); // float(5000000000)
11 ?>
```



Karena terdapat perbedaan cara penulisan bilangan pecahan antara Amerika, Eropa dan Indonesia. Didalam PHP (dan bahasa pemrograman lainnya) nilai desimal ditulis menggunakan tanda “titik”, bukan “koma”. Nilai 0,87 harus ditulis menjadi 0.87. PHP akan menampilkan pesan error jika sebuah angka ditulis dengan 0,87.

Penulisan angka float di dalam PHP juga boleh menggunakan **scientific notation**, yakni menggunakan angka dengan bilangan “pangkat 10”.

Sebagai contoh, angka 156.23 bisa ditulis sebagai  $1.5623 \times 10^2$ , yang di dalam PHP ditulis: 1.5623E2. Contoh lain, angka 0.00013 bisa ditulis sebagai  $1.3 \times 10^{-4}$ , yang di dalam PHP ditulis menjadi 1.3E-4.

Berikut contoh penggunaannya:

```
1 <?php
2   $var_float1=1.5623E2;
3   $var_float2=1.3E-4;
4
5   var_dump($var_float1); // float(156.23)
6   echo "<br />";
7   var_dump($var_float2); // float(0.00013)
8 ?>
```

## Keterbatasan Perhitungan float

Walaupun kita belum membahas tentang operator perbandingan, namun saya ingin memperlihatkan keterbatasan perhitungan float di dalam PHP. Perhatikan kode berikut ini:

```

1 <?php
2   $x = 10/3;
3   $y = 3.3333333333333;
4
5   var_dump($x); // float(3.3333333333333)
6   echo "<br>";
7   var_dump($y); // float(3.3333333333333)
8   echo "<br>";
9
10  var_dump($x == $y); // bool(false)
11 ?>

```

Kode diatas memperlihatkan di dalam PHP, dua angka yang ‘seharusnya’ sama dianggap berbeda oleh PHP. Contoh berikutnya memperlihatkan hal yang lebih jelas:

```

1 <?php
2   $x = 8 - 6.4;;
3   $y = 1.6;
4
5   var_dump($x); // float(1.6)
6   echo "<br>";
7   var_dump($y); // float(1.6)
8   echo "<br>";
9
10  var_dump($x == $y); // bool(false)
11 ?>

```

Kali ini angka 1.6 dianggap **tidak sama dengan** 1.6. Ini terjadi karena sistem pembulatan yang digunakan oleh PHP.

Jika anda bermaksud menggunakan operasi perbandingan yang melibatkan tipe data float, sebaiknya mempertimbangkan kemungkinan seperti kasus diatas.

## 7.6 Tipe Data String

**String** adalah tipe data yang terdiri dari karakter atau kumpulan karakter. Contoh tipe data string adalah teks seperti: ‘A’, ‘b’, ‘aku’, ‘kamu’, atau ‘Ini adalah sebuah kalimat’.

Karakter yang didukung oleh string PHP terdiri dari 256 karakter ASCII, jadi kita bisa membuat teks khusus seperti: ‘câfè’. Setiap karakter menggunakan memory sebesar **1 byte**. Total, PHP mendukung hingga 2GB (2147483647 karakter) di dalam satu variabel string.



Daftar 256 karakter ASCII bisa anda lihat di [www.ascii-code.com](http://www.ascii-code.com)<sup>1</sup>

Berikut contoh pembuatan tipe data string:

---

<sup>1</sup><http://www.ascii-code.com>

```

1 <?php
2   $huruf = 'A';
3   $nama = "Anto";
4   $situs = "www.duniailkom.com";
5   $kalimat = "Sedang serius belajar PHP";
6
7   var_dump($huruf);    // string(1) "A"
8   echo "<br>";
9   var_dump($nama);    // string(4) "Anto"
10  echo "<br>";
11  var_dump($situs);   // string(18) "www.duniailkom.com"
12  echo "<br>";
13  var_dump($kalimat); // string(25) "Sedang serius belajar PHP"
14 ?>

```

Keempat variabel diatas bertipe **string**. Perhatikan pada hasil tampilan *var\_dump()* terdapat angka di dalam tanda kurung. Angka ini memperlihatkan jumlah karakter yang ada di dalam sebuah string.

Untuk membuat data string, PHP menyediakan 4 cara penulisan:

- Single Quoted
- Double Quoted
- Heredoc
- Nowdoc

Kita akan membahas satu persatu cara penggunaan keempat metode diatas.

## Membuat String Menggunakan Single Quoted

Single quote adalah istilah bahasa inggris yang berarti ‘*tanda kutip satu*’, yakni karakter ( ‘ ). Didalam PHP, sebuah karakter atau kumpulan karakter yang diapit tanda kutip akan dianggap sebagai string. Berikut contohnya:

```

1 <?php
2   $huruf = 'A';
3   $nama = 'Anto';
4   $kalimat = 'Dia berkata "aku akan kembali.."' ;
5
6   var_dump($huruf);    // string(1) "A"
7   echo "<br>";
8   var_dump($nama);    // string(4) "Anto"
9   echo "<br>";
10  var_dump($kalimat); // string(32) "Dia berkata "aku akan kembali.."""
11 ?>

```

Dalam variabel `$kalimat` pada contoh diatas, kita bisa menggunakan tanda kutip dua (") di dalam string yang dibuat menggunakan tanda kutip satu ( ' ). Bagaimana jika kita ingin menulis tanda kutip satu di dalam string?

```
1 <?php
2   $kalimat = 'Saya sedang belajar 'PHP';
3   echo $kalimat;
4   // Parse error: syntax error, unexpected 'PHP' (T_STRING)
5 ?>
```

Untuk karakter khusus seperti ini, kita harus menggunakan tanda *forward slash* ( \ ). Berikut contoh penggunaannya:

```
1 <?php
2   $kalimat = 'Saya sedang belajar \'PHP\'';
3   echo $kalimat;
4   // Saya sedang belajar 'PHP'
5 ?>
```

Dalam pemrograman, karakter *forward slash* () seperti ini disebut sebagai '*escape character*'. **Escape character** digunakan untuk menulis karakter yang tidak bisa ditulis langsung atau karakter yang memiliki arti khusus.

Apabila kita ingin menampilkan karakter "\", maka harus menulisnya 2 kali, seperti contoh berikut:

```
1 <?php
2   $alamat_php = 'Folder PHP berada di C:\\xampp\\\\php\\';
3   echo $alamat_php;
4   // Folder PHP berada di C:\\xampp\\\\php\\
5 ?>
```

## Membuat String Menggunakan Double Quoted

Selain menggunakan tanda kutip satu, PHP juga mendukung penulisan string menggunakan tanda kutip dua ( " ) atau **double quote**. Berikut contoh penggunaannya:

```

1 <?php
2   $huruf = "Z";
3   $nama = "Andika";
4   $kalimat = "Saya berkata 'aku akan pergi..'";
5
6   var_dump($huruf);    // string(1) "Z"
7   echo "<br>";
8   var_dump($nama);     // string(6) "Andika"
9   echo "<br>";
10  var_dump($kalimat); // string(31) "Saya berkata 'aku akan pergi..'"
11 ?>

```

Seperti yang terlihat pada variabel **\$kalimat**, kita juga bisa menuliskan tanda kutip satu di dalam *double quote*. Namun untuk menulis tanda kutip dua, harus menggunakan *escape character* seperti contoh berikut:

```

1 <?php
2   $kalimat = "Saya sedang belajar \"PHP\"";
3   echo $kalimat;
4   // Saya sedang belajar "PHP"
5 ?>

```

Salah satu perbedaan paling mendasar antara *single quote* dengan *double quote* adalah ketika menampilkan variabel. Perhatikan contoh berikut:

```

1 <?php
2   $situs = "www.duniailkom.com";
3
4   $belajar1 = 'Sedang belajar programming di $situs';
5   echo $belajar1;
6   // Sedang belajar programming di $situs
7
8   echo "<br>";
9
10  $belajar2 = "Sedang belajar programming di $situs";
11  echo $belajar2;
12  // Sedang belajar programming di www.duniailkom.com
13 ?>

```

Ketika kita menggunakan *double quote*, nilai variabel juga akan ditampilkan (seperti hasil dari perintah **echo \$belajar2**). Sedangkan jika menggunakan *single quote*, yang ditampilkan adalah nama variabelnya (seperti hasil dari perintah **echo \$belajar1**).

Perbedaan ini juga berlaku jika string langsung ditampilkan menggunakan **echo**:

```

1 <?php
2   $situs = "www.duniailkom.com";
3
4   echo 'Sedang belajar programming di $situs';
5   // Sedang belajar programming di $situs
6
7   echo "<br>";
8
9   echo "Sedang belajar programming di $situs";
10  // Sedang belajar programming di www.duniailkom.com
11 ?>

```

Bagaimana jika variabel yang ingin ditampilkan tersambung dengan string? berikut contohnya:

```

1 <?php
2   $kata="sepak";
3   echo "$kata bola"; // sepak bola
4   echo "<br>";
5
6   echo "$katabola"; // Notice: Undefined variable: katabola
7 ?>

```

Saya ingin menampilkan string “sepakbola” (tanpa spasi diantaranya), tetapi jika langsung disambung, PHP mendeteksi bahwa yang akan dijangan adalah variabel \$katabola. Bagaimana cara memisahkannya? Kita bisa menggunakan pemisah kurung kurawal, seperti berikut:

```

1 <?php
2   $kata="sepak";
3
4   echo "{$kata}bola"; // sepakbola
5 ?>

```

Sekarang, kita bisa menampilkan variabel walaupun tersambung dengan string lain.

Selanjutnya, bagaimana jika ingin menulis karakter ‘\$’ yang bukan bagian dari variabel?

```

1 <?php
2   echo "variabel $situs berisi www.duniailkom.com";
3   // Notice: Undefined variable: situs
4 ?>

```

Solusi untuk hal diatas, kita bisa mengubahnya menjadi *single quote*, atau menggunakan *escape character*:

```

1 <?php
2 echo "variabel \$situs berisi www.duniailkom.com";
3 // variabel $situs berisi www.duniailkom.com
4 ?>

```

Selain *escape character* (") dan (\$), PHP juga menyediakan cara penulisan karakter khusus lain, seperti tab, enter (*new line*), tanda kurung, dll. Berikut tabel *escape character* dalam PHP:

Cara Penulisan String	Karakter Yang Ditampilkan
\"	Karakter Tanda petik dua
\n	Karakter Newline
\r	Karakter Carriage return
\t	Karakter Tab
\	Karakter Backslash
\\$	Karakter Dollar Sign
\{	Karakter Pembuka Kurung Kurawal
\}	Karakter Penutup Kurung Kurawal
\[	Karakter Pembuka Kurung Siku
\]	Karakter Penutup Kurung Kurawal
\0 sampai \777	Karakter ASCII menggunakan nilai oktal
\x0 sampai \xFF	Karakter ASCII menggunakan nilai heksadesimal

Beberapa *escape character* seperti \n atau \t baru bisa terlihat jika output program disimpan kedalam file, bukan ditampilkan di web browser (kita akan membahasnya nanti).

Berikut contoh penggunaan *escape character* untuk karakter ASCII:

```

1 <?php
2 echo "B\xEAl\xE3j\xE4r PHP";
3 // Belajar PHP
4 ?>

```

## Membuat string menggunakan Heredoc

Heredoc adalah cara pembuatan string dengan menggunakan penanda akhir string yang bisa kita buat sendiri. Agar lebih mudah dipahami, perhatikan kode berikut ini:

```

1 <?php
2 $kalimat = <<<habis
3 Sedang belajar bahasa pemrograman PHP,
4 mohon jangan diganggu! lagi serius!!
5 habis;
6
7 var_dump($kalimat);
8 // string(77) "Sedang belajar bahasa pemrograman PHP,
9 // mohon jangan diganggu! lagi serius!!"
10 ?>

```

Ketika membuat variabel `$kalimat`, saya menulis: `<<<habis`. Tanda `<<<` digunakan sebagai awalan pembuatan string menggunakan **heredoc**. Kata “**habis**” merupakan *identifier* khusus untuk mengakhiri string, anda boleh menggunakan kata/karakter apapun, dengan syarat kata ini tidak boleh ada di dalam string dan hanya digunakan pada akhir string.

Untuk mengakhiri string, kita tinggal menulis “**habis**;” (kata ‘habis’ dan tanda titik koma ‘ ; ’). Penutup ini harus berada dalam 1 baris dan tidak boleh diikuti dengan karakter lain apapun, termasuk tab atau spasi sebelum dan sesudahnya.

Berikut contoh lainnya:

```
1 <?php
2 $bahasa="pemrograman PHP";
3 $kalimat = <<<"end"
4 Sedang belajar bahasa $bahasa,
5 mohon jangan diganggu! lagi serius!!
6 end;
7
8 var_dump($kalimat);
9 // string(77) "Sedang belajar bahasa pemrograman PHP,
10 // mohon jangan diganggu! lagi serius!!"
11 ?>
```

Pada contoh kali ini, saya menggunakan kata “**end**” untuk mengakhiri string. Perhatikan bahwa nilai variabel `$bahasa` juga diproses oleh **heredoc**. Kata pembuka heredoc juga boleh diapit dengan tanda kutip `<<<"end"` atau `<<<end`.

## Membuat string menggunakan Nowdoc

Membuat string dengan **nowdoc** sangat mirip dengan **heredoc**. Perbedaannya, di dalam **newdoc** variabel tidak akan diproses. Berikut contohnya:

```
1 <?php
2 $bahasa="pemrograman PHP";
3 $kalimat = <<<'selesai'
4 Sedang belajar bahasa $bahasa,
5 mohon jangan diganggu! lagi serius!!
6 selesai;
7
8 var_dump($kalimat);
9 // string(69) "Sedang belajar bahasa $bahasa,
10 // mohon jangan diganggu! lagi serius!!"
11 ?>
```

Apakah anda bisa melihat perbedaannya?

Di dalam **nowdoc**, kita menggunakan tanda kutip satu (*single quote*) ketika menulis tanda akhir string, yakni: <<<‘selesai’.

 Agar mudah dibedakan, **heredoc** bisa disamakan dengan **double quote**, dimana variabel akan diproses (termasuk karakter khusus yang ditulis dengan escape character). Sedangkan **nowdoc** dapat diibaratkan dengan **single quote**, dimana variabel tidak akan diproses.

## 7.7 Tipe Data Boolean

Tipe data **boolean** adalah tipe data paling sederhana. Tipe data ini hanya memiliki 2 nilai, yaitu **true** (benar) dan **false** (salah). Boolean biasanya digunakan dalam operasi logika seperti kondisi if atau perulangan (looping).

Berikut contoh penulisan tipe data boolean:

```
1 <?php
2     $benar=true;
3     $salah=false;
4
5     var_dump($benar); // bool(true)
6     echo "<br>";
7     var_dump($salah); // bool(false)
8 ?>
```

Penulisan nilai boolean tidak harus menggunakan huruf kecil. Baik **TRUE**, **True**, atau **tRuE** dianggap sama oleh PHP.

```
1 <?php
2     $benar=TRUE;
3     if ($benar)
4     {
5         echo "Anda Benar!"; // Anda Benar!
6     }
7 ?>
```

Peran tipe data boolean baru terlihat jika digunakan untuk kondisi IF seperti diatas. Kita akan membahas alur logika IF dalam bab tersendiri.

## 7.8 Tipe Data Array

**Array** (atau *larik* dalam Bahasa Indonesia) adalah tipe data bentukan yang terdiri dari kumpulan tipe data lain dan tersusun berurutan. Walaupun data ini umumnya sejenis (misalkan: kumpulan

nama, kumpulan angka, dll), di dalam PHP sebuah array bisa terdiri dari berbagai jenis tipe data seperti *integer*, *float*, *string*, *boolean*, bahkan juga *array* lain.

Untuk data yang banyak, menggunakan array jauh lebih mudah dan efisien. Sebagai contoh, jika butuh untuk menyimpan beberapa nama siswa, saya bisa membuatnya seperti berikut ini:

```

1 <?php
2   $siswa0="Andri";
3   $siswa1="Joko";
4   $siswa2="Sukma";
5   $siswa3="Rina";
6   $siswa4="Sari";
7
8   echo $siswa1; // Joko
9 ?>

```

Dengan menggunakan array, saya bisa menulisnya menjadi:

```

1 <?php
2   $siswa = array("Andri", "Joko", "Sukma", "Rina", "Sari");
3   echo $siswa[1]; // Joko
4 ?>

```

Jauh lebih praktis dan efisien. Bayangkan jika butuh 1000 data siswa. Tanpa array, saya harus membuat 1000 variabel. Dengan array, cukup 1 variabel saja.

## Cara Membuat Array

PHP menyediakan 2 cara pembuatan array. Cara pertama dan paling umum adalah dengan menggunakan keyword **array** seperti contoh berikut ini:

```

1 <?php
2   $siswa = array("Andri", "Joko", "Sukma", "Rina", "Sari");
3 ?>

```

Perhatikan bahwa isi dari array dibuat didalam tanda kurung dan masing-masingnya dipisah dengan tanda koma.

Cara kedua adalah dengan ‘**short syntax array**’, yakni menggunakan tanda kurung siku *tanpa keyword “array”*. Fitur ini baru tersedia pada PHP 5.4 keatas:

```

1 <?php
2   $siswa = [ "Andri", "Joko", "Sukma", "Rina", "Sari" ];
3 ?>

```

Ketika kita membuat array, urutan penulisan menentukan posisinya di dalam array. Misalkan, “Andri” berada di urutan pertama array **\$siswa**, “Joko” diurutan kedua, dan seterusnya hingga “Sari” di urutan kelima. Data yang ada di dalam array ini dikenal juga dengan istilah **element array**.

## Cara Mengakses Element Array

Untuk mengakses *element* array, dilakukan dengan cara menulis nama variabel array kemudian diikuti nomor **index** di dalam kurung siku. Berikut contohnya:

```
1 <?php
2     $siswa = array("Andri", "Joko", "Sukma", "Rina", "Sari");
3
4     echo $siswa[2];    // Sukma
5     echo "<br>";
6     echo "Murid itu bernama $siswa[0]"; // Murid itu bernama Andri
7 ?>
```

**Index** adalah urutan element di dalam array. Yang harus wajib diingat, **penomoran index di mulai dari 0, bukan 1**. Dengan demikian, element pertama array memiliki index 0. Element kedua ber-index 1, dst. Penomoran seperti ini bukan hanya di PHP saja, tapi juga dalam bahasa pemrograman lain seperti C, JavaScript, Java, dll.

Di dalam PHP, element array juga bisa terdiri dari campuran berbagai tipe data, seperti contoh berikut:

```
1 <?php
2     $macam2 = array(121, "Joko", 44.99, "Belajar PHP", true);
3
4     echo $macam2[0]; // 121
5     echo "<br>";
6     echo $macam2[1]; // Joko
7     echo "<br>";
8     echo $macam2[2]; // 44.99
9     echo "<br>";
10    echo $macam2[3]; // Belajar PHP
11    echo "<br>";
12    echo $macam2[4]; // 1 (boolean true)
13 ?>
```

Dalam contoh diatas, array **\$macam2** berisi berbagai macam tipe data, mulai dari *integer*, *float*, *string* dan *boolean*. Untuk data yang lebih rumit, kita juga bisa menyimpan array di dalam array.

## Array 2 Dimensi

Array yang kita coba sebelumnya hanya terdiri dari 1 dimensi. Untuk data yang lebih kompleks (seperti titik koordinat), kita bisa membuat array 2 dimensi. Array 2 dimensi sebenarnya sebutan untuk array di dalam array. Berikut contohnya:

```
1 <?php
2     $koordinat = array(
3             array(8,2),
4             array(2,4),
5             array(1,7)
6         );
7     echo $koordinat[0][0]; // 8
8     echo "<br>";
9     echo $koordinat[0][1]; // 2
10    echo "<br>";
11    echo $koordinat[2][1]; // 7
12    echo "<br>";
13 ?>
```

Dalam mengakses array seperti ini, kita menggunakan 2 buah index, seperti `$koordinat[2][1]`. Index pertama untuk mencari posisi array pertama, index kedua untuk posisi array di dalam array. Sebagai latihan, berapakah index untuk menampilkan angka 4? saya bisa menggunakan `$koordinat[1][1]` (ingat, penomoran index dimulai dari nol).

Agar array yang kompleks seperti ini bisa dilihat dengan lebih detail, kita bisa menggunakan fungsi `print_r()`. Fungsi `print_r()` mirip seperti `var_dump()`, tetapi tanpa menampilkan tipe datanya:

```
1 <?php
2     $koordinat = array(array(8,2), array(2,4), array(1,7));
3
4     echo "<pre>";
5     print_r($koordinat);
6     echo "</pre>";
7 ?>
8
9 /* Output-----
10 Array
11 (
12     [0] => Array
13     (
14         [0] => 8
15         [1] => 2
16     )
17     [1] => Array
18     (
19         [0] => 2
20         [1] => 4
21     )
22     [2] => Array
23     (
```

```
24      [0] => 1
25      [1] => 7
26  )
27 )
28 -----*/
```

Saya menambahkan tag `<pre>` agar tampilan menjadi lebih rapi. Dengan menggunakan fungsi `print_r()`, seluruh element array akan terlihat lengkap berserta nomor indexnya.

## Menambah Element Array

Setelah sebuah array didefinisikan, kita bisa menambah element baru ke dalam array tersebut. Berikut contohnya:

```
1 <?php
2 $macam2 = array(121, "Joko", 44.99, "Belajar PHP");
3 $macam2[4] = "DuniaIlkom";
4 $macam2[5] = 212;
5 $macam2[6] = 3.14;
6
7 echo "<pre>";
8 print_r($macam2);
9 echo "</pre>";
10 ?>
11
12 /* Output-----
13 Array
14 (
15     [0] => 121
16     [1] => Joko
17     [2] => 44.99
18     [3] => Belajar PHP
19     [4] => DuniaIlkom
20     [5] => 212
21     [6] => 3.14
22 )
23 -----*/
```

Array `$macam2`, terdiri dari 4 element, dengan nomor index 0, 1, 2, dan 3. Dengan demikian nomor index berikutnya adalah 4. Kita bisa menambah element baru dengan cara yang sama seperti mengisi variabel.

Bagaimana jika kita tidak tahu berapa index terakhir dari array tersebut? Tinggal abaikan penulisan index, seperti contoh berikut:

```

1  <?php
2      $macam2 = array(121, "Joko", 44.99, "Belajar PHP");
3      $macam2[] = "DuniaIlkom";
4      $macam2[] = 212;
5      $macam2[] = 3.14;
6
7      echo "<pre>";
8      print_r($macam2);
9      echo "</pre>";
10
11
12  /* Output-----
13  Array
14  (
15      [0] => 121
16      [1] => Joko
17      [2] => 44.99
18      [3] => Belajar PHP
19      [4] => DuniaIlkom
20      [5] => 212
21      [6] => 3.14
22  )
23 -----*/

```

Jika variabel array ditulis tanpa nomor index, PHP akan menambah element baru di urutan terakhir.

## Associative Array

**Associative array** adalah array yang indexnya bukan dibuat dengan nomor, tetapi dengan objek lain, seperti *string*. Sebagai contoh, kita bisa mengakses array dengan `$siswa["satu"]` atau `$siswa["terbaik"]`.

Untuk membuat *associative array*, caranya hampir sama dengan array ‘normal’, tapi menggunakan tanda “`=>`”. Berikut contohnya:

```

1  <?php
2      $siswa = array(
3          "satu"  => "Andri",
4          "dua"   => "Joko",
5          "tiga"  => "Sukma",
6          "empat" => "Rina"
7      );
8
9      echo $siswa["dua"]; // Joko
10

```

```

11 echo $siswa["empat"]; // Rina
12 ?>

```

Dalam contoh diatas, “satu”, “dua”, “tiga” dan “empat” adalah index array. Index seperti ini sering disebut sebagai label atau **key**. Sedangkan nilainya disebut dengan **value**.

Pembuatan *associative array* dilakukan dengan “key” => “value”, sedangkan pengaksesan nilai array dilakukan dengan menulis `$nama_array["key"]`.

Cara penulisan array diatas sengaja saya buat demikian agar mudah dibaca. Kita juga bisa membuatnya dalam 1 baris:

```

1 <?php
2   $siswa = array("satu"  => "Andri", "dua"   => "Joko",
3                 "tiga"  => "Sukma", "empat" => "Rina" );
4
5   echo "<pre>";
6   print_r($siswa);
7   echo "</pre>";
8 ?>
9
10 /* Output-----
11 Array
12 (
13   [satu] => Andri
14   [dua] => Joko
15   [tiga] => Sukma
16   [empat] => Rina
17 )
18 -----*/

```

Label atau **key** dari *associative array* tidak harus berupa string, tapi juga bisa berupa angka. Penambahan elemen baru juga mirip dengan array normal:

```

1 <?php
2   $assoc = array(
3                 "satu"  => 10000,
4                 "dua"   => "DuniaIlkom",
5                 4 => "Belajar PHP",
6                 1000 => "Jadi programmer"
7   );
8
9   $assoc["baru"]="Data Baru";
10  $assoc[99]="Sembilan puluh sembilan";
11
12  $assoc[]="Ini akan pakai index berapa?";

```

```

13 $assoc[] = 123456;
14
15 echo "<pre>";
16 print_r($assoc);
17 echo "</pre>";
18 ?>
19
20 /* Output-----
21 Array
22 (
23     [satu] => 10000
24     [dua] => DuniaIlkom
25     [4] => Belajar PHP
26     [1000] => Jadi programmer
27     [baru] => Data Baru
28     [99] => Sembilan puluh sembilan
29     [1001] => Ini akan pakai index berapa?
30     [1002] => 123456
31 )
32 -----*/

```

Dalam contoh diatas, saya membuat array `$assoc` dengan berbagai nilai *key* dan *value*. Selanjutnya saya menambahkan beberapa element baru ke dalam array ini.

Perhatikan, ketika *associative array* ditambah element baru tanpa menulis *key*-nya, element tersebut akan disimpan ke dalam *key* integer terakhir. Karena saya memiliki *key* dengan index integer tertinggi “1000”, maka string “*Ini akan pakai index berapa?*” akan disimpan ke dalam index **1001**, begitu juga dengan *value* 123456 akan disimpan ke dalam index **1002**.

Dengan *associative array*, kita bisa membuat struktur data yang rapi. Seperti contoh berikut:

```

1 <?php
2 $siswa = array(
3     "kelas_x"    => array ( "Santi", "Yanto", "Reza"),
4     "kelas_xi"   => array ( "Tia", "Siska", "Nova"),
5     "kelas_xii"  => array ( "Robert", "Rudi", "Alex")
6 );
7
8 echo $siswa["kelas_xi"][0]; //Tia
9 echo "<br>";
10 echo $siswa["kelas_xii"][2]; // Alex
11 echo "<br>";
12 echo $siswa["kelas_x"][1]; // Yanto
13
14 echo "<pre>";
15 print_r($siswa);
16 echo "</pre>";

```

```
17 ?>
18
19 /* Output-----
20 Tia
21 Alex
22 Yanto
23
24 Array
25 (
26     [kelas_x] => Array
27         (
28             [0] => Santi
29             [1] => Yanto
30             [2] => Reza
31         )
32     [kelas_xi] => Array
33         (
34             [0] => Tia
35             [1] => Siska
36             [2] => Nova
37         )
38     [kelas_xii] => Array
39         (
40             [0] => Robert
41             [1] => Rudi
42             [2] => Alex
43         )
44 )
45 -----*/
```

Kali ini saya membuat array `$siswa` yang langsung dikelompokkan menurut kelasnya masing-masing.



Associative array sangat sering digunakan di dalam PHP. Terutama untuk menampilkan data dari database. Kita akan melihat prakteknya ketika membahas tentang PHP dan MySQL.

## 7.9 Tipe Data Object

**Object** adalah tipe data khusus yang juga terdiri dari berbagai data (mirip dengan array). Namun di dalam object kita bisa memiliki *function* sendiri (dikenal sebagai *method*).

Berikut adalah contoh penggunaan tipe data *object*:

```
1  <?php
2      class siswa {
3          public $nama;
4          public $umur;
5          public $tgl_lahir;
6
7          function get_nama(){
8              return $this->nama;
9          }
10     }
11
12     $andi = new siswa;
13     $andi->nama="Andi";
14     $andi->umur=13;
15     $andi->tgl_lahir="13 Des 1990";
16
17     echo "<pre>";
18     print_r($andi);
19     echo "</pre>";
20 ?>
21
22 /* Output-----
23 siswa Object
24 (
25     [nama] => Andi
26     [umur] => 13
27     [tgl_lahir] => 13 Des 1990
28 )
-----*/
```

Tipe data object digunakan dalam pemrograman berbasis object (*object oriented programming / OOP*). Pembahasan mengenai OOP cukup banyak dan tidak akan dibahas dalam buku ini.

## 7.10 Tipe Data Resource

Tipe data **resources** adalah tipe data khusus PHP yang digunakan untuk menyimpan referensi kepada function atau aplikasi di luar PHP. Umumnya kita tidak akan mengakses tipe data ini secara langsung.

Berikut adalah contoh tipe data *resources*, berupa hasil koneksi dari database:

```
1 <?php
2   $koneksi = mysql_connect("localhost", "root", "");
3   var_dump($koneksi);
4   // resource(3) of type (mysql link)
5 ?>
```

## 7.11 Tipe Data NULL

Di dalam pemrograman, **NULL** berarti ‘tidak ada data’, **NULL** berbeda dengan 0, atau string kosong. PHP menyediakan tipe data **NULL** yang hanya bisa diisi dengan nilai **NULL**.

Sebuah variabel di dalam PHP akan bernilai **NULL** jika:

- Diberikan nilai **NULL**.
- Belum memiliki nilai awal.
- Di hapus menggunakan fungsi **unset()**.

Berikut contoh hasil yang dianggap **NULL** di dalam PHP:

```
1 <?php
2   $var1 = 0;
3   var_dump($var1); // int(0)
4
5   $var2 = '';
6   var_dump($var2); // string(0) ""
7
8   $var3 = NULL;
9   var_dump($var3); // NULL
10
11  $var4;
12  var_dump($var4); // NULL + Notice: Undefined variable: var4
13
14  $var5 = 100;
15  unset($var5);
16  var_dump($var5); // NULL + Notice: Undefined variable: var5
17 ?>
```

Dalam contoh diatas, **\$var1** dan **\$var2** tidak dianggap **NULL**. Keduanya adalah tipe data integer 0 dan tipe data string “” (string kosong).

Tipe data **NULL** tidak terlalu sering digunakan. Jika anda mendapati hasil variabel dengan tipe **NULL**, kemungkinan besar terdapat error / kesalahan program.

## 7.12 Type Casting

Type casting adalah proses untuk mengubah sebuah tipe data menjadi tipe data lain (konversi tipe data). Walaupun PHP secara otomatis akan mengubah tipe data tergantung keperluan (akan dibahas pada bab selanjutnya), kita juga bisa mengubah tipe data secara manual.

Untuk proses ini PHP meminjam cara penulisan bahasa C, yakni dengan menulis tipe data tujuan di dalam tanda kurung.

Berikut contohnya:

```
1 <?php
2     $angka_int = 100;
3     var_dump($angka_int);    // int(100)
4     echo "<br>";
5
6     $angka_str = (string) $angka_int;
7     var_dump($angka_str);    // string(3) "100"
8     echo "<br>";
9
10    $angka_bool = (bool) $angka_int;
11    var_dump($angka_bool);    // bool(true)
12    echo "<br>";
13 ?>
```

Dalam kode diatas, saya menggunakan perintah **(string)** `$angka_int` untuk mengubah tipe data `$angka_int` dari integer menjadi string, dan perintah **(bool)** `$angka_int` untuk mengubahnya menjadi boolean.

Berikut adalah perintah yang bisa digunakan untuk mengubah tipe data di dalam PHP:

- (int) atau (integer) – mengubah tipe data menjadi integer.
- (bool) atau (boolean) – mengubah tipe data menjadi boolean.
- (float) atau (double) atau (real) – mengubah tipe data menjadi float.
- (string) – mengubah tipe data menjadi string.
- (array) – mengubah tipe data menjadi array.
- (object) – mengubah tipe data menjadi object.
- (unset) – mengubah tipe data menjadi NULL (dalam PHP 5 keatas).

Perubahan sebuah tipe data menjadi tipe data lain kadang tidak terlalu jelas. Sebagai contoh string “ ” (spasi) akan menjadi boolean **true**, tetapi string “0” akan menjadi boolean **false**. Mari kita lihat bagaimana PHP memperlakukan konversi antar tipe data ini.

## Perubahan Data Menjadi Integer

Jika sebuah variabel di ubah menjadi **integer**, aturan perubahannya adalah sebagai berikut:

- Dari **float**: angka dibelakang desimal akan dihapus.
- Dari **string**: jika terdapat angka di awal string, angka tersebut akan menjadi integer, jika tidak terdapat angka di awal string, akan menjadi integer 0.
- Dari **boolean**: true akan menjadi integer 1, false akan menjadi integer 0.
- Dari **array**: array dengan 0 element akan menjadi integer 0, selain itu akan menjadi integer 1.
- Dari **NULL**: akan menjadi integer 0.
- Dari **object**: akan menjadi integer 1 (menghasilkan error: Notice).
- Dari **resource**: menjadi angka integer yang menunjukkan id resource, ini dihasilkan secara otomatis oleh PHP.

Berikut adalah contoh konversi beberapa tipe data menjadi integer:

```
1 <?php
2 $angka_float=99.75;
3 $string_kosong="";
4 $string_nol="0";
5 $string_spasi=" ";
6 $string_angka="99 ekor kucing";
7 $string_kata="belajar PHP";
8 $bool_true=true;
9 $bool_false=false;
10 $array_kosong = array ();
11 $array_isi = array ("a", "b", "c");
12 $null = null;
13 $object = new stdclass;
14
15 $angka_float = (int) $angka_float;
16 var_dump($angka_float); // int(99)
17 echo "<br>";
18
19 $string_kosong = (int) $string_kosong;
20 var_dump($string_kosong); // int(0)
21 echo "<br>";
22
23 $string_nol = (int) $string_nol;
24 var_dump($string_nol); // int(0)
25 echo "<br>";
26
27 $string_spasi = (int) $string_spasi;
28 var_dump($string_spasi); // int(0)
```

```
29 echo "<br>";
30
31 $string_angka = (int) $string_angka;
32 var_dump($string_angka); // int(99)
33 echo "<br>";
34
35 $string_kata = (int) $string_kata;
36 var_dump($string_kata); // int(0)
37 echo "<br>";
38
39 $bool_true = (int) $bool_true;
40 var_dump($bool_true); // int(1)
41 echo "<br>";
42
43 $bool_false = (int) $bool_false;
44 var_dump($bool_false); // int(0)
45 echo "<br>";
46
47 $array_kosong = (int) $array_kosong;
48 var_dump($array_kosong); // int(0)
49 echo "<br>";
50
51 $array_isi = (int) $array_isi;
52 var_dump($array_isi); // int(1)
53 echo "<br>";
54
55 $null = (int) $null;
56 var_dump($null); // int(0)
57 echo "<br>";
58
59 $object = (int) $object;
60 var_dump($object); // int(1)
61 // Notice: Object of class stdClass could not be converted to int
62 ?>
```

## Perubahan Data Menjadi Float

Perubahan sebuah data menjadi float sama dengan konversi menjadi integer, namun jika data string memiliki angka desimal di awal, akan dikonversi menjadi angka float.

```
1 <?php
2     $string_angka="9.75 kg obat";
3
4     $string_angka = (float) $string_angka;
5     var_dump($string_angka); // float(9.75)
6 ?>
```

## Perubahan Data Menjadi String

Jika sebuah variabel di ubah menjadi string, aturan perubahannya adalah sebagai berikut:

- Dari **integer** dan **float**: seluruh angka tetap, namun bertipe string.
- Dari **boolean**: true akan menjadi string “1”, false akan menjadi string kosong “”.
- Dari **array**: akan menjadi string “Array” (menghasilkan error: Notice).
- Dari **NULL**: menjadi string kosong “”.
- Dari **object**: jika objek memiliki method `_toString()`, akan menampilkan hasil method tersebut, jika tidak akan menghasilkan error (*catchable fatal error*).
- Dari **resource**: menjadi string “Resource id #1” dimana angka 1 menunjukkan id resource, ini dihasilkan secara otomatis oleh PHP.

Berikut adalah contoh konversi beberapa tipe data menjadi string:

```
1 <?php
2     $angka_nol=0;
3     $angka_int=12;
4     $angka_float=99.75;
5     $bool_true=true;
6     $bool_false=false;
7     $array_isi = array ("a", "b", "c");
8     $null = null;
9     $object = new stdClass;
10    $resource = mysql_connect("localhost", "root", "");
11
12    $angka_nol = (string) $angka_nol;
13    var_dump($angka_nol); // string(1) "0"
14    echo "<br>";
15
16    $angka_int = (string) $angka_int;
17    var_dump($angka_int); // string(2) "12"
18    echo "<br>";
19
20    $angka_float = (string) $angka_float;
21    var_dump($angka_float); // string(5) "99.75"
22    echo "<br>";
```

```

23
24     $bool_true = (string) $bool_true;
25     var_dump($bool_true); // string(1) "1"
26     echo "<br>";
27
28     $bool_false = (string) $bool_false;
29     var_dump($bool_false); // string(0) ""
30     echo "<br>";
31
32     $array_isi = (string) $array_isi;
33     var_dump($array_isi); // string(5) "Array" (dengan error Notice)
34     echo "<br>";
35
36     $null = (string) $null;
37     var_dump($null); // string(0) ""
38     echo "<br>";
39
40     $resource = (string) $resource;
41     var_dump($resource); // string(14) "Resource id #3"
42     echo "<br>";
43
44     $object = (string) $object;
45     var_dump($object); // error
46     // Catchable fatal error: Object of class stdClass
47     // could not be converted to string
48 ?>

```

Jika kita menggunakan perintah `echo` kepada sebuah variabel, PHP secara otomatis akan menkonversi tipe data tersebut menjadi string:

```

1 <?php
2     $bool_true=true;
3     $bool_false=false;
4
5     echo $bool_true; // 1
6     echo $bool_false; // "" (string kosong)
7 ?>

```

## Perubahan Data Menjadi Boolean

Perubahan tipe data menjadi boolean cukup banyak digunakan, terutama untuk penentu alur program dengan menggunakan logika IF ELSE. Berikut adalah aturan PHP jika sebuah tipe data dikonversi menjadi boolean:

- Dari **integer**: angka 0 menjadi boolean false, selebihnya menjadi true (termasuk angka negatif).

- Dari **float**: angka 0.0 menjadi boolean false, selebihnya menjadi true (termasuk angka negatif dan angka yang sangat kecil, seperti 0.000001).
- Dari **string**: string "0" dan string kosong "" akan menjadi boolean false, selebihnya menjadi true (termasuk string spasi " ").
- Dari **array**: array dengan 0 element akan menjadi false, array dengan paling sedikit memiliki 1 element menjadi boolean true.
- Dari **NULL**: menjadi boolean false.
- Dari **object**: menjadi boolean true.
- Dari **resource**: menjadi boolean true.

Berikut adalah contoh konversi beberapa tipe data menjadi string:

```
1 <?php
2 $angka_nol=0;
3 $angka_int=12;
4 $angka_float=99.75;
5 $angka_float_kecil=0.000001;
6 $array_kosong = array ();
7 $array_isi = array ("a", "b", "c");
8 $null = null;
9 $object = new stdClass;
10 $resource = mysql_connect("localhost", "root", "");
11
12 $angka_nol = (bool) $angka_nol;
13 var_dump($angka_nol); // bool(false)
14 echo "<br>";
15
16 $angka_int = (bool) $angka_int;
17 var_dump($angka_int); // bool(true)
18 echo "<br>";
19
20 $angka_float = (bool) $angka_float;
21 var_dump($angka_float); // bool(true)
22 echo "<br>";
23
24 $angka_float_kecil = (bool) $angka_float_kecil;
25 var_dump($angka_float_kecil); // bool(true)
26 echo "<br>";
27
28 $array_kosong = (bool) $array_kosong;
29 var_dump($array_kosong); // bool(false)
30 echo "<br>";
31
32 $array_isi = (bool) $array_isi;
33 var_dump($array_isi); // bool(true)
```

```
34 echo "<br>";  
35  
36 $null = (bool) $null;  
37 var_dump($null); // bool(false)  
38 echo "<br>";  
39  
40 $object = (bool) $object;  
41 var_dump($object); // bool(true)  
42 echo "<br>";  
43  
44 $resource = (bool) $resource;  
45 var_dump($resource); // bool(true)  
46 ?>
```

Konversi tipe data array, object, null dan resources tidak sering digunakan. Anda bisa melakukan percobaan sendiri jika ingin mengetahui hasilnya.

---

Dalam bab ini kita telah mempelajari berbagai tipe data di dalam PHP. Berikutnya, kita akan lanjut membahas operasi apa yang bisa dilakukan dengan tipe data ini dalam Operator PHP.

# 8. Operator PHP

Setelah memahami berbagai tipe data PHP, kali ini kita akan mencoba melakukan ‘sesuatu’ dengan data tersebut. Dalam bab saya akan membahas tentang **operator**, **operand** dan **jenis-jenis operator** di dalam PHP.

## 8.1 Pengertian Operator dan Operand

**Operator** adalah sesuatu yang menghasilkan nilai dari satu atau lebih data. Sebagai contoh, tanda tambah ( + ) adalah operator aritmatika yang menghasilkan nilai dari penambahan dua buah angka.

**Operand** adalah nilai asal yang digunakan oleh operator. Sebagai contoh, dalam operasi  $5+2$ , angka 5 dan 2 disebut sebagai **operand**. Operasi tersebut membutuhkan 2 buah operand dan 1 operator.

Operator di dalam PHP banyak meminjam contoh dari bahasa C dan Perl.

## 8.2 Jenis-Jenis Operator PHP

Berdasarkan jumlah operand, operator dapat dibedakan menjadi 3 jenis, yaitu **operator unary**, **binary** dan **ternary**.

- **Operator unary** adalah operator yang hanya memiliki 1 operand, contohnya operator – (tanda minus) dan + (tanda plus). Tanda minus digunakan untuk membuat angka menjadi negatif, contohnya:  $-5$ , sedangkan tanda plus digunakan untuk menegaskan nilai positif, seperti:  $+5$ .
- **Operator binary** adalah operator yang memiliki 2 operand. Operator jenis ini adalah yang paling banyak, misalkan operator ( \* ) untuk perkalian seperti  $5 * 2$ , atau operator ( / ) untuk pembagian, seperti:  $10/3$ .
- **Operator ternary** adalah operator yang memiliki 3 **operand**. Di dalam PHP hanya dikenal 1 operator **ternary**, yaitu operator kondisi ( ? : ).

Jika dilihat berdasarkan jenis operasinya, operator PHP dapat dibedakan menjadi 10 operator:

- Operator Aritmatika
- Operator Increment dan Decrement
- Operator Perbandingan
- Operator Logika
- Operator String

- Operator Bitwise
- Operator Assignment
- Operator Error Control
- Operator Array
- Operator Type

Kita akan membahas 8 operator di dalam bab ini. Operator array akan dibahas dalam bab tentang array sedangkan operator type memerlukan pemahaman tentang object.

## 8.3 Operator Aritmatika

**Operator Aritmatika** adalah operator matematis yang kita gunakan sehari-hari. PHP mengenal 8 operator aritmatika, yakni penambahan, pengurangan, perkalian, pembagian, modulus, plus, minus dan eksponensial.

Tabel berikut merangkum jenis operator aritmatika dalam PHP:

Nama Operator	Operator	Contoh	Hasil
positif / plus	+	+\$a	nilai positif dari \$a
negatif / minus	-	-\$a	nilai negatif dari \$a
penambahan	+	\$a + \$b	total dari \$a dan \$b
pengurangan	-	\$a - \$b	selisih dari \$a dan \$b
perkalian	*	\$a * \$b	hasil kali dari \$a dan \$b
div/pembagian	/	\$a / \$b	hasil bagi dari \$a dan \$b
modulus/sisa hasil bagi	%	\$a % \$b	sisa pembagian \$a bagi \$b
eksponensial*	**	\$a ** \$b	hasil dari \$a pangkat \$b

\* Khusus untuk operator eksponensial, baru ditambahkan pada PHP 5.6.

Dari ke 8 operator aritmatika diatas, operator **modulus** ( $$a \% $b$ ) mungkin terdengar asing. Operator ini digunakan untuk mencari sisa hasil bagi. Sebagai contoh,  $10 \% 3 = 1$ , karena  $3^*3 = 9$  (sisa 1). Contoh lain,  $15 \% 5 = 0$ , karena 15 habis dibagi 5.

Operator aritmatika relatif mudah dipahami, karena kita telah terbiasa menggunakanannya. Berikut contoh kode program PHP untuk operator aritmatika:

```

1  <?php
2      $hasil1 = +11;
3      $hasil2 = -3;
4      $hasil3 = 3+5;
5      $hasil4 = 8-4.5;
6      $hasil5 = 2*5;
7      $hasil6 = 3+8/5-3;
8      $hasil7 = 10 % 4;
9      $hasil8 = 2**4;
10
11     var_dump($hasil1); // int(11)
12     echo "<br>";
13     var_dump($hasil2); // int(-3)
14     echo "<br>";
15     var_dump($hasil3); // int(8)
16     echo "<br>";
17     var_dump($hasil4); // float(3.5)
18     echo "<br>";
19     var_dump($hasil5); // int(10)
20     echo "<br>";
21     var_dump($hasil6); // float(1.6)
22     echo "<br>";
23     var_dump($hasil7); // int(2)
24     echo "<br>";
25     var_dump($hasil8); // int(16)
26 ?>

```

Dari kode diatas, terlihat bagaimana PHP menyesuaikan jenis tipe data dengan hasil operasi. Operasi aritmatika yang menghasilkan bilangan bulat, ditampung ke dalam tipe data **integer**, selain itu ditampung dengan tipe data **float**.

Operator aritmatika juga memiliki aturan prioritas. Operasi perkalian dan pembagian memiliki prioritas lebih tinggi daripada penambahan dan pengurangan. Untuk mengubah urutan ini, kita bisa menggunakan tanda kurung, seperti contoh berikut:

```

1  <?php
2      $hasil1=3+8/5-3*9+9;
3      var_dump($hasil1);      // float(-13.4)
4      echo "<br>";
5
6      $hasil2=((3+8)/(5-3))*(9+9);
7      var_dump($hasil2);      // float(99)
8 ?>

```

Penggunaan tanda kurung untuk operasi yang ‘rumit’ seperti diatas sangat disarankan agar kode program lebih mudah dibaca.

Operator aritmatika juga bisa dijalankan untuk tipe data bukan angka. PHP akan mengkonversinya menjadi angka (integer atau float) secara otomatis. Mengenai aturan konversi tipe data ke integer/float telah kita bahas pada bab sebelumnya.

Berikut contoh operasi aritmatika pada variabel bukan angka:

```
1 <?php
2     $hasil = "180" + 20;
3     var_dump($hasil); // int(200)
4     echo "<br>";
5
6     $hasil = "9 anak" + 3;
7     var_dump($hasil); // int(12)
8     echo "<br>";
9
10    $hasil = "5 ekor sapi" + "3 ekor kambing";
11    var_dump($hasil); // int(8)
12    echo "<br>";
13
14    $hasil = "17.5cm" + "1.99cm";
15    var_dump($hasil); // float(19.49)
16    echo "<br>";
17
18    $hasil = true + true + true;
19    var_dump($hasil); // int(3)
20 ?>
```

Dalam contoh diatas, setiap string akan dikonversi menjadi integer dan float, boolean true juga dikonversi menjadi angka 1, sehingga **true + true + true** akan dikonversi menjadi **1+1+1**. Walaupun operasi seperti ini tidak error, tapi sangat tidak disarankan.

## 8.4 Operator Increment dan Decrement

Operator **increment** dan **decrement** adalah penyebutan untuk operasi seperti `$a++` atau `$a--`. Kedua operator ini sering digunakan dalam perulangan (*looping*).

Operator **increment** digunakan untuk menambah variabel sebanyak 1 angka. Penulisannya menggunakan tanda tambah 2 kali baik diawal maupun diakhir variabel, seperti contoh berikut: `$a++` atau `++$a`. Operasi `$a++` dapat disamakan dengan penulisan singkat dari `$a = $a + 1`.

Operator **decrement** digunakan untuk mengurangi variabel sebanyak 1 angka. Penulisannya menggunakan tanda kurang 2 kali baik diawal maupun diakhir variabel, seperti contoh berikut: `$a--` atau `--$a`. Operasi `$a--` dapat disamakan dengan penulisan singkat dari `$a = $a - 1`.

Walaupun sama-sama menambahkan dan mengurangkan angka, operator `++$a` dan `$a++` diproses dengan cara berbeda. Penulisan `++$a` dinamakan dengan **pre-increment**, sedangkan penulisan

`$a++` dinamakan **post-increment**. Demikian juga dengan `--$a` yang disebut dengan **pre-decrement** dan `$a--` sebagai **post-decrement**.

Tabel berikut memperlihatkan perbedaan keempat penulisan ini:

Nama	Contoh	Hasil
pre-increment	<code>++\$a</code>	tambah nilai <code>\$a</code> dengan 1, lalu kirim nilai <code>\$a</code>
post-increment	<code>\$a++</code>	kirim nilai <code>\$a</code> , lalu tambah nilai <code>\$a</code> dengan 1
pre-decrement	<code>-\$a</code>	kurangi nilai <code>\$a</code> dengan 1, lalu kirim nilai <code>\$a</code>
post-decrement	<code>\$a-</code>	kirim nilai <code>\$a</code> , lalu kurangi nilai <code>\$a</code> dengan 1

Berikut hasil yang didapat di dalam kode PHP:

```

1 <?php
2   $a = 5;
3   echo ++$a;    // 6
4   echo $a;      // 6
5   echo "<br>";
6
7   $b = 5;
8   echo $b++;    // 5
9   echo $b;      // 6
10  echo "<br>";
11
12  $a = 5;
13  echo --$a;    // 4
14  echo $a;      // 4
15  echo "<br>";
16
17  $b = 5;
18  echo $b--;    // 5
19  echo $b;      // 4
20 ?>

```

Terlihat bahwa **post-increment** (`$a++`) akan memberikan hasilnya dulu, baru menambahkan nilai variabel `$a` sebanyak 1 angka. Sedangkan untuk **pre-increment**, `$a` akan ditambahkan 1 angka, baru nilainya ditampilkan. Begitu juga hal nya dengan operasi **post-decrement** dan **pre-decrement**.

Operator increment dan decrement umumnya digunakan dalam perulangan (akan dibahas dalam bab selanjutnya).

## 8.5 Operator Perbandingan

Operator perbandingan digunakan untuk membandingkan 2 buah operand atau lebih. Hasil dari operator ini berupa nilai boolean **true** atau **false**.

Sebagai contoh, operasi `5 > 3` akan menghasilkan nilai `true`, karena 5 lebih besar dari 3. Sedangkan operasi `1 > 2` akan menghasilkan nilai `false`, karena 1 tidak lebih besar dari 2.

Tabel berikut merangkum operator perbandingan di dalam PHP:

Nama Operator	Operator	Contoh	Hasil
Sama dengan	<code>==</code>	<code>\$a == \$b</code>	TRUE jika nilai <code>\$a</code> sama dengan <code>\$b</code>
Identik dengan	<code>===</code>	<code>\$a === \$b</code>	TRUE jika nilai <code>\$a</code> sama dengan <code>\$b</code> , dan memiliki tipe data yang sama
Tidak sama dengan	<code>!=</code>	<code>\$a != \$b</code>	TRUE jika nilai <code>\$a</code> tidak sama dengan <code>\$b</code>
Tidak sama dengan	<code>&lt;&gt;</code>	<code>\$a &lt;&gt; \$b</code>	TRUE jika nilai <code>\$a</code> tidak sama dengan <code>\$b</code>
Tidak identik dengan	<code>!==</code>	<code>\$a !== \$b</code>	TRUE jika nilai <code>\$a</code> tidak sama dengan <code>\$b</code> , dan memiliki tipe data yang tidak sama
Kurang dari	<code>&lt;</code>	<code>\$a &lt; \$b</code>	TRUE jika nilai <code>\$a</code> kurang dari <code>\$b</code>
Lebih dari	<code>&gt;</code>	<code>\$a &gt; \$b</code>	TRUE jika nilai <code>\$a</code> lebih dari <code>\$b</code>
Kurang dari atau sama dengan	<code>&lt;=</code>	<code>\$a &lt;= \$b</code>	TRUE jika nilai <code>\$a</code> kurang dari atau sama dengan <code>\$b</code>
Lebih dari atau sama dengan	<code>&gt;=</code>	<code>\$a &gt;= \$b</code>	TRUE jika nilai <code>\$a</code> lebih dari atau sama dengan <code>\$b</code>

Berikut contoh penggunaan operator perbandingan diatas:

```

1 <?php
2     var_dump(12<14); // bool(true)
3     echo "<br />";
4
5     var_dump(14<14); // bool(false)
6     echo "<br />";
7
8     var_dump(14<=14); // bool(true)
9     echo "<br />";
10
11    var_dump(10<>14); // bool(true)
12    echo "<br />";
13
14    var_dump(15==10); // bool(false)
15    echo "<br />";
16
17    var_dump(10==10); // bool(true)
18    echo "<br />";
19
20    var_dump(150==1.5e2); // bool(true)
21 ?>

```

Operasi perbandingan tidak hanya bisa dilakukan untuk tipe data angka saja (*integer* dan *float*),

namun juga bisa untuk tipe data lain, seperti *string*, *array*, bahkan *object*.

Perhatikan kode berikut ini:

```
1 <?php
2     var_dump('andi' < 'andri'); // bool(true)
3     echo "<br>";
4     var_dump('anto' < 'anti'); // bool(false)
5 ?>
```

Jika yang dibandingkan adalah sesama string, PHP akan membandingkan urutan huruf berdasarkan abjad.

Pada contoh pertama, huruf ‘a’, ‘n’ dan ‘d’ sama-sama terdapat di masing-masing string, namun pada saat karakter ke-4, huruf ‘i’ akan dibandingkan dengan huruf ‘r’ (andi vs andri), sehingga andi akan dianggap lebih kecil daripada andri (urutan abjad ‘i’ lebih dulu mucul daripada ‘r’).

Sedangkan pada contoh kedua, huruf ‘o’ lebih belakangan urutannya dalam abjad dibandingkan huruf ‘i’ (anto vs anti) sehingga ‘anto’ < ‘anti’ akan menghasilkan **false**.

Hal yang sama juga berlaku untuk tipe data array:

```
1 <?php
2     $siswa1 = array("anto", "andra");
3     $siswa2 = array("anto", "andri");
4     var_dump($siswa1 < $siswa2); // bool(true)
5 ?>
```

Yang cukup menarik adalah tipe data boolean **false**, dianggap lebih kecil daripada **true**:

```
1 Perhatikan kode berikut ini:
2 <?php
3     var_dump(false < true); // bool(true)
4 ?>
```

PHP juga membolehkan operasi perbandingan antar data tipe, dan hal ini sering menjadi error yang susah dideteksi. Berikut contohnya:

```
1 <?php
2     var_dump('duniailkom' == 0); // bool(true)
3 ?>
```

Operasi perbandingan diatas akan menghasilkan **true**!

Hal ini terjadi karena PHP akan menkonversi string menjadi angka terlebih dahulu, kemudian baru membandingkan keduanya. String ‘**duniailkom**’ dikonversi menjadi angka 0, sehingga **0 == 0** adalah **true** (mengenai aturan konversi ini telah kita bahas pada bab sebelumnya).

Aturan konversi dan perbandingan antar tipe data ini dapat dilihat dalam tabel berikut:

Tipe operand 1	Tipe operand 2	Proses perbandingan
angka	angka	angka
string berupa angka	string berupa angka	string dikonversi menjadi angka
string berupa angka	number	string dikonversi menjadi angka
string berupa angka	string yang bukan angka	string dikonversi menjadi angka
string yang bukan angka	angka	string dikonversi menjadi angka
string yang bukan angka	string yang bukan angka	urutan huruf dalam alfabet
boolean	boolean	false lebih kecil dari true
boolean	tipe apa saja	operand kedua dikonversi menjadi boolean
array	array	array dengan data yang paling sedikit dianggap lebih kecil
null	tipe apa saja	null akan dikonversi menjadi string kosong atau false
array	tipe data selain array dan objek	array selalu lebih besar
objek	tipe data selain objek	objek selalu lebih besar
objek	objek	dibandingkan seluruh element objek

Berikut adalah contoh hasil membandingkan antar tipe data:

```

1 <?php
2  var_dump(10=='10'); // bool(true)
3  echo "<br />";
4
5  var_dump(10=='10'); // bool(false)
6  echo "<br />";
7
8  var_dump('Andi'==0); // bool(true)
9  echo "<br />";
10
11 var_dump('10 ayam'==10); // bool(true)
12 echo "<br />";
13
14 var_dump('10 ayam'==10); // bool(false)
15 echo "<br />";
16
17 var_dump(true<false); // bool(false)
18 echo "<br />";
19
20 $siswa1 = array("anto","andri");
21 $siswa2 = array("anto","andri");
22 var_dump($siswa1==$siswa2); // bool(true)

```

23 ?&gt;

Perbedaan antara operator `==` dengan `===` juga perlu dibahas. Operator ‘*sama dengan*’ (`==`) akan membandingkan 2 nilai tanpa memperhatikan tipe datanya, sehingga ‘aku’ `== 0` adalah `true`. Jika kita ingin proses perbandingan juga memperhatikan tipe data yang sama, maka gunakan operator ‘*identik dengan*’ (`==`). Operator ini akan menghasilkan `true` jika nilai data sama **dan** tipe data yang dibandingkan juga sama.

Operator perbandingan sering digunakan untuk membuat percabangan program, seperti logika IF atau switch.

## 8.6 Operator Logika

**Operator logika** adalah operator yang digunakan untuk membandingkan 2 kondisi logika, yaitu logika benar (`true`) dan logika salah (`false`). Nilai yang dibandingkan harus bertipe boolean. Jika tidak, PHP akan menkonversinya secara otomatis.

Tabel berikut merangkum jenis-jenis operator logika dalam PHP:

Nama Operator	Operator	Contoh	Hasil
and	and atau <code>&amp;&amp;</code>	<code>\$a and \$b</code> atau <code>\$a &amp;&amp; \$b</code>	true, jika <code>\$a</code> dan <code>\$b</code> sama-sama bernilai true
or	or atau <code>  </code>	<code>\$a or \$b</code> atau <code>\$a    \$b</code>	true, jika salah satu dari <code>\$a</code> atau <code>\$b</code> bernilai true
xor	<code>xor</code>	<code>\$a xor \$b</code>	true, jika salah satu dari <code>\$a</code> atau <code>\$b</code> bernilai true, tapi tidak keduanya.
not	<code>!</code>	<code>! \$a</code>	true, jika <code>\$a</code> bernilai false.

Penulisan operator logika adalah *case insensitive*, artinya tidak memperhatikan huruf besar atau kecil. Baik ‘`and`’, ‘`AND`’, atau ‘`aNd`’ dianggap sah oleh PHP.

Operator ‘`and`’ dan ‘`&&`’ sama-sama berarti logika ‘`and`’, namun operator ‘`&&`’ memiliki urutan prioritas yang lebih tinggi. Sebagai contoh, operasi `$hasil = $a and $b && $c` akan diproses sebagai `$hasil = $a and ($b && $c)`. Begitu juga dengan operator ‘`or`’ dan ‘`||`’, dimana ‘`||`’ memiliki prioritas lebih tinggi daripada ‘`or`’.

Berikut contoh penggunaan operator logika di dalam PHP:

```
1  <?php
2      $hasil = (true and false);
3      var_dump($hasil); // bool(false)
4      echo "<br/>";
5
6      $hasil = (true or false);
7      var_dump($hasil); // bool(true)
8      echo "<br/>";
9
10     $hasil = (true xor false);
11     var_dump($hasil); // bool(true)
12     echo "<br/>";
13
14     $hasil = false;
15     var_dump(!$hasil); // bool(true)
16     echo "<br/>";
17
18     $hasil = (false or true && false);
19     var_dump($hasil); // bool(false)
20     echo "<br/>";
21
22     $hasil = ('duniailkom and true');
23     var_dump($hasil); // bool(true)
24     echo "<br/>";
25
26     $hasil = ('000 or false');
27     var_dump($hasil); // bool(true)
28 ?>
```

Dalam kode diatas, saya mencoba menggunakan beberapa operasi logika. Perhatikan bahwa **false or true && false** akan diproses sebagai **false or (true && false)** sehingga hasilnya adalah **false**.

Untuk operasi ‘**duniailkom** and **true**’, string ‘**duniailkom**’ akan dikonversi menjadi boolean **true**, sehingga ‘**true and true**’ menghasilkan **true**. Hal serupa juga terjadi pada ‘**000**’ or **false**, string ‘**000**’ akan dikonversi menjadi boolean **true**, sehingga **true or false** menghasilkan **true**.

Operasi logika di dalam PHP juga menerapkan prinsip yang disebut dengan **short circuit**. Yaitu jika dengan memeriksa satu nilai saja sudah dipastikan hasil logikanya, maka perintah selanjutnya tidak perlu dijalankan.

Agar dapat dipahami, perhatikan contoh berikut:

```
$hasil = $a and $b and $c and $d;
```

Jika pada saat program dijalankan **\$a** sudah bernilai **false**, maka variabel **\$b**, **\$c** dan **\$d** tidak akan diperiksa lagi. Karena, apapun nilai variabel setelahnya, hasilnya akan tetap **false**.

Hal ini akan berguna untuk kasus-kasus tertentu, seperti contoh berikut:

```
<?php
$result = fopen($filename) or exit();
?>
```

Fungsi **exit()** yang akan membuat program PHP berhenti diproses, tidak akan dijalankan selama fungsi **fopen(\$filename)** menghasilkan nilai true.

Prinsip *short-circuit* ini sering digunakan dalam contoh-contoh aplikasi PHP lanjutan.

## 8.7 Operator String

Dalam PHP, hanya terdapat 1 jenis operator string, yakni operasi penyambungan string (*string concatenation*). Operator ini menggunakan karakter titik ( . ). Jika operand bukan string, akan dikonversi menjadi string secara otomatis.

Berikut contoh operasi penyambungan string dalam PHP:

```
1 <?php
2   $hasil = "Belajar"."PHP";
3   echo $hasil; // BelajarPHP
4   echo "<br>";
5
6   $a = "Ssst!";
7   $b = " lagi";
8   $c = " serius";
9   $d = " belajar PHP";
10  $hasil = $a.$b.$c.$d;
11  echo $hasil; // Ssst! lagi serius belajar PHP
12  echo "<br>";
13
14 $hasil = 9 . " ekor anak beruang";
15 echo $hasil; // 9 ekor anak beruang
16 echo "<br>";
17
18 $hasil = true . " adalah data boolean";
19 echo $hasil; // 1 adalah data boolean
20 ?>
```

Dalam contoh-contoh diatas, saya juga mencoba menyambung tipe data lain dengan string, dan seperti yang terlihat, PHP secara otomatis menkonversinya menjadi string. Boolean **true** akan dikonversi menjadi string "1".

Operator 'titik' ini juga bisa digunakan untuk menggabungkan variabel dengan string di dalam perintah **echo**, seperti contoh berikut ini:

```

1 <?php
2   $a = 3;
3   $b = " ekor kucing ";
4   echo 'Andi memiliki '.$a.$b.'betina.';
5   // Andi memiliki 3 ekor kucing betina.
6 ?>

```

Penulisan seperti ini sangat sering digunakan. Selain dengan operator titik ( . ), kita juga bisa menggabungkan 2 variabel string dengan menggunakan **double quote**.

Ketika membuat string menggunakan **double quote** (tanda kutip dua), setiap variabel akan diproses menjadi nilainya. Hal ini bisa dimanfaatkan untuk penggabungan string. Perhatikan kode program berikut:

```

1 <?php
2   $a = "Belajar";
3   $b = "PHP";
4   $c = "mengasyikkan";
5   $hasil = "$a$b$c";
6   echo $hasil; // BelajarPHPmengasyikkan
7 ?>

```

Sekarang, ketiga variabel string disatukan di dalam variabel **\$hasil**.

## 8.8 Operator Bitwise

Operator **bitwise** adalah operator khusus untuk menangani operasi logika *bilangan biner*. **Bilangan biner** atau *binary* adalah jenis bilangan yang hanya terdiri dari 2 jenis angka, yakni 0 dan 1. Operator **bitwise** tidak terlalu sering digunakan, kecuali jika anda membuat program yang langsung berkaitan dengan pemrosesan bilangan biner (biasanya untuk pemrosesan bit-bit karakter).



Dalam bagian ini saya tidak akan menjelaskan tentang apa itu bilangan biner, karena penerapannya di dalam PHP tidak terlalu banyak. Anda boleh melewati pembahasan operator ini jika kurang dimengerti.

PHP menyediakan 6 operator bitwise. Lengkapnya dapat dilihat dalam tabel berikut:

Nama	Operator	Contoh	Hasil
and	&	\$a & \$b	1, jika kedua bit bernilai 1.
or (inclusive or)		\$a   \$b	1, jika salah satu bit bernilai 1.
xor (exclusive or)	^	\$a ^ \$b	1, jika salah satu bit bernilai 1, tapi bukan keduanya
not (negasi)	~	~ \$a	bit 0 menjadi 1, dan bit 1 menjadi 0.

Nama	Operator	Contoh	Hasil
shift left	<<	\$a << \$b	menggeser sebanyak \$b bit ke kiri (setiap 1 kali pergeseran = kelipatan 2)
shift right	>>	\$a >> \$b	menggeser sebanyak \$b bit ke kanan (setiap 1 kali pergeseran = bagi 2)

Berikut adalah contoh penggunaan dan perhitungan operator bitwise di dalam PHP:

```

1 <?php
2     $a=0b10110101;      // 10110101 biner = 181 desimal
3     $b=0b01101100;      // 01101100 biner = 108 desimal
4
5     $hasil = $a & $b;
6     echo $hasil;         // 36 (desimal) = 00100100 biner
7     echo "<br />";
8
9     $hasil = $a | $b;
10    echo $hasil;         // 253 (desimal) = 11111101 biner
11    echo "<br />";
12
13    $hasil = $a ^ $b;
14    echo $hasil;         // 217 (desimal) = 11011001 biner
15    echo "<br />";
16
17    $hasil = ~$a;
18    echo $hasil;         // -182 (desimal)
19    echo "<br />";
20
21    $hasil = $a >> 1;
22    echo $hasil;         // 90 (desimal) = 1011010 biner
23    echo "<br />";
24
25    $hasil = $b << 2;
26    echo $hasil;         // 432 (desimal) = 0110110000 biner
27 ?>

```

Dalam contoh kode PHP diatas, saya mendefenisikan 2 variabel yakni \$a dan \$b. Keduanya diisi dengan nilai awal berupa angka biner (cara penulisan angka biner telah kita bahas dalam bab sebelumnya).

Variabel \$a berisi angka biner: **10110101**, yang nilai desimalnya adalah 181, sedangkan variabel \$b berisi angka biner: **01101100**, yang nilai desimalnya adalah 108.

Pada contoh 1, saya melakukan operasi **and (&)** terhadap kedua variabel. Operasi bitwise “**and**” akan memproses bit per bit dari kedua variabel. Jika kedua bit sama-sama bernilai 1, maka hasilnya juga 1. Selain kondisi tersebut, akan menjadi 0. Berikut proses perhitungannya:

```
$a = 10110101
$b = 01101100
-----
$a & $b = 00100100 = 36 (desimal)
```

Dan perintah **echo**, terlihat hasilnya adalah 36 (dalam bentuk desimal). Ini karena PHP otomatis mengkonversi nilai biner ke bentuk angka desimal ketika ditampilkan.

Contoh kedua, adalah operasi **or** ( | ). Operasi **or** akan menghasilkan 0 jika kedua bit bernilai 0, sebaliknya nilai bit akan diset menjadi 1. Berikut perhitungannya:

```
$a = 10110101
$b = 01101100
-----
$a | $b = 11111101 = 253 (desimal)
```

Pada contoh ketiga, merupakan operasi **xor** ( ^ ). Operasi **xor** akan menghasilkan nilai 1 jika salah satu dari kedua variabel bernilai 1, tapi tidak keduanya. Berikut perhitungannya:

```
$a = 10110101
$b = 01101100
-----
$a ^ $b = 11011001 = 217 (desimal)
```

Contoh keempat, adalah operasi **not** ( ~ ). Operasi ini akan membalikkan nilai bit dari 0 menjadi 1, dan dari 1 menjadi nol. Namun perhitungan operasi **not** di dalam PHP sedikit membingungkan. Jika kita hanya membalikkan seluruh bitnya saja, hasilnya tidak sesuai dengan apa yang dihitung oleh PHP. Berikut adalah perhitungan awal yang saya lakukan:

```
$a = 10110101
-----
~$a = 01001010 = 74 (desimal) ==> salah ???
```

Dari hasil program, dapat dilihat bahwa  $\sim \$a$  bernilai -182, darimanaakah angka ini?

Masalah ini terkait dengan cara PHP menyimpan angka biner menggunakan sistem 32 bit. PHP menyimpan bit dalam perhitungan matematis komputer yang di sebut dengan “*two’s complement*”. Penjelasan tentang “*two’s complement*” dapat anda baca lebih lanjut di [wikipedia](http://en.wikipedia.org/wiki/Two%27s_complement)<sup>1</sup> dan [stackoverflow](http://stackoverflow.com/questions/18754198/confusing-php-bitwise-not-behavior)<sup>2</sup>.

Cara perhitungan singkatnya adalah sebagai berikut:

---

<sup>1</sup>[http://en.wikipedia.org/wiki/Two%27s\\_complement](http://en.wikipedia.org/wiki/Two%27s_complement)

<sup>2</sup><http://stackoverflow.com/questions/18754198/confusing-php-bitwise-not-behavior>

```
$a      = 00000000000000000000000010110101 (32 bit)
-----
~$a      = 111111111111111111111110110101 (32 bit negative)
Flip & -1 = 00000000000000000000000010110101 - 1
~$a      = -182 (desimal) ==> benar
```

Karena PHP menggunakan pemrosesan 32 bit, maka kita harus mengikutkan seluruh bit 0 yang berada di depan angka biner hingga total 32 digit, lalu menegatifkannya. Jika angka paling kiri terdapat angka 1, maka bahwa hasilnya adalah angka negatif, kemudian hasil negatif di flip (dinegatifkan kembali) lalu dikurang 1, sehingga menjadi angka -182 dalam desimal.

Contoh ke 5, adalah operator **shift right**. Operator ini menggeser bit variabel **\$a** ke kanan sebanyak 1 tempat. Berikut proses yang terjadi:

```
$a      = 10110101 = 181
$a >> 1 = 1011010 = 90 (desimal)
```

Operator **shift right** akan menggeser nilai biner variabel **\$a** ke arah kanan, kemudian digit paling kanan dihapus. Setiap penggeseran 1 tempat ke kanan akan membagi 2 nilai asal. Dalam contoh, hasilnya adalah  $180/2 = 90$  (dibulatkan).

Contoh ke 6 adalah operator **shift left**. Operator ini menggeser nilai variabel **\$b** sebanyak 2 digit ke kiri. Berikut proses yang terjadi:

```
$b      = 01101100 = 108
$b << 2 = 0110110000 = 432 (desimal)
```

Ketika hasil pergeseran ke kanan, digit paling kiri akan diisi dengan nilai 0. Setiap penggeseran 1 tempat ke kiri akan mengkali 2 nilai asal.



Anda tidak harus memahami maksud dari operator **bitwise** ini, karena seperti yang telah saya bilang di awal, operator ini sangat jarang digunakan.

## 8.9 Operator Assignment

Operator **assignment** adalah operator untuk memasukkan/menginput sebuah nilai kedalam variabel. PHP memiliki 3 jenis operator assignment, yaitu :

- **Assignment nilai** (assignment by value)
- **Assignment referensi** (assignment by reference)
- **Assignment array**

Saya akan membahas ketiga operator assignment ini.

## Assignment Nilai (assignment by value)

**Assignment nilai**, atau dalam bahasa inggris dikenal dengan istilah **assignment by value** adalah operator yang digunakan untuk menginput suatu nilai ke dalam variabel. Dari awal buku ini secara tidak langsung kita telah sering menggunakan, yakni tanda sama dengan ‘=’, seperti contoh berikut:

```
1 <?php
2   $a = 99;
3   $b = "Belajar PHP";
4 ?>
```

Operasi assignment diproses dari kanan ke kiri. Perintah `$a = 99` berarti saya menginput nilai **99** ke dalam variabel `$a`. Perintah `$b="Belajar PHP"` berarti menginput nilai **“Belajar PHP”** ke dalam variabel `$b`. Bagi pemula yang belajar pemrograman, hal ini memang sedikit membingungkan, karena sehari-hari kita terbiasa memproses (dan membaca) dari kiri ke kanan.

Agar lebih paham, dapatkah anda menghitung nilai akhir variabel `$a` dari kode berikut?

```
1 <?php
2   $a = 3;
3   $a = $a + 5;
4 ?>
```

Nilai akhir dari variabel `$a` adalah 8. Darimana datangnya?

Satu hal yang harus selalu anda ingat adalah: **operasi assignment diproses dari kanan ke kiri**. Pada perintah pertama, variabel `$a` berisi angka 3. Pada perintah kedua, variabel `$a` saat ini (3) ditambah 5, dan hasilnya akan menimpa nilai variabel `$a`. Dengan demikian, variabel `$a` bernilai 8.

Bagaimana dengan contoh ini?

```
1 <?php
2   $a = 10;
3   $b = 3;
4   $a = $a - 5;
5   $a = $a + $b - 4;
6   echo $a;
7 ?>
```

Tanpa menjalankan kode tersebut, dapatkah anda mencari hasil akhir variabel `$a`? Konsep seperti ini sangat amat penting untuk dipahami, karena akan selalu kita gunakan sepanjang pembuatan program. Pada baris ke 3, variabel `$a` bernilai 5 ( $10-5$ ). Pada baris ke 4, variabel `$a` akan bernilai 4 ( $5+3-4$ ).

## Assignment Referensi (assignment by reference)

Pembahasan tentang Assignment Referensi (*assignment by reference*) memang agak susah dipelajari. Ketika pertama kali belajar pemrograman, saya juga butuh waktu cukup lama untuk bisa memahaminya. Jika anda bingung dengan penjelasan ini, silahkan diabaikan untuk sementara.

Berdasarkan pengalaman saya, assignment referensi relatif jarang digunakan, tapi konsepnya sebaiknya tetap diketahui.

Untuk memahami **assignment referensi** atau dalam bahasa inggris: *assignment by reference*, kita harus membahas cara sebuah nilai di simpan ke dalam variabel.

Variabel pada dasarnya adalah sebuah ‘ruang’ di dalam memory komputer. Ruang ini bisa diisi dengan nilai apa saja. Sebagai contoh, perintah `$a=99` berarti saya mengisi nilai 99 ke dalam alamat memory yang dialokasikan untuk variabel `$a`. Alamat memory ini diproses secara internal oleh PHP (kita tidak perlu tahu sebenarnya dimana alamat ini).

Ketika saya menambah variabel baru, misalnya `$b="Belajar PHP"`, sebuah alamat memory kembali dialokasikan kepada variabel `$b`. Dengan demikian, saat ini saya memiliki 2 alamat memory untuk menyimpan kedua variabel.

Berikutnya saya membuat variabel ketiga: `$c`. Namun saya tidak akan mengisi nilai apa-apa ke dalam variabel ini. Melainkan ‘menyamakan’ alamat memory dari variabel `$c` dengan alamat memory variabel `$b`. Dengan demikian, variabel `$c` dan `$b` menggunakan alamat memory yang sama. Kedua variabel ini akan sama-sama memiliki nilai “Belajar PHP”. Inilah yang disebut dengan **assignment by reference**.

Assignment by reference di dalam PHP menggunakan operator “`=&`”. Mari kita masuk ke dalam contoh kode program:

```
1 <?php
2   $a = 99;
3   $b = "Belajar PHP";
4   $c = & $b;
5
6   echo "$a , $b"; // 99 , Belajar PHP
7   echo "<br />";
8   echo "$a , $c"; // 99 , Belajar PHP
9   echo "<br />";
10  echo "$a , $b , $c"; // 99 , Belajar PHP , Belajar PHP
11 ?>
```

Kode diatas menggambarkan konsep yang kita bahas sebelumnya, dimana variabel `$c` menggunakan alamat memory yang sama dengan variabel `$b`. Kedua variabel ini berisi nilai yang sama: “Belajar PHP”.

*Assignment by reference* juga membawa konsekuensi lain. Ketika dua buah variabel (atau lebih) menggunakan alamat memory yang sama, jika kita mengubah nilai satu variabel, nilai variabel yang lain juga akan ikut berubah. Berikut contohnya:

```

1 <?php
2   $a = 99;
3   $b = "Belajar PHP";
4   $c =& $b;
5
6   echo "$a , $b , $c"; // 99 , Belajar PHP , Belajar PHP
7   echo "<br>";
8
9   $c = "Operator Assignment";
10  echo "$a , $b , $c"; // 99 , Operator Assignment , Operator Assignment
11 ?>

```

Seperti yang terlihat, ketika saya mengubah isi variabel \$c, nilai dari variabel \$b juga ikut berubah. Ini karena keduanya menggunakan ‘tempat’ yang sama di memory.

Konsep mengenai **assignment by reference** cukup penting untuk dipahami, dan tidak hanya berguna di dalam PHP. Hampir setiap bahasa pemrograman memiliki konsep yang sama. Pemograman PHP lanjutan seperti object juga banyak menggunakan *assignment by reference*.

## Assignment array

**Assignment array** adalah operasi pemberian nilai ke dalam array. Operator yang digunakan adalah tanda panah ( => ). Berikut contohnya:

```

1 <?php
2   $siswa = array(
3     0 => "Andri",
4     1 => "Joko",
5     2 => "Sukma",
6     3 => "Rina",
7     4 => "Sari"
8   );
9
10  echo $siswa[2]; // Sukma
11 ?>

```

Operator ini juga telah kita pelajari ketika membahas tentang tipe data array dalam bab sebelumnya.

## Operator Gabungan Assignment

**Operator gabungan assignment** adalah cara penulisan singkat operator assignment dengan operator lainnya. Dalam PHP, operator gabungan ini bisa dilakukan antara *operator assignment* dengan *operator aritmatika*, *string*, dan *bitwise*.

Sebagai contoh, operasi `$a = $a + 5` dapat ditulis menjadi `$a += 5`, sedangkan operasi `$b = $b."aku"` bisa disingkat menjadi `$b .= "aku"`.

Tabel berikut merangkum operator gabungan assignment di dalam PHP:

Nama Operator	Operator	Contoh	Hasil
plus-equals	<code>+=</code>	<code>\$b += 10</code>	<code>\$b = \$b + 10</code>
minus-equals	<code>-=</code>	<code>\$b -= 10</code>	<code>\$b = \$b - 10</code>
divide-equals	<code>/=</code>	<code>\$b /= 10</code>	<code>\$b = \$b / 10</code>
multiply-equals	<code>*=</code>	<code>\$b *= 10</code>	<code>\$b = \$b * 10</code>
modulus-equals	<code>%=</code>	<code>\$b %= 10</code>	<code>\$b = \$b % 10</code>
bitwise-xor-equals	<code>^=</code>	<code>\$b ^= 10</code>	<code>\$b = \$b ^ 10</code>
bitwise-and-equals	<code>&amp;=</code>	<code>\$b &amp;= 10</code>	<code>\$b = \$b &amp; 10</code>
bitwise-or-equals	<code> =</code>	<code>\$b  = 10</code>	<code>\$b = \$b   10</code>
concatenate-equals	<code>.=</code>	<code>\$b .= "duniaIlkom"</code>	<code>\$b = \$b."duniaIlkom"</code>

Berikut contoh penggunaannya:

```

1 <?php
2   $a = 10;
3   $a += 5;
4   echo $a; // 15
5   echo "<br>";
6
7   $a = 55;
8   $a /= 5;
9   echo $a; // 11
10  echo "<br>";
11
12 $a = "Belajar";
13 $a .= " PHP";
14 echo $a; // Belajar PHP
15 echo "<br>";
16 ?>

```

Penulisan operator secara singkat seperti ini cukup umum digunakan dan juga didukung oleh berbagai bahasa pemrograman lain.

## 8.10 Operator Error Control

PHP memiliki 1 jenis **operator error control**, yakni menggunakan tanda '@'. Operator ini digunakan untuk mengabaikan pesan kesalahan pada sebuah perintah.

Sebagai contoh, perhatikan kode berikut:

```
1 <?php
2     $a = 13;
3     $b = 0;
4     $hasil = $a/$b; // Warning: Division by zero
5     echo "Halaman ini terdapat error";
6 ?>
```

Kode diatas akan memberikan error **Warning: Division by zero**, karena saya melakukan operasi pembagian dengan angka 0 (yang hasilnya memang tidak terdefinisi).

Jika kita menambahkan tanda ‘@’ diawal operasi tersebut, PHP tidak akan menampilkan error:

```
1 <?php
2     $a = 13;
3     $b = 0;
4     @$hasil = $a/$b;
5     echo "Halaman ini terdapat error, tapi tidak ditampilkan.";
6 ?>
```

Kali ini, pesan error tidak ditampilkan ke dalam browser, walaupun secara sistem tetap terdapat error.

Penggunaan operator error control seperti diatas hanya sebagai contoh. Jika tujuannya adalah untuk tidak menampilkan error sama sekali (untuk keamanan), sebaiknya diset melalui file konfigurasi php: **php.ini**.

## 8.11 Urutan Operator dalam PHP

Seluruh operator yang telah kita pelajari pada bab ini memiliki aturan prioritas tertentu. Sebagai contoh, perhatikan kode berikut:

```
1 <?php
2     $a = true and false;
3     var_dump($a); // bool(true)
4 ?>
```

Variabel **\$a** yang digunakan untuk menampung hasil **true and false** menghasilkan **true!** Padahal kita telah pelajari bahwa operasi logika “**and**” hanya akan bernilai **true** jika kedua operand adalah **true**. Bagaimana ini bisa terjadi?

Kasus diatas disebabkan operator assignment ‘=’ memiliki prioritas lebih tinggi daripada operator logika ‘**and**’. Apa yang sebenarnya diproses oleh PHP adalah ‘**(\$a = true) and false**’’. Sehingga hasilnya adalah **true**.

Kode diatas seharusnya diubah menjadi seperti berikut:

```

1 <?php
2 $a = (true and false);
3 var_dump($a); // bool(false)
4 ?>

```

Kesalahan seperti ini sangat sangat sulit di deteksi (karena PHP tidak mengeluarkan error). Oleh sebab itu kita perlu mengetahui urutan prioritas operator. Tabel berikut merangkum urutan prioritas operator di dalam PHP:

Urutan Proses	Operator	Keterangan
tidak-diurutkan	clone new	pembuatan object
kiri	[ array()	pembuatan array
kanan	++ - ~ (int) (float) (string) (array) (object) (bool) @	increment, decrement, tipe casting, error control
tidak-diurutkan	instanceof	cek class
kanan	!	logika not / negasi
kiri	* / %	operator aritmatika
kiri	+ - .	operator aritmatika dan penyambungan string
kiri	<< >>	operator bitwise
tidak-diurutkan	< <= > >=	operator perbandingan
tidak-diurutkan	== != === !== <>	operator perbandingan
kiri	&	operator bitwise dan referensi
kiri	^	operator bitwise
kiri		operator bitwise
kiri	&&	operator logika
kiri		operator logika
kiri	? :	operator kondisi
kanan	= += -= *= /= .= %= &=  = ^= <<= >>= ==	operator assignment
kiri	and	operator logika
kiri	xor	operator logika
kiri	or	operator logika
kiri	,	berbagai fungsi

Dari tabel diatas, urutan prioritas dimulai dari baris paling atas ke bawah. Sebagai contoh, operator **new** (untuk membuat object) lebih kuat dibandingkan operator **aritmatika**.

Jika dalam 1 operasi terdapat beberapa operator dengan urutan prioritas yang sama (karena pada dalam tabel diatas berada dalam 1 baris), urutan prioritas ditentukan kolom **urutan proses**

(dalam manual PHP disebut dengan **associativity**).

Sebagai contoh, bagaimana urutan proses operasi  $6-9+10-5$ ? Operasi ini melibatkan operator pengurangan aritmatika (-). Di dalam kolom ‘arah proses’, operator ini bernilai ‘kiri’, sehingga operasi diatas akan diproses dari kiri ke kanan, yakni sebagai  $((6-9)+10)-5$ .

Contoh lain, bagaimana dengan operasi  $\$a = \$b = \$c = \$d$ ? Apakah variabel  $\$a$  akan diisi dengan nilai dari variabel  $\$d$ ? atau sebaliknya?

Kembali ke tabel diatas, operator assignment memiliki nilai arah kolom ‘kanan’, sehingga akan dikerjakan dari kanan ke kiri sebagai berikut  $\$a = (\$b = (\$c = \$d))$ .

Apabila kolom arah proses berisi “tidak-diurutkan”, berarti operator itu tidak bisa digunakan secara berdampingan. Misalkan  $4 < 6 > 2$  tidak dapat diproses oleh PHP, namun  $1 <= 1 == 1$  bisa diproses karena operator == memiliki urutan prioritas lebih rendah daripada <=.

Walaupun kita bisa membuat urutan operasi berjalan dengan seharusnya, sangat disarankan menggunakan tanda kurung sebagai penegasan urutan proses. Penggunaan tanda kurung juga memudahkan pembacaan program.

Contohnya operasi  $\$a$  and  $\$b$  or  $\$c$ , akan lebih mudah dimengerti ketika ditulis menjadi  $(\$a$  and  $\$b)$  or  $\$c$ , walaupun sebenarnya operator and memiliki urutan prioritas lebih tinggi daripada or.

## 8.12 Type Juggling

Dalam bab sebelumnya (tentang tipe data), kita telah membahas cara mengubah sebuah tipe data menjadi tipe data lain atau dikenal dengan istilah *Type Casting*. Namun dengan menggunakan operator, sebuah tipe data bisa langsung berubah menjadi tipe data lain secara otomatis tanpa kita instruksikan. Ini dikenal dengan istilah **Type Juggling**.

Perhatikan kode program berikut:

```
<?php
$a=12;
$b="9 kucing";
echo $a+$b; // 21
?>
```

Dalam kode diatas, saya mendefenisikan variabel  $\$a$  sebagai integer (angka) dan variabel  $\$b$  sebagai string. Operasi penambahan seharusnya membutuhkan 2 nilai angka, namun seperti yang terlihat, PHP dengan senang hati akan menjalankan perintah tersebut tanpa mengeluarkan error.  $12+9$  kucing” akan menghasilkan nilai 21.

PHP akan mencoba “menebak” dan mengubah tipe data agar sesuai dengan peruntukannya. Misalkan operator aritmatika seharusnya membutuhkan 2 buah inputan (operand) yang bertipe angka (baik berupa integer maupun float). Apabila salah satu atau kedua operand itu bukan bertipe angka, PHP akan mengkonversinya menjadi angka.

Sebagai contoh lain, perhatikan kode berikut:

```
<?php
$a=12;
$b="9 kucing";
echo $a AND $b;
?>
```

Nilai dari variabel \$a dan \$b masih sama dengan contoh pertama, namun kali ini saya menggunakan operator logika AND. Jika anda menjalankan program diatas, di browser akan tampil angka 1. Dari manakah angka 1 ini berasal?

Operator AND membutuhkan 2 inputan bertipe boolean, yakni nilai TRUE atau FALSE. Karena saya menggunakan tipe integer “12” dan type string “9 kucing”, kedua operand ini akan dikonversi menjadi TRUE. TRUE AND TRUE adalah TRUE, sehingga hasil \$a AND \$b pada contoh program diatas adalah TRUE.

Proses *type juggling* belum selesai. Perintah echo membutuhkan inputan tipe data string, bukan tipe boolean. Sehingga PHP akan mengkonversi boolean TRUE menjadi string “1”.

Sekilas fitur type juggling terasa memudahkan, tapi ini adalah salah satu sumber masalah yang sangat sulit di deteksi. PHP tidak akan mengeluarkan error apapun jika kita membuat operasi yang tidak cocok seperti ini.

Untuk menghindarinya, sedapat mungkin kita tidak melakukan operasi beda tipe data. Jika memang harus, gunakan perintah *type casting* seperti (**int**), (**float**) atau (**string**) untuk memastikan tipe data sudah sesuai dengan keinginan.

---

Dalam bab ini kita telah membahas operator-operator dalam PHP. Berikutnya saya akan lanjut ke struktur pemrograman PHP seperti IF ELSE dan perulangan (looping).

# 9. Struktur Control Pemrograman PHP

Struktur control pemrograman PHP berkaitan dengan bagaimana sebuah kode program berjalan. Dalam bab ini kita akan mempelajari struktur kontrol seperti if, else if, dan switch. Setelah itu juga akan dibahas tentang perulangan for, while, do while dan foreach.

## 9.1 Struktur Logika IF

Struktur logika IF digunakan untuk mengatur kapan sebuah kode program akan dijalankan. Sebagai contoh, kita bisa menampilkan seluruh tabel mahasiswa hanya untuk user bernama “admin”. Jika user bukan “admin”, tabel tidak akan ditampilkan. Berikut contoh penulisannya:

```
<?php
if ($user=="admin") {
    // tampilkan tabel mahasiswa
}
?>
```

Dalam contoh diatas, saya membuat operasi perbandingan antara isi variabel `$user` dengan string “admin”. Jika cocok, barulah tabel ditampilkan.

Penulisan struktur IF setidaknya membutuhkan 2 bagian, yakni **kondisi** (atau disebut juga sebagai *expression*) dan **statement**:

```
<?php
if (kondisi) {
    statement;
}
?>
```

**Statement** bisa terdiri dari satu, dua atau ratusan baris perintah PHP. Baris ini hanya akan dijalankan selama **kondisi** bernilai TRUE. Jika kondisi menghasilkan nilai FALSE, statement tidak akan dijalankan. Tanda kurung kurawal ‘{’ dan ‘}’ menandakan blok kode program yang akan diproses ketika kondisi TRUE.

**Kondisi** yang ingin diproses haruslah ‘sesuatu’ yang menghasilkan boolean TRUE atau FALSE. Perhatikan kode berikut:

```

1 <?php
2   $user="guest";
3
4   if ($user=="admin") {
5     echo "Selamat datang Admin!";
6   }
7 ?>

```

Dapatkankah anda menebak apa hasil yang akan tampil? Yup, tidak ada tampilan sama sekali. Ini karena kondisi `$user=="user"` akan menghasilkan FALSE. Variabel `$user` saat ini berisi string `'guest'`, bukan `"admin"`. Teks `"Selamat datang Admin!"` baru akan tampil ketika saya mengubah kode diatas menjadi:

```

1 <?php
2   $user="admin";
3
4   if ($user=="admin") {
5     echo "Selamat datang Admin!"; // Selamat datang Admin!
6   }
7 ?>

```

Pada umumnya, kita hanya menggunakan operasi perbandingan untuk menentukan kondisi, seperti `if ($password=="1234")`, atau `if ($user<>"admin")`.

Karena di dalam PHP ada efek **type juggling** , saya juga bisa menulis kondisi seperti berikut:

```

1 <?php
2   if (9) {
3     echo "Selamat datang Admin!";
4   }
5 ?>

```

Perintah `echo` akan dijalankan, karena integer 9 secara otomatis dikonversi PHP menjadi boolean TRUE.

Khusus untuk statement hanya terdiri dari 1 baris, tanda kurung kurawal boleh tidak ditulis, seperti berikut:

```

1 <?php
2   $user="admin";
3
4   if ($user=="admin")
5     echo "Selamat datang Admin!"; // Selamat datang Admin!
6 ?>

```

Penulisan seperti ini memang lebih praktis, tapi tidak disarankan. Jika kita menambahkan statement baru dan lupa menulis kurung kurawal, kode program akan error atau tampil bukan seperti yang diinginkan.

## Multiple IF

Kita juga bisa menulis beberapa kondisi IF untuk membuat percabangan kode program, seperti berikut:

```
1 <?php
2     if (kondisi1)
3     {
4         statement1;
5         statement2;
6     }
7     if (kondisi2)
8     {
9         statement3;
10        statement4;
11    }
12 ?>
```

Untuk kasus yang lebih spesifik, kita juga bisa membuat struktur IF di dalam IF, ini dikenal sebagai **nested IF** atau **IF bersarang**. Berikut contohnya:

```
1 <?php
2     if (kondisi1)
3     {
4         statement1;
5         if (kondisi2)
6         {
7             statement1;
8         }
9     }
10 ?>
```

PHP tidak membatasi berapa banyak kondisi IF di dalam IF (nested IF). Namun perlu diperhatikan jika kita membuat struktur IF yang kompleks, tanda kurung kurawal ini harus dikelola dengan benar. Kesalahan penutupan kurung kurawal akan menghasilkan error atau kode program tidak berjalan seperti yang diharapkan.

Whitespace juga bukan suatu keharusan. Ketiga gaya penulisan ini identik dan bisa dipakai:

```
1 <?php
2     if (kondisi1)
3     {
4         statement1;
5         statement2;
6     }
7
8     if (kondisi1) {
9         statement1;
10        statement2;
11    }
12
13     if (kondisi1){ statement1; statement2;}
14 ?>
```

## Alternatif Penulisan Struktur IF

Selain menggunakan kurung kurawal, PHP juga menyediakan cara penulisan alternatif untuk mengawali blok kondisi if, yakni dengan tanda titik dua (:) dan diakhiri dengan **endif**. Berikut format dasar penulisannya:

```
1 <?php
2     if (kondisi) :
3         statement1;
4         statement1;
5     endif
6 ?>
```

Perbedaan cara penulisan ini ada di tanda titik dua (:) setelah **kondisi**, dan *keyword* **endif** di akhir statement.

Cara penulisan seperti ini cocok jika blok kondisi if cukup rumit dan di dalamnya juga terdapat struktur program lain seperti perulangan. Kode program aplikasi **WordPress** juga menggunakan cara penulisan seperti ini.

## 9.2 Struktur Logika ELSE dan ELSE IF

Pada dasarnya, struktur logika **ELSE** dan **ELSE IF** hanya perpanjangan dari struktur IF. Bagian **ELSE** dijalankan ketika kondisi IF tidak sesuai atau FALSE. Berikut contohnya:

```

1 <?php
2   $user="guest";
3
4   if ($user=="admin"){
5     echo "Selamat datang Admin!";
6   }
7   else {
8     echo "Maaf, anda bukan Admin";
9   }
10 ?>

```

Kode program diatas akan menampilkan teks “*Maaf, anda bukan Admin*”. Ini karena variabel **\$user** tidak berisi “admin”. Sehingga kondisi **\$nama==”admin”** akan menghasilkan FALSE.

## Multiple ELSE dan ELSE IF

Untuk kode program yang cukup kompleks, kita bisa mengkombinasikan beberapa struktur ELSE IF, seperti contoh berikut:

```

1 <?php
2   $user="guest";
3
4   if ($user=="admin"){
5     echo "Selamat datang Admin!";
6   }
7   else if ($user=="user"){
8     echo "Selamat datang User";
9   }
10  else if ($user=="guest"){
11    echo "Selamat datang Tamu";
12  }
13  else {
14    echo "Maaf, saya tidak kenal anda";
15  }
16 ?>

```

Silahkan anda pelajari sebentar kode program diatas, kemudian coba ubah variabel **\$user** menjadi string “admin”, “user”, atau string sembarang. Selain itu, perhatikan juga saya menulis “else if” (dengan spasi). Penulisan ini juga bisa disambung menjadi “elseif”.

## Alternatif Penulisan Struktur IF ELSE

Sama seperti struktur IF, kita juga bisa menggunakan alternatif penulisan selain dengan tanda kurung kurawal:

```

1 <?php
2   if (expression) :
3     statement1;
4     statement2;
5   elseif (expression):
6     statement3;
7   else
8     statement4;
9   endif
10  ?>

```

Khusus untuk cara penulisan ini, kita tidak bisa memisahkan penulisan ELSE-IF menjadi “**else if**”, tetapi harus ditulis menyatu menjadi “**elseif**”.

## 9.3 Latihan IF, ELSE, dan ELSE IF

Struktur IF, ELSE dan ELSE IF merupakan salah satu materi terpenting dalam pemrograman. Dengan struktur ini kita bisa membuat alur percabangan program tergantung kondisi dan situasi yang dihadapi.

Agar pemahaman anda semakin mantap tentang struktur IF, ELSE dan ELSE IF (terutama bagi yang belum pernah belajar bahasa pemrograman komputer), saya telah membuat 5 latihan kode program. Latihan ini juga akan menggunakan konsep variabel, tipe data dan operator yang telah kita pelajari sebelumnya.

Saya sangat sarankan anda mencoba terlebih dahulu (walaupun salah), agar bisa memahami dimana letak salahnya dan apa yang mesti diperbaiki. Solusi yang saya berikan juga tidak mutlak. Kode program yang anda buat bisa jadi lebih baik, selama sesuai dengan kondisi yang diminta.



Kelima latihan yang akan kita coba disini sebenarnya adalah latihan algoritma, bukan latihan kode PHP. Algoritma adalah bidang ilmu pemrograman yang fokus kepada alur logika pemecahan masalah.

Membuat lagoritma untuk mencari solusi adalah kerjaan sehari-hari programmer. Skill ini akan semakin meningkat dengan banyaknya anda latihan kode program.

### Latihan 1: Genap atau Ganjil

Buatlah kode program yang bisa membedakan angka genap dengan angka ganjil. Misalkan variabel `$a` berisi 50, akan tampil “*50 adalah angka genap*”. Jika `$a` berisi 7, maka akan tampil “*7 adalah angka ganjil*”.

Tips: gunakan operator modulus (%) untuk membedakan bilangan genap dan ganjil.

### Jawaban Latihan 1

Untuk membedakan antara angka genap dan ganjil, kita bisa menggunakan fakta bahwa bilangan genap adalah bilangan yang habis dibagi 2. Kondisi ini bisa di cek dengan operator **mod** (%). Berikut kode program yang saya gunakan:

```

1 <?php
2   $a=10;
3   if ($a % 2 == 0){
4     echo "$a adalah angka genap";
5   }
6   else {
7     echo "$a adalah angka ganjil";
8   }
9 ?>

```

Jika variabel `$a` berisi angka genap (kelipatan 2), maka kondisi `if ($a % 2 == 0)` menghasilkan TRUE. Dengan demikian, yang dijalankan adalah `echo "$a adalah angka genap"`. Selain genap, tentu angka ganjil. Saya tinggal menambahkan perintah **ELSE** yang berisi `echo "$a adalah angka ganjil"`.

## Latihan 2: Lebih Besar, Lebih Kecil atau Sama Dengan

Buatlah sebuah kode program yang terdiri dari 2 variabel: `$a` dan `$b`. Kedua variabel ini berisi angka integer atau float. Tampilan akhir adalah salah satu kondisi berikut:

- `$a` lebih kecil dari `$b`
- `$a` lebih besar dari `$b`
- `$a` sama dengan `$b`

Sebagai contoh, jika saya memberikan nilai `$a=6` dan `$b=10`, tampilan akhir adalah: “*6 lebih kecil dari 10*”. Jika `$a=5` dan `$b=5`, tampilan akhir adalah: “*5 sama dengan 5*”.

### Jawaban Latihan 2

Kita harus membuat kondisi untuk 3 kemungkinan: apakah `$a < $b`, atau `$a > $b`, atau `$a == $b`. Berikut kode programnya:

```

1 <?php
2   $a=8;
3   $b=10;
4   if ($a < $b){
5     echo "$a lebih kecil dari $b";
6   }
7   else if ($a > $b){
8     echo "$a lebih besar dari $b";
9   }
10  else {
11    echo "$a sama dengan $b";
12  }
13 ?>

```

Tidak ada hal yang baru disini. Perhatikan bahwa saya tidak membuat kondisi `$a == $b`. Kenapa? Karena jika `$a` tidak lebih besar dari `$b`, dan `$a` juga tidak lebih kecil dari `$b`, satu-satunya kemungkinan lain adalah `$a == $b`.

## Latihan 3: Username dan Password

Buatlah kode program yang akan memeriksa apakah `$username` berisi string "admin" dan `$password` berisi string "qwerty". Jika benar, tampilkan pesan: "*Username dan Password sesuai, hak akses diberikan*". Jika salah, tampilkan pesan: "*Username atau Password tidak sesuai!*".

Tips: `username` dan `password` harus sesuai, jika salah satu tidak cocok, tampilkan pesan: "*Username atau Password tidak sesuai!*".

### Jawaban Latihan 3

Kata kunci disini adalah `$username` berisi "admin" dan `$password` berisi "qwerty". Jika salah satu saja salah (selain "admin" dan "qwerty"), tampilkan "*Username atau password tidak sesuai!*". Berikut kode programnya:

```

1 <?php
2     $username="admin";
3     $password="qwerty";
4
5     if ($username=="admin" AND $password=="qwerty"){
6         echo "Username dan password sesuai, hak akses diberikan";
7     }
8     else {
9         echo "Username atau password tidak sesuai!";
10    }
11 ?>

```

Kali ini kita harus memeriksa kedua kondisi (keduanya harus cocok). Solusinya dengan menggabungkan 2 kondisi menggunakan operator AND.

## Latihan 4: Username dan Password Lanjutan

Latihan kali ini mirip seperti latihan 3, tapi saya ingin agar terdapat 4 kondisi:

- Username sesuai, password sesuai: "*Username dan password sesuai, hak akses diberikan*".
- Username sesuai, password tidak sesuai: "*Username sesuai, password tidak sesuai!*".
- Username tidak sesuai, password sesuai: "*Username tidak sesuai, password tidak sesuai!*".
- Username tidak sesuai, password tidak sesuai: "*Nama dan Password tidak sesuai!*".

### Jawaban Latihan 4

Terdapat 2 alternatif yang bisa digunakan. Yang pertama dan yang paling gampang adalah membuat 4 struktur IF dengan memeriksa satu-satu kondisi yang ada. Berikut kode programnya:

```

1  <?php
2      $username="admin";
3      $password="qwerty";
4
5      if ($username=="admin" AND $password=="qwerty"){
6          echo "Nama dan password sesuai, hak akses diberikan..";
7      }
8      else
9          if ($username=="admin" AND $password!="qwerty"){
10             echo "Nama sesuai, password tidak sesuai!";
11         }
12     else
13         if ($username!="admin" AND $password=="qwerty"){
14             echo "Nama tidak sesuai, password sesuai!";
15         }
16     else {
17         echo "Nama dan password tidak sesuai!";
18     }
19 ?>

```

Cara kedua yang mungkin sedikit lebih rumit adalah menggunakan IF bersarang, yakni IF di dalam IF:

```

1  <?php
2      $username="admin";
3      $password="qwerty";
4
5      if ($username=="admin"){
6          if ($password=="qwerty") {
7              echo "Nama dan password sesuai, hak akses diberikan..";
8          }
9          else {
10              echo "Nama sesuai, password tidak sesuai!";
11          }
12      }
13      else if ($password=="qwerty") {
14          echo "Nama tidak sesuai, password sesuai!";
15      }
16      else {
17          echo "Nama dan password tidak sesuai!";
18      }
19 ?>

```

Cara kedua ini perlu pemahaman tersendiri, karena ada IF di dalam IF. Silahkan anda pelajari sebentar, dan mudah-mudahan anda bisa memahami apa maksud kode diatas. Walaupun alternatif pertama lebih mudah, ada kalanya kita terpaksa menggunakan struktur *nested IF* seperti ini.

## Latihan 5: Nama Hari

Buatlah kode program yang menampilkan string nama hari: “Hari Senin”, “Hari Selasa”, sampai dengan “Hari Minggu”, berdasarkan variabel `$hari` yang bisa diisi dengan angka integer 1-7. Sebagai contoh, jika `$hari == 1`, tampilkan “Hari Senin”. Jika `$hari == 5`, tampilkan “Hari Jum’at”. Jika angka diluar 1-7 tampilkan: “Nama hari cuma ada 7!”.

### Jawaban Latihan 5

Kode program yang dibutuhkan cukup sederhana, yakni 7 kondisi if else. Berikut kode programnya:

```

1  <?php
2      $hari=9;
3
4      if ($hari==1){ echo "Hari Senin"; }
5      else
6          if ($hari==2){ echo "Hari Selasa"; }
7      else
8          if ($hari==3){ echo "Hari Rabu"; }
9      else
10         if ($hari==4){ echo "Hari Kamis"; }
11     else
12         if ($hari==5){ echo "Hari Jum'at"; }
13     else
14         if ($hari==6){ echo "Hari Sabtu"; }
15     else
16         if ($hari==7){ echo "Hari Minggu"; }
17     else {
18         echo "Nama hari cuma ada 7!";
19     }
20 ?>

```

Tidak ada yang baru selain mengulang struktur IF sampai 7 kali. Untuk kondisi sederhana dan berulang seperti ini, terdapat alternatif selain ELSE IF, yakni struktur SWITCH. Inilah yang akan kita bahas berikutnya.

## 9.4 Struktur Logika SWITCH

Struktur logika **switch** adalah sebuah struktur percabangan yang akan memeriksa satu variabel, lalu menjalankan perintah sesuai dengan kondisi. Struktur switch ini mirip dengan struktur IF yang ditulis berulang.

Sebagai contoh, kode program untuk latihan 5 sebelumnya bisa ditulis menjadi:

```

1  <?php
2      $hari=4;;
3      switch ($hari)
4      {
5          case 1 :
6              echo "Hari Senin";
7              break;
8          case 2 :
9              echo "Hari Selasa";
10             break;
11          case 3 :
12              echo "Hari Rabu";
13              break;
14          case 4 :
15              echo "Hari Kamis";
16              break;
17          case 5 :
18              echo "Hari Jum'at";
19              break;
20          case 6 :
21              echo "Hari Sabtu";
22              break;
23          case 7 :
24              echo "Hari Minggu";
25              break;
26      default :
27          echo "Nama hari cuma ada 7!";
28          break;
29      }
30  ?>

```

Seperti yang terlihat, struktur **switch** terdiri dari beberapa bagian, berikut format dasar penulisan **switch** dalam PHP:

```

switch ($var)
{
    case value1:
        statement1;
        break;
    case value2:
        statement2;
        break;
}

```

Setelah perintah **switch**, kita menulis variabel yang akan diperiksa. Variabel ini ditempatkan dalam tanda kurung. Seluruh *block switch* berada di antara kurung kurawal '{' dan '}'.

Tiap kondisi yang mungkin terjadi ditulis pada perintah **case**, lalu diikuti dengan kondisi yang ingin diuji. Jika sesuai, baris statement akan dijalankan. Alur program untuk switch akan dieksekusi dari baris pertama sampai terakhir. Kata kunci **break** digunakan untuk keluar dari **switch**.

Apa sebenarnya fungsi dari perintah **break**? Mari kita coba dengan kode program berikut:

```
1 <?php
2     $a=1;
3     switch ($a)
4     {
5         case 0:
6             echo "Angka Nol ";
7         case 1 :
8             echo "Angka Satu ";
9         case 2 :
10            echo "Angka Dua ";
11        case 3 :
12            echo "Angka Tiga ";
13    }
14 ?>
```

Program diatas memeriksa nilai variabel **\$a** dan memberikan output tergantung kondisi **\$a**. Jika dilihat sekilas, keluaran program seharusnya: “Angka Satu”. Akan tetapi, jika anda menjalankan program diatas, hasilnya adalah:

Angka Satu Angka Dua Angka Tiga

Apa yang terjadi? ini terkait dengan bagaimana PHP menjalankan proses **switch**.

Ketika program dijalankan, PHP pertama kali akan memeriksa case 0. Apakah **\$a** sama dengan 0? tidak, PHP akan lanjut ke case berikutnya. Apakah **\$a** sama dengan 1? iya, PHP akan menjalankan echo “Angka Satu” **beserta seluruh statement pada case-case setelahnya**. Hal ini mungkin terkesan aneh, tapi inilah alur kerja **switch PHP** dan berbagai bahasa pemrograman lain seperti **JavaScript**.

Jadi, bagaimana supaya hasilnya hanya “Angka Satu” saja? Tambahkan perintah **break** pada setiap **case**:

```

1 <?php
2   $a=1;
3   switch ($a)
4   {
5     case 0:
6       echo "Angka Nol ";
7       break;
8     case 1 :
9       echo "Angka Satu ";
10    break;
11   case 2 :
12     echo "Angka Dua ";
13     break;
14   case 3 :
15     echo "Angka Tiga ";
16     break;
17   }
18 ?>

```

Selain **break**, juga terdapat perintah **default** pada *block switch*. Perintah ini berfungsi seperti **ELSE** di dalam struktur **IF**, yang akan dijalankan ketika tidak ada kondisi **case** yang cocok. Perintah **default** diletakkan di baris paling akhir.

Berikut penambahan perintah **default** dari contoh sebelumnya:

```

1 <?php
2   $a=9;
3   switch ($a)
4   {
5     case 0 : echo "Angka Nol "; break;
6     case 1 : echo "Angka Satu "; break;
7     case 2 : echo "Angka Dua "; break;
8     case 3 : echo "Angka Tiga "; break;
9     default : echo "Angka diluar jangkauan"; break;
10   }
11 ?>

```

Kali ini jika variabel **\$a** bukan 0, 1, 2 atau 3, hasilnya adalah: "Angka diluar jangkauan".

Selain itu, perhatikan bentuk penulisan struktur **switch**. Saya membuatnya memanjang untuk memperlihatkan bahwa whitespace tidak berpengaruh di dalam PHP.

Bagaimana jika saya ingin menyatukan beberapa **case**? Misalkan jika **\$a** berisi 0, 1, 2 atau 3 jalankan: "Angka berada di dalam range 0-3". Jika **\$a** berisi nilai 4, 5, 6, dan 7 jalankan: "Angka berada di dalam range 4-7". Terdapat 2 cara, cara pertama adalah membuat satu-satu **case** tersebut:

```

1 <?php
2   $a=6;
3   switch ($a)
4   {
5     case 0 : echo "Angka berada di dalam range 0-3"; break;
6     case 1 : echo "Angka berada di dalam range 0-3"; break;
7     case 2 : echo "Angka berada di dalam range 0-3"; break;
8     case 3 : echo "Angka berada di dalam range 0-3"; break;
9     case 4 : echo "Angka berada di dalam range 4-7"; break;
10    case 5 : echo "Angka berada di dalam range 4-7"; break;
11    case 6 : echo "Angka berada di dalam range 4-7"; break;
12    case 7 : echo "Angka berada di dalam range 4-7"; break;
13    default : echo "Angka diluar jangkauan";           break;
14  }
15 ?>

```

Tidak ada yang salah dalam kode program tersebut, tetapi saya bisa memanfaatkan fitur break untuk membuat kode program seperti ini:

```

1 <?php
2   $a=3;
3   switch ($a)
4   {
5     case 0 :
6     case 1 :
7     case 2 :
8     case 3 : echo "Angka berada di dalam range 0-3"; break;
9     case 4 :
10    case 5 :
11    case 6 :
12    case 7 : echo "Angka berada di dalam range 4-7"; break;
13    default : echo "Angka diluar jangkauan";           break;
14  }
15 ?>

```

Ketika variabel \$a berisi nilai 0, 1 atau 2, tidak ada perintah atau statement yang akan dijalankan. Tapi, apakah anda masih ingat bagaimana struktur switch bekerja jika tanpa perintah break? Case tersebut akan ‘turun’ dan menjalankan seluruh statement dibawahnya sampai ditemukan perintah **break**. Oleh karena itu, hasilnya adalah “Angka berada di dalam range 0-3”.

Sebagai contoh lain, perhatikan kode program berikut:

```

1  <?php
2      $user="sari";
3      switch ($user)
4      {
5          case "joni" :
6          case "andi" :
7          case "joko" :
8          case "alex" : echo "Good morning boys..."; break;
9          case "ria" :
10         case "sari" :
11         case "tia" :
12         case "nova" : echo "Good morning ladies..."; break;
13     }
14 ?>

```

Jika nama yang dipanggil adalah nama laki-laki, hasil yang tampil adalah “*Good morning boys...*”, sedangkan jika variabel \$user berisi nama perempuan, yang akan tampil adalah “*Good morning ladies...*”.

## Perbedaan Antara IF dengan Switch

Walaupun memiliki tujuan yang hampir sama, struktur **IF** dan **switch** memiliki perbedaan mendasar.

Di dalam struktur **switch**, kondisi logika hanya diperiksa satu kali saja, yaitu di awal perintah **switch**. Sedangkan di dalam struktur **IF**, kondisi logika akan selalu diperiksa pada setiap perintah **IF**. Untuk struktur percabangan yang banyak, **switch** lebih cepat dieksekusi daripada **IF**.

Disisi lain, **switch** memiliki keterbatasan karena hanya bisa digunakan untuk tipe data sederhana seperti memeriksa nilai dari sebuah variabel. Struktur **switch** tidak bisa menangani kondisi yang lebih rumit seperti logika **AND** atau operasi perbandingan. Dengan keterbatasan ini, kita akan lebih sering menggunakan **IF** daripada **switch**.

## 9.5 Operator Conditional

Materi tentang operator sudah kita pelajari dalam bab sebelumnya, tapi ada 1 lagi operator yang belum saya bahas, yakni *operator conditional*.

**Operator conditional** merupakan satu-satunya *operator ternary* dalam PHP (operator yang membutuhkan 3 buah operand). Fungsi dari operator conditional mirip dengan kondisi **IF ELSE** tapi lebih singkat. Oleh karena itulah juga sering disebut sebagai *PHP Shorthand If Else*.

Sebagai contoh, perhatikan kode berikut:

```

1 <?php
2   if (7 > 5) {
3     $hasil = "Benar";
4   }
5   else {
6     $hasil = "Salah";
7   }
8
9   echo $hasil; // Benar
10 ?>

```

Dalam kode program ini saya membuat sebuah kondisi If Else sederhana. Jika kondisi  $(7 > 5)$  bernilai TRUE, variabel \$hasil akan berisi string “Benar”, jika kondisi bernilai FALSE, variabel \$hasil akan berisi string “Salah”.

Dengan menggunakan *operator conditional*, kode program diatas bisa ditulis menjadi:

```

1 <?php
2   $hasil = (7 > 5) ? "Benar" : "Salah";
3
4   echo $hasil; // benar
5 ?>

```

Perhatikan bagaimana penulisannya. *Operator conditional* dibuat dengan tiga karakter: ‘=’, ‘?’ dan ‘:’. Cara kerjanya bisa digambarkan sebagai berikut:

```
$hasil = kondisi ? "hasil jika kondisi true" : "hasil jika kondisi false";
```

Nilai dari variabel \$hasil tergantung dari “**kondisi**”. Jika kondisi benar, \$hasil akan berisi nilai sebelum tanda titik dua ( : ). Jika kondisi salah, variabel \$hasil akan diisi dengan nilai setelah tanda titik dua.

Berikut contoh lain dari penggunaan operator ini:

```

1 <?php
2   $user="guest";
3
4   $salam = ($user=="admin") ? "Welcome Admin!" : "Maaf, anda bukan Admin";
5
6   echo $salam; // Maaf, anda bukan Admin
7 ?>

```

Ini sebenarnya sama dengan kondisi If Else berikut:

```

1 <?php
2 $user="guest";
3
4 if ($user=="admin"){
5   $salam = "Welcome Admin!";
6 }
7 else {
8   $salam = "Maaf, anda bukan Admin";
9 }
10
11 echo $salam; // Maaf, anda bukan Admin
12 ?>

```

Penggunaan **Operator conditional** memang sangat singkat, tapi cukup susah dibaca terutama bagi yang belum paham tentang operator ini.

## 9.6 Struktur Perulangan FOR

Struktur perulangan (atau dalam bahasa inggris disebut dengan **loop**) adalah kode program untuk mengulang beberapa baris perintah. Di dalam PHP terdapat beberapa jenis instruksi perulangan: **FOR**, **WHILE**, **DO WHILE** dan **FOREACH**. Saya akan membahas struktur perulangan FOR terlebih dahulu.

Untuk membuat perulangan, harus tersedia 3 komponen, yaitu:

- Kondisi awal perulangan.
- Perintah program yang akan diulang.
- Kondisi akhir dimana perulangan harus berhenti.

Berikut struktur dasar perulangan FOR dalam PHP:

```

1 for (start; condition; increment)
2 {
3   statement;
4   statement;
5 }

```

Start adalah kondisi awal perulangan. Bagian ini biasanya diisi dengan sebuah variabel yang berfungsi sebagai **counter** (variabel pengontrol perulangan). Menjadi standar tidak resmi, variabel counter ditulis dengan **\$i**. Jika kita ingin mulai dari angka 1, kondisi start ditulis sebagai **\$i = 1**.



Membuat variabel counter dengan **\$i** bukanlah sebuah keharusan. Anda bisa menggantinya dengan variabel apapun, seperti **\$counter**, **\$hitung**, dll. Ini hanya kebiasaan mayoritas programmer.

**Condition** adalah kondisi yang menentukan kapan perulangan selesai. Dalam setiap perulangan (di kenal juga dengan istilah **iterasi**), *condition* akan terus diperiksa. Selama *condition* bernilai TRUE, iterasi akan dilakukan terus menerus.

Sebagai contoh, jika saya ingin menjalankan perulangan sebanyak 20 kali, di bagian condition ini bisa ditulis `$i <= 20`. Artinya, selama variabel counter `$i` nilainya kurang dari atau sama dengan 20, terus lakukan perulangan.

**Increment** adalah bagian yang digunakan untuk mengatur penambahan variabel counter pada setiap iterasi. Biasanya kita menggunakan operator increment seperti `$i++`.

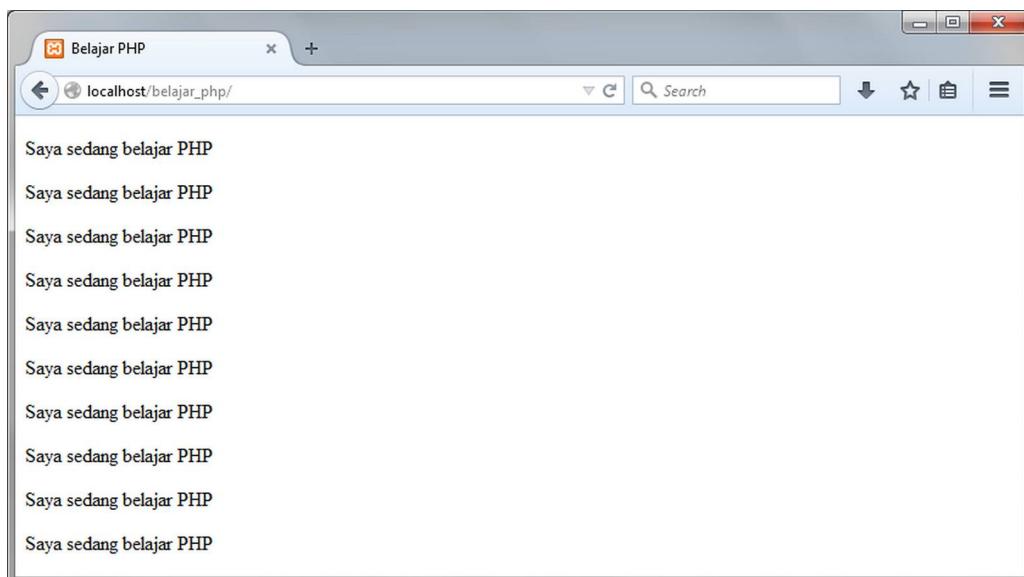
**Statement** adalah bagian kode program yang akan diulang. Statement harus berada di dalam tanda kurung kurawal ‘{’ dan ‘}’, kecuali statement tersebut hanya terdiri dari 1 baris.

Sebagai contoh, saya ingin menampilkan 10 baris kalimat “Saya sedang belajar PHP”. Berikut kode programnya:

```

1 <?php
2   for ($i= 1; $i <= 10; $i++)  {
3     echo "<p>Saya sedang belajar PHP</p>";
4   }
5 ?>

```



Gambar: Perulangan string ‘Saya sedang belajar PHP’

Bagian `for ($i= 1; $i <= 10; $i++)` bisa dibaca: *Lakukan perulangan mulai dari \$i = 1, dalam setiap iterasi tambah nilai \$i sebanyak 1 (\$i++), perulangan dilakukan terus menerus selama nilai \$i <= 10.*

Apakah anda bisa memodifikasi kode diatas untuk menampilkan 20 string? Yup, kita tinggal mengubah bagian **condition** menjadi `$i <= 20`:

```

1 <?php
2   for ($i= 1; $i <= 20; $i++)  {
3     echo "<p>Saya sedang belajar PHP</p>";
4   }
5 ?>

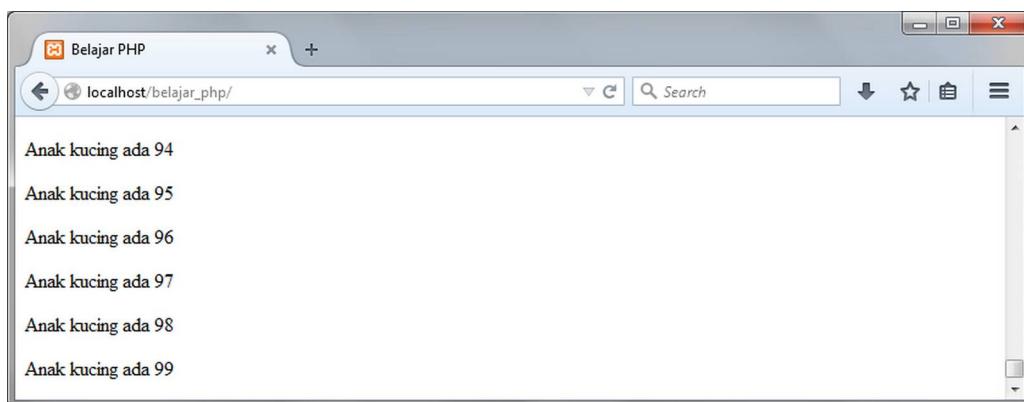
```

Selanjutnya, saya ingin menampilkan string “Anak kucing ada 1”, “Anak kucing ada 2”, “Anak kucing ada 3”, dst hingga anak kucing ke 99. Dapatkah anda membayangkan kode programnya?:

```

1 <?php
2   for ($i= 1; $i < 100; $i++)  {
3     echo "<p>Anak kucing ada $i</p>";
4   }
5 ?>

```



Gambar: Perulangan ‘anak kucing’ 99 kali

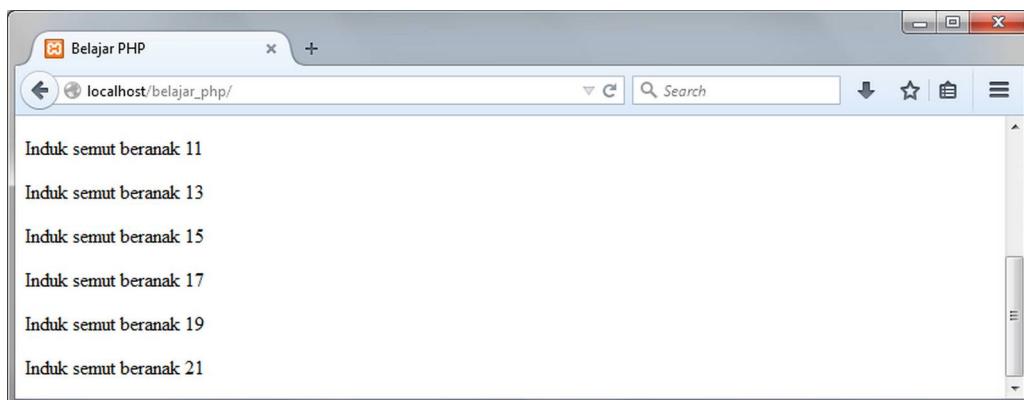
Perhatikan bahwa variabel counter `$i` bisa diakses dari dalam perulangan. Dalam tiap iterasi, nilai dari `$i` akan terus menaik sesuai kondisi bagian counter `$i++`. Selain itu, pada bagian *condition* saya menulis `$i < 100`. Ini berarti nilai maksimal `$i` adalah 99, bukan seratus. Tapi jika saya menulis `$i <= 100`, nilai maksimal `$i` menjadi 100.

Contoh berikutnya, saya ingin membuat teks dengan tampilan “Induk semut beranak 1”, “Induk semut beranak 3”, “Induk semut beranak 5”, dst hingga “Induk semut beranak 21”. Kali ini dalam tiap iterasi anak semut bertambah 2, dari 1, 3, 5, 7, hingga 21. Berikut kode programnya:

```

1 <?php
2   for ($i= 1; $i <= 21; $i = $i + 2)  {
3     echo "<p>Induk semut beranak $i</p>";
4   }
5 ?>

```



Gambar: Perulangan ‘induk semut beranak’ 99 kali

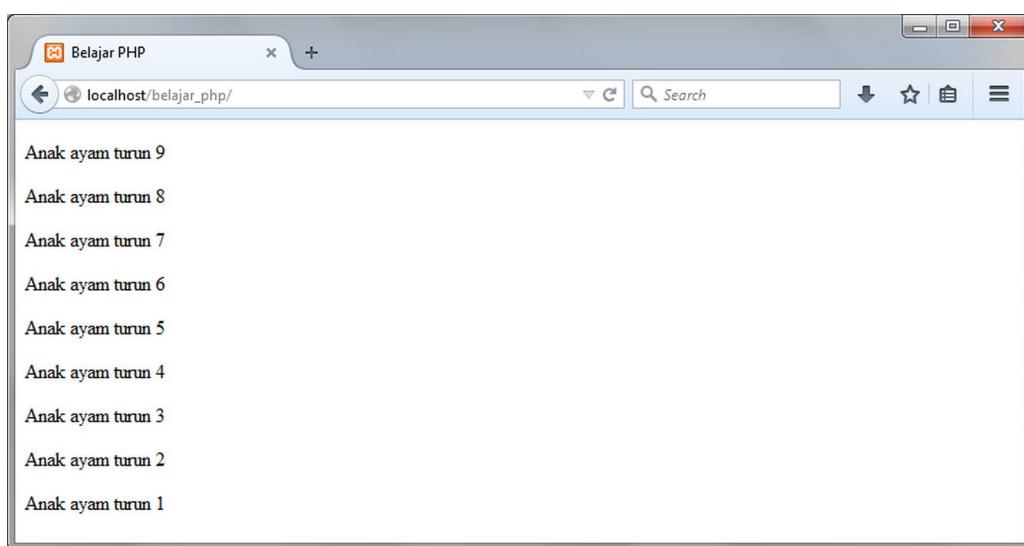
Sekarang, perhatikan bagian *increment*. Saya membuat `$i = $i + 2`, ini berarti dalam setiap iterasi variabel counter `$i` akan naik 2 angka, bukan 1 sebagaimana kita menggunakan operator increment `$i++`.

Bagaimana dengan lagu “Anak ayam turun 9”? Bisakah anda membuat string dengan “Anak ayam turun 9”, “Anak ayam turun 8”, hingga “Anak ayam turun 1”? Berikut kode programnya:

```

1 <?php
2   for ($i = 9; $i > 0; $i--) {
3     echo "<p>Anak ayam turun $i</p>";
4   }
5 ?>

```

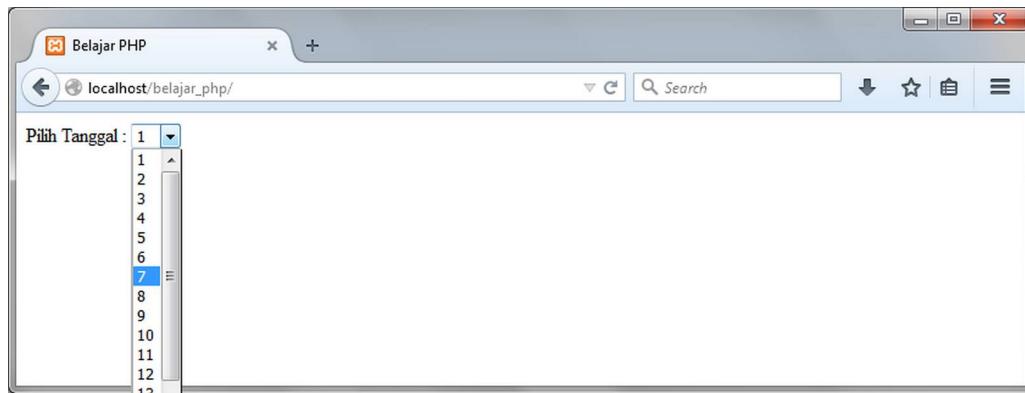


Gambar: Perulangan ‘anak ayam’ yang menurun dari 9 ke 1

Sekarang perulangan mundur dari 9 ke 1. Perhatikan cara penulisan perintah FOR. Ini bisa dibaca: *Mulai dari \$i = 10, dalam setiap iterasi kurangi nilai \$i sebanyak 1 (\$i-), lakukan perulangan selama nilai \$i > 0*.

Baik, kita sudahi bermain-main dengan anak ayam, mari masuk ke contoh yang lebih ‘nyata’.

Dapatkah anda membuat menu dropdown untuk memilih tanggal? Element dropdown dibuat dari tag `<select>` dan tag `<option>` HTML. Pilihan tanggal tersedia dari 1 sampai 31. Berikut tampilan yang saya inginkan:



Gambar: Pilihan tanggal HTML

Jika menggunakan HTML 'murni', kita harus membuat satu-satu tag `<option>`, seperti berikut:

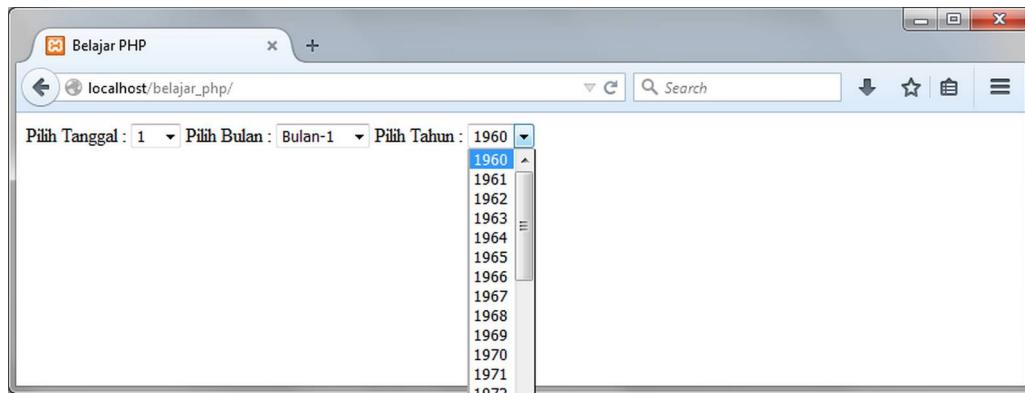
```
<select name="tgl" id="tgl">
  <option value=1>1</option>
  <option value=2>2</option>
  ...
  ...
  <option value=31>31</option>
</select>
```

Total akan membutuhkan 33 baris untuk membuat pilihan tersebut. Bagaimana jika menggunakan perulangan PHP? berikut kode yang saya gunakan:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <meta charset="UTF-8">
5    <title>Belajar PHP</title>
6  </head>
7  <body>
8  Pilih Tanggal :
9  <select name="tgl">
10 <?php
11   for ($i = 1; $i <= 31; $i++) {
12     echo "<option value = $i > $i </option>";
13   }
14 <?>
15 </select>
16 </body>
17 </html>
```

Hanya butuh 7 baris (termasuk tag <select>), jauh lebih efisien. Contoh ini juga memperlihatkan bahwa dengan apa yang sudah kita pelajari sejauh ini, sudah bisa langsung diimplementasikan ke dalam proses pembuatan web.

Sebagai latihan tambahan, dapatkah anda melengkapinya dengan pilihan bulan dan tahun? Seperti tampilan berikut:



Gambar: Tag <select> untuk pilihan tanggal, bulan, dan tahun

Berikut kode yang saya gunakan:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6  </head>
7  <body>
8  Pilih Tanggal :
9  <select name="tgl">
10 <?php
11     for ($i = 1; $i <= 31; $i++) {
12         echo "<option value = $i > $i </option>";
13     }
14     ?>
15 </select>
16
17 Pilih Bulan :
18 <select name="bln">
19 <?php
20     for ($i = 1; $i <= 12; $i++) {
21         echo "<option value = Bulan-$i > Bulan-$i </option>";
22     }
23     ?>
24 </select>
25
26 Pilih Tahun :
```

```

27 <select name="thn">
28   <?php
29     for ($i = 1960; $i <= 2015; $i++) {
30       echo "<option value = $i > $i </option>";
31     }
32   ?>
33 </select>
34 </body>
35 </html>

```

## Infinity Loop

Ketika membuat kondisi akhir dari perulangan, kita harus memperhatikan perulangan harus berhenti. Jika kondisi akhir tidak pernah terpenuhi, perulangan akan berjalan terus menerus (selamanya!). Hal ini dikenal sebagai **Infinity loop**.

Perhatikan kode berikut:

```

1 <?php
2   for ($i= 20; $i >= 1; $i++) {
3     echo $i;
4   }
5 ?>

```

Jika anda menjalankan kode diatas, proses perulangan akan berjalan terus menerus dan tidak akan pernah berhenti. Untuk menghentikannya harus dengan menutup paksa web browser.

Kesalahan dari struktur FOR, berada pada kondisi akhir perulangan. Saya membuat kondisi akhir **\$i >= 1**, tetapi variabel counter sudah dimulai dari **\$i= 20**, sedangkan variabel **\$i** akan terus bertambah 1 dalam tiap iterasi (**\$i++**). Dengan demikian, nilai **\$i** selalu lebih besar dari 1, dan kondisi **\$i >= 1** akan selalu bernilai TRUE.

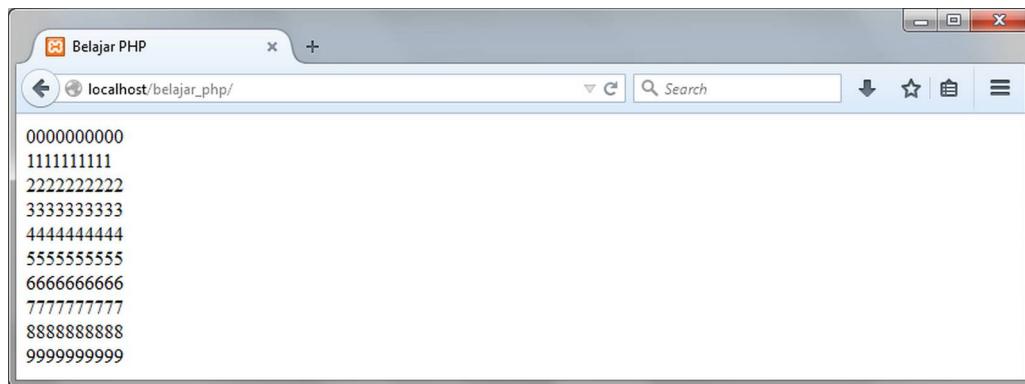
**Infinity loop** sebenarnya juga diperlukan dalam situasi tertentu. Tapi dalam kebanyakan kasus, kita akan menghindari kode program seperti ini.

## Nested Loop (Perulangan Bersarang)

Selain **infinity loop**, terdapat istilah lain yang sering digunakan dalam proses perulangan, yakni **nested loop**. Terjemahan bebasnya: *perulangan bersarang*.

**Nested loop** artinya membuat perulangan di dalam perulangan. Berikut contohnya:

```
1 <?php
2     for ($i=0; $i <10; $i++)
3     {
4         for ($j=0; $j <10; $j++)
5         {
6             echo $i;
7         }
8         echo "<br>";
9     }
10 ?>
```



Gambar: Nested loop

Dalam contoh diatas, saya membuat perulangan di dalam perulangan. Variabel counter `$i` digunakan untuk perulangan luar (*outer loop*). Sedangkan variabel counter `$j` digunakan untuk perulangan dalam (*inner loop*),

Nested loop biasanya digunakan untuk kode program yang cukup kompleks, seperti mengakses data dari dalam array 2 dimensi.



Kebiasaan umum programmer, menggunakan variabel `$i`, `$j`, `$k`, dst sebagai variabel counter dalam nested loop.

## Alternatif Penulisan Perulangan FOR

Sama seperti struktur IF, PHP juga memiliki alternatif penulisan perulangan FOR. Tanda kurung kurawal yang berfungsi sebagai penanda blok program, diganti dengan titik dua dan `endfor`.

Berikut contoh alternatif penulisan perulangan FOR:

```

1 <?php
2 for ($i= 1; $i <= 15; $i++) :
3   echo $i;
4   echo "<br />";
5 endfor;
6 ?>

```

Perbedaan dengan penulisan FOR ‘biasa’ terdapat pada penanda awal blok, dimana kita menggunakan tanda titik dua (:) dan diakhiri dengan **endfor**.

## 9.7 Struktur Perulangan WHILE

Salah satu syarat dari perulangan FOR yang baru saja kita pelajari adalah kondisi akhir perulangan sudah harus diketahui. Misalnya kita harus bisa memastikan perulangan akan dilakukan 10 kali, 100 kali, atau 1000 kali. Tapi bagaimana jika kondisi akhir ini belum bisa dipastikan?

Contohnya, kita ingin membuat program tebak angka. Pengunjung akan menebak 1 angka dari 1 sampai 10. Untuk kondisi ini, kita tidak bisa memastikan berapa kali pengunjung akan mencoba hingga benar. Mungkin butuh 1, 2, 5, atau 10 kali sebelum angka tersebut berhasil ditebak.

Untuk kasus seperti ini, PHP (dan juga bahasa pemograman lain) menyediakan struktur perulangan **WHILE**. Perulangan WHILE cocok digunakan dimana kondisi akhir perulangan belum diketahui pada saat perulangan ditulis.

Struktur dasar perulangan WHILE di dalam PHP adalah sebagai berikut:

```

start;
while (condition)
{
  statement;
  statement;
  increment;
}

```

**Start** adalah kondisi awal perulangan, disini kita bisa mempersiapkan variabel *counter* yang digunakan untuk mengatur **condition**.

**Condition** adalah kondisi yang harus dipenuhi agar perulangan berlangsung. Kondisi ini mirip seperti dalam perulangan FOR. Selama *condition* bernilai TRUE, perulangan akan terus dilakukan. *Condition* akan diperiksa pada tiap perulangan. Jika hasilnya FALSE, proses perulangan akan berhenti.

**Statement** adalah kode program yang akan diulang. Kita bisa membuat kode program yang simple seperti perintah *echo*, atau kumpulan perintah yang lebih kompleks. Tanda kurung kurawal diperlukan untuk membatasi blok program yang akan diulang. Jika *statement* hanya terdiri dari 1 baris, tanda kurung kurawal boleh tidak ditulis.

**Increment** juga sama seperti pada perulangan FOR, yakni untuk mengatur agar variabel counter naik/turun. Pada perulangan WHILE, bagian *increment* ini juga bisa berupa sebuah statement yang akan merubah *condition* menjadi FALSE. Jika di dalam perulangan tidak ada suatu kode program yang mengubah variabel *counter*, proses perulangan tidak akan pernah berhenti (*infinity loop*).

Sebagai contoh pertama, saya ingin menampilkan 10 string “*Saya sedang belajar PHP*”, berikut kode programnya:

```

1 <?php
2     $i = 1;
3     while ($i <= 10) {
4         echo "<p>Saya sedang belajar PHP</p>";
5         $i++;
6     }
7 ?>

```

Perhatikan perbedaan cara penulisannya dengan perulangan FOR.

Pada awal program saya mempersiapkan variabel \$i dan memberikan nilai 1. Variabel \$i inilah yang akan digunakan sebagai counter untuk kondisi WHILE.

Setelah penulisan **while**, didalam tanda kurung adalah *condition* yang harus dipenuhi agar perulangan berjalan. Saya membuat kondisi (\$i <= 10), yang berarti selama nilai variabel \$i kurang dari 10, terus lakukan perulangan.

Penting untuk diperhatikan adalah logika pemograman untuk *condition*. **while (\$i <= 10)** juga berarti bahwa jika nilai variabel \$i = 11, hasilnya akan FALSE dan perulangan berhenti.

Di dalam kode program, kita harus membuat sebuah baris statement yang digunakan untuk terus merubah nilai \$i agar bisa mencapai angka lebih dari 10 supaya perulangan berhenti. Untuk ini, saya menggunakan operator increment \$i++. Baris inilah yang akan menambahkan nilai variabel counter \$i sebanyak 1 angka pada tiap perulangan.

Proses looping akan terus berjalan hingga pada perulangan ke 10, nilai \$i akan menjadi 11. Nilai ini menyebabkan kondisi while bernilai FALSE, sehingga proses perulangan berhenti. Kesalahan dalam memahami logika while sering menghasilkan perulangan yang akan memproses secara terus menerus (*infinity loop*).

Masih ingat dengan anak ayam? Berikut perulangan WHILE-nya:

```

1 <?php
2     $i = 9;
3     while ($i > 0) {
4         echo "<p>Anak ayam turun $i</p>";
5         $i--;
6     }
7 ?>

```

Secara umum, semua perulangan FOR bisa dikonversi menjadi perulangan WHILE. Tapi tidak sebaliknya. Perhatikan contoh kode berikut:

```

1 <?php
2   $salah = TRUE;
3   $tebak_angka = 8;
4
5   while ($salah) {
6     if ($tebak_angka==8) {
7       $salah = FALSE;
8       echo "<p>Anda benar!</p>";
9     }
10    else {
11      echo "<p>Jawaban anda salah, silahkan ulangi kembali</p>";
12    }
13  }
14 ?>

```

Ide dari kode diatas adalah sebuah program tebak angka. Selama angka yang ditebak bukan 8, perulangan akan terus dilakukan. Dalam aplikasi sebenarnya, nilai variabel `$tebak_angka` akan berasal dari form HTML. Jika jawaban salah, kita bisa me-load kembali halaman tersebut, sampai jawaban benar.



Jika anda mengubah variabel `$tebak_angka` menjadi selain 8, akan terjadi *infinity loop*. Kenapa? Karena variabel ‘counter’ `$salah` selalu bernilai TRUE.

Contoh seperti ini tidak bisa dibuat dengan perulangan FOR, karena kita tidak bisa pastikan berapa kali user menebak hingga benar.

## Infinity Loop

Struktur perulangan **WHILE** dan **DO WHILE** (yang akan kita pelajari sesaat lagi), lebih mudah terjadi *infinity loop* dibandingkan perulangan FOR. Karena besar kemungkinan kita lupa membuat kondisi kapan perulangan harus berhenti. Sebagai contoh, kode program berikut akan menghasilkan *infinity loop*:

```

1 <?php
2   $i=1;
3   while ($i <= 10)
4   {
5     echo "$i";
6   }
7 ?>

```

Bisakah anda menebak apa yang kurang dari kode program diatas? Yup, saya ‘lupa’ menambahkan perintah *increment* untuk menaikkan nilai variabel `$i`. Kode diatas seharusnya ditulis menjadi:

```

1 <?php
2   $i=1;
3   while ($i <= 10)
4   {
5     echo "$i";
6     $i++;
7   }
8 ?>

```

## Nested Loop (Perulangan Bersarang)

Walaupun struktur WHILE agak jarang digunakan untuk *nested loop*, kita juga bisa membuat perulangan bersarang, seperti contoh berikut:

```

1 <?php
2   $i=0;
3   while ($i < 10)
4   {
5     $j=0;
6     while ($j < 10)
7     {
8       echo $i;
9       $j++;
10    }
11   echo "<br>";
12   $i++;
13 }
14 ?>

```

## Alternatif Penulisan Perulangan WHILE

Sama seperti perulangan FOR, kita bisa mengganti penulisan blok perulangan WHILE dengan tanda titik dua dan **endwhile**, seperti contoh berikut:

```

1 <?php
2   $i=1;
3   while ($i <= 10):
4     echo "$i";
5     echo "<br />";
6     $i=$i+1;
7   endwhile;
8 ?>

```

## 9.8 Struktur Perulangan DO WHILE

Perulangan **WHILE** dan **DO WHILE** pada dasarnya hampir sama. Perbedaan terletak pada lokasi pengecekan kondisi perulangan.

Dalam struktur **WHILE**, pengecekan kondisi perulangan di lakukan di awal. Jika kondisi tidak terpenuhi, perulangan tidak akan pernah dijalankan.

Namun pada perulangan **DO WHILE**, pengecekan kondisi dilakukan di akhir, sehingga walaupun kondisi menghasilkan FALSE, perulangan akan tetap berjalan minimal 1 kali.

Berikut adalah format dasar struktur penulisan DO WHILE dalam PHP:

```
start;
do {
    statement;
    statement;
    increment;
} while (condition);
```

Sebagai contoh, perhatikan perulangan WHILE berikut:

```
1 <?php
2     $i=1000;
3     while ($i <= 10)
4     {
5         echo "$i";
6         echo "<p>Tidak akan tampil di browser</p>";
7         $i=$i+1;
8     }
9 ?>
```

Kode program diatas tidak akan menampilkan apa-apa, karena **condition while (\$i<=10)** sudah langsung menghasilkan nilai FALSE (karena saya mendefenisikan nilai \$i=1000, yang jelas lebih besar dari 10). Tapi jika perulangan diatas dijalankan dengan struktur **DO WHILE**, hasilnya akan berbeda:

```
1 <?php
2     $i=1000;
3     do
4     {
5         echo "$i";
6         echo "<p>Akan tampil di browser</p>";
7         $i=$i+1;
8     } while ($i <= 10);
9 ?>
```

Program diatas akan menampilkan "1000 <p>Akan tampil di browser </p>". Ini karena pada struktur DO WHILE, perulangan program akan tampil setidaknya 1 kali walaupun kondisi WHILE menghasilkan FALSE.

Anda juga bisa menggunakan struktur DO WHILE untuk perulangan bersarang (*nested loop*), dan alternatif penulisan dengan **endwhile**. Penjelasannya mirip dengan struktur WHILE, sehingga tidak akan saya bahas lagi.

Dibandingkan dengan FOR dan WHILE, perulangan DO WHILE relatif jarang dipakai.

## 9.9 Struktur Perulangan FOREACH

Struktur FOREACH adalah perulangan khusus yang digunakan untuk menangai data **array**. Sebagai contoh, untuk menampilkan isi dari sebuah array, saya bisa menulis sebagai berikut:

```
1  <?php
2      $nama = array("Andri", "Joko", "Sukma", "Rina", "Sari");
3
4      for ($i = 0; $i < 5; $i++)
5      {
6          echo "$nama[$i]";
7          echo "<br />";
8      }
9  ?>
10
11 /* Output -----
12 Andri
13 Joko
14 Sukma
15 Rina
16 Sari
17 ----- */
```

Pada contoh diatas, saya membuat perulangan FOR sebanyak 5 kali dengan variabel counter **\$i** dimulai dari angka 0 (karena index array dimulai dari 0). Tidak ada yang salah dari kode diatas, namun sebagai alternatif, saya bisa menggunakan perulangan **foreach**:

```

1 <?php
2     $nama = array("Andri", "Joko", "Sukma", "Rina", "Sari");
3
4     foreach ($nama as $val)
5     {
6         echo "$val";
7         echo "<br />";
8     }
9 ?>
10
11 /* Output-----
12 Andri
13 Joko
14 Sukma
15 Rina
16 Sari
17 -----*/

```

Perulangan **foreach** diatas juga akan menampilkan seluruh isi element array, namun dengan perintah yang lebih singkat. Selain itu, kita tidak perlu menulis berapa kali iterasi harus dilakukan, serta variabel increment. Perulangan foreach otomatis dijalankan sebanyak element yang ada di dalam array.

Berikut format dasar penulisan *foreach*:

```

foreach ($nama_array as $value)
{
    statement (...$value...)
}

```

Variabel **\$nama\_array** adalah array yang akan diproses. Sedangkan **\$value** adalah nama ‘variabel perantara’ yang berisi element array pada saat perulangan. Anda bebas ingin menggunakan nama lain untuk variabel perantara ini, seperti **\$value**, **\$val**, atau yang lainnya . Diantara kedua variabel ini terdapat *keyword* “as”.

Bagaimana dengan **assosiative array**? perulangan FOREACH juga punya format khusus untuk tipe array ini:

```

foreach ($nama_array as $key => $value)
{
    statement ($key...$value...)
}

```

Perbedaan dengan format sebelumnya, kali ini kita menggunakan variabel perantara kedua, yaitu **\$key**. Variabel **\$key** digunakan untuk menampung nilai *key* atau *label* dari *assosiative array*. Berikut contohnya:

```

1  <?php
2      $nama = array(
3          1 => "Andri",
4          6 => "Joko",
5          12 => "Sukma",
6          45 => "Rina",
7          55 => "Sari"
8      );
9
10     foreach ($nama as $kunci => $isi)
11     {
12         echo "Urutan ke-$kunci adalah $isi";
13         echo "<br />";
14     }
15 ?>
16
17 /* Output-----
18 Urutan ke-1 adalah Andri
19 Urutan ke-6 adalah Joko
20 Urutan ke-12 adalah Sukma
21 Urutan ke-45 adalah Rina
22 Urutan ke-55 adalah Sari
23 -----*/

```

Variabel `$nama` saya definisikan sebagai *assosiative array*. Pada perulangan `foreach`, saya menggunakan variabel perantara `$kunci => $isi`. Selama dalam perulangan, variabel `$kunci` akan berisi *key* dari array, sedangkan variabel `$isi` akan berisi *value* atau nilai dari array.

Proses menampilkan dan memproses array akan lebih mudah menggunakan perulangan `foreach` dibandingkan perulangan dasar seperti `for`. Terlebih lagi kita tidak perlu tau berapa banyak perulangan harus dilakukan. Perulangan `foreach` akan otomatis berhenti pada data array terakhir.

Jika anda sedang menangani data array dan ingin menampilkan hasilnya, pertimbangkan untuk menggunakan `foreach`.

## 9.10 Fungsi Keyword Break dan Continue dalam Perulangan

Ketika proses loop (perulangan) sedang berjalan, ada kalanya kita ingin menghentikan segera perulangan walaupun belum mencapai kondisi akhir.

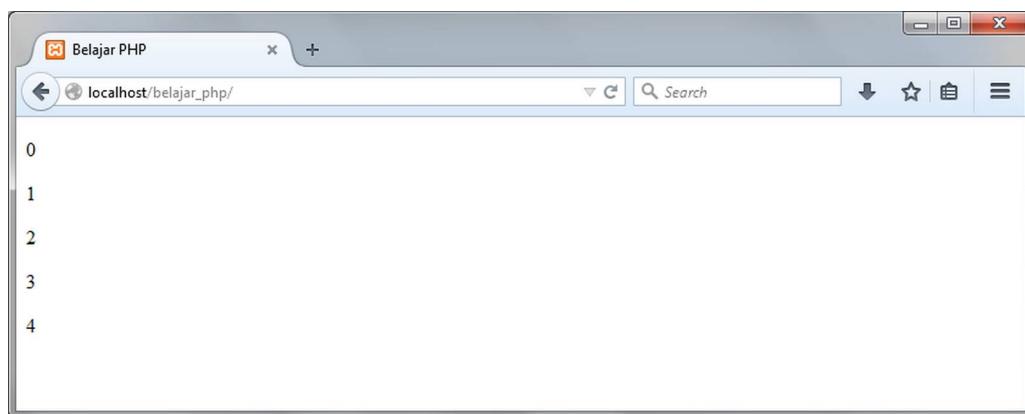
Sebagai contoh, saya ingin membuat kode program untuk mencari nama mahasiswa yang tersimpan di dalam sebuah array. Programnya bisa dibuat dengan melakukan pengecekan array mulai dari element pertama hingga element terakhir. Perulangan akan dilakukan sebanyak daftar mahasiswa yang ada, karena bisa saja nama mahasiswa yang ingin dicari berada di posisi terakhir.

Tapi, bagaimana jika nama mahasiswa itu sudah di dapatkan pada loop ke-3? Akan lebih efisien apabila perulangan langsung berhenti daripada tetap melanjutkan hingga akhir. Untuk keperluan inilah PHP menyediakan perintah (*keyword*) **break** dan **continue**.

Perintah **Break** berfungsi untuk menghentikan seluruh loop. Sedangkan **continue** hanya akan menghentikan 1 iterasi saja.

Perhatikan contoh berikut:

```
1 <?php
2     for ($i=0; $i <= 10; $i++) {
3         if ($i==5) {
4             break;
5         }
6         echo "<p>$i</p>";
7     }
8 ?>
```



Gambar: Perulangan berhenti di posisi  $\$i = 5$

Saya membuat perulangan for dari 0 sampai 10. Dalam keadaan normal, perintah **for** ( $\$i=0; \$i <= 10; \$i++$ ) akan memproses perulangan sebanyak 10 kali.

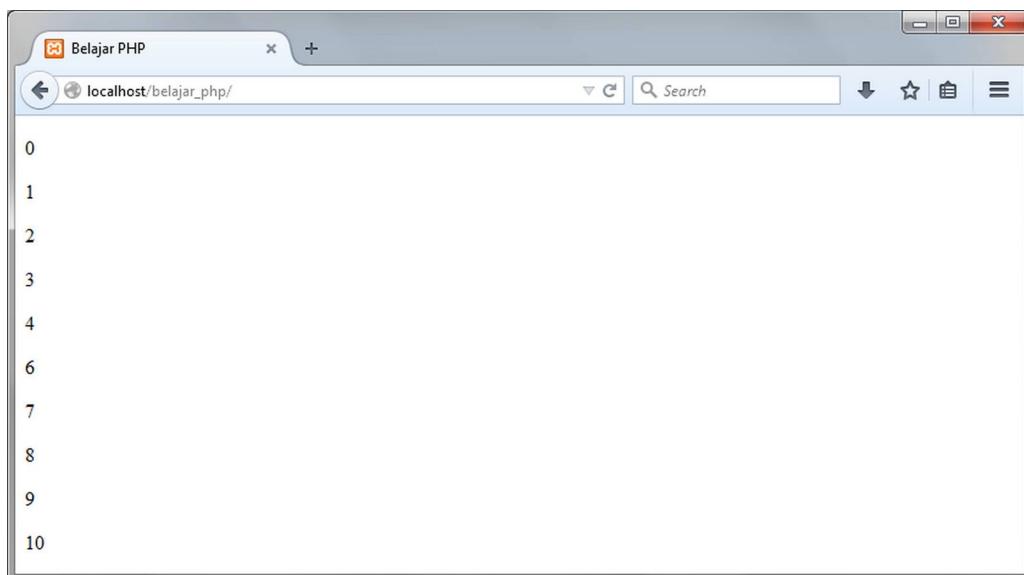
Pada baris ke-4 saya menambahkan sebuah struktur IF. Ketika nilai variabel counter **\$i** sama dengan 5, maka **break**. Perintah **break** akan membuat perulangan for langsung dihentikan pada saat itu juga.

Bagaimana dengan perintah **continue**? Mari kita coba:

```

1 <?php
2   for ($i=0; $i <=10; $i++) {
3     if ($i==5) {
4       continue;
5     }
6     echo "<p>$i</p>";
7   }
8 ?>

```



Gambar: \$i = 5 tidak ditampilkan

Seperti yang terlihat, perulangan tetap jalan terus hingga akhir. Tapi kemana angka 5? Ini terjadi karena ketika variabel counter **\$i** mencapai nilai 5, perintah **continue** akan dijalankan. Akibatnya, proses iterasi akan berhenti dan langsung lanjut ke iterasi berikutnya, dimana nilai **\$i** menjadi 6.

Kedua contoh diatas menggunakan perulangan **FOR**. Bagaimana untuk perulangan **WHILE**? Cara kerjanya tidak jauh berbeda:

```

1 <?php
2   $i=0;
3   while ($i <= 10)
4   {
5     $i++;
6     if ($i==5)
7     {
8       break;
9     }
10    echo "<p>$i</p>";
11  }
12 ?>

```

## Break pada Perulangan Bersarang (Nested Loop)

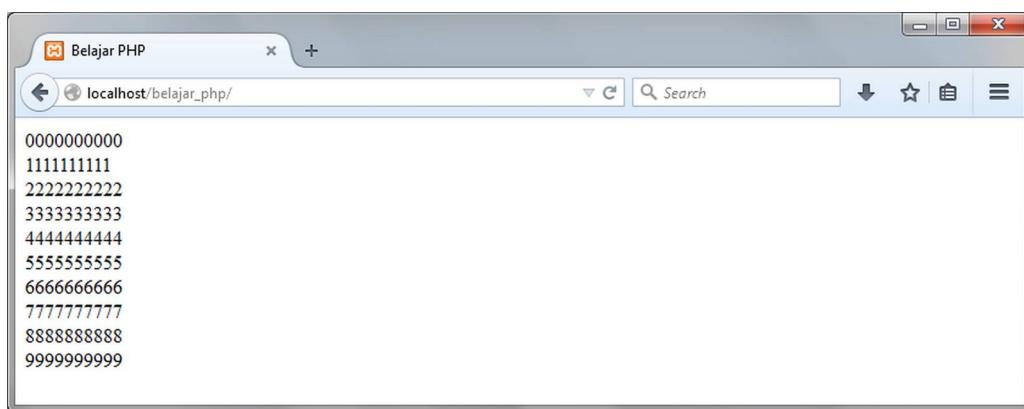
Untuk perulangan bersarang (*nested loop*), perintah **break** bisa bermakna ganda, yaitu apakah kita ingin menghentikan perulangan luar, atau perulangan dalam.

Agar lebih mudah dipahami, perhatikan contoh berikut:

```

1 <?php
2   for ($i=0; $i <10; $i++)
3   {
4     for ($j=0; $j <10; $j++)
5     {
6       echo $i;
7     }
8     echo "<br />";
9   }
10 ?>

```



Gambar: Nested Loop

Dalam perulangan tersebut, variabel counter **\$i** digunakan untuk perulangan luar (*outer loop*), sedangkan variabel counter **\$j** digunakan untuk perulangan dalam (*inner loop*).

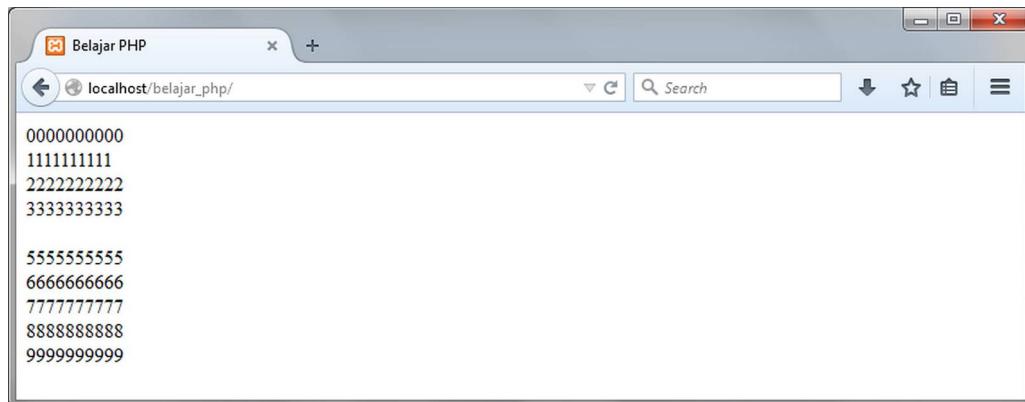
Jika saya membuat perintah **break** di dalam perulangan **\$j** (inner loop), maka yang akan dihentikan hanya perulangan **\$j** saja, seperti pada contoh program berikut ini:

```

1 <?php
2   for ($i=0; $i <10; $i++)
3   {
4     for ($j=0; $j <10; $j++)
5     {
6       if ($i==4)
7       {
8         break;
9       }
10      echo $i;

```

```
11      }
12  echo "<br>";
13  }
14 ?>
```



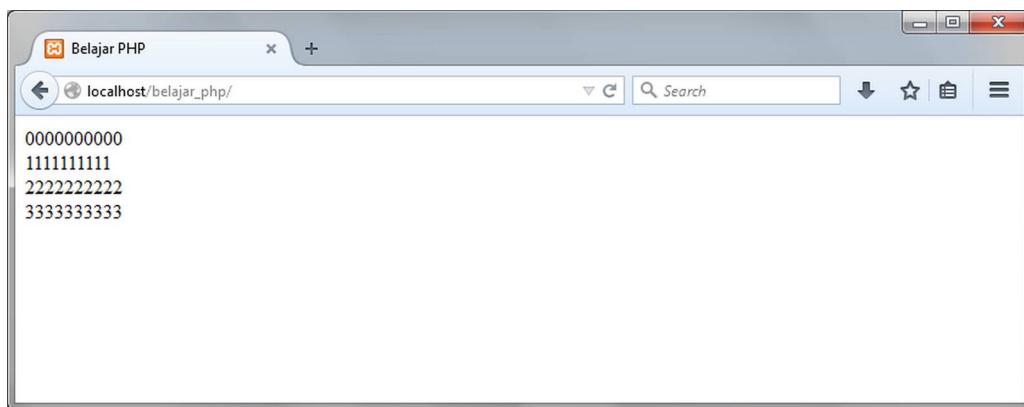
Gambar: Efek break pada nested loop

Bagaimana jika saya juga ingin menghentikan perulangan \$i?

Caranya adalah dengan mengubah perintah **break** menjadi **break 2**. Angka 2 berfungsi untuk menghentikan 2 level perulangan.

Berikut contoh program sebelumnya, dengan ditambahkan **break 2**:

```
1 <?php
2  for ($i=0; $i <10; $i++)
3  {
4      for ($j=0; $j <10; $j++)
5      {
6          if ($i==4)
7          {
8              break 2;
9          }
10         echo $i;
11     }
12     echo "<br>";
13 }
14 ?>
```



Gambar: Efek break 2 pada nested loop

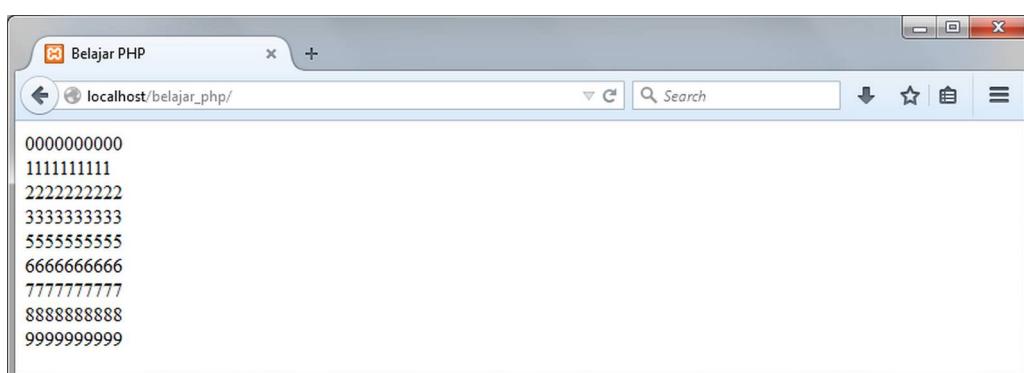
Jika anda membuat nested loop dengan 3 tingkatan, atau 3 level, maka kita bisa menggunakan perintah **break 3** untuk keluar dari perulangan terdalam.

Untuk perintah **continue**, caranya juga tidak berbeda:

```

1 <?php
2   for ($i=0; $i <10; $i++)
3   {
4     for ($j=0; $j <10; $j++)
5     {
6       if ($i==4)
7       {
8         continue 2;
9       }
10      echo $i;
11    }
12    echo "<br />";
13  }
14 ?>

```



Gambar: Efek continue 2 pada nested loop



Perintah **break 2** juga bisa ditulis sebagai **break(2)**. Begitu juga dengan **continue 2** bisa ditulis dengan **continue(2)**.

Dalam bab ini kita telah membahas struktur pemrograman PHP yang terdiri dari logika IF, ELSE IF, SWITCH, perulangan FOR, WHILE, DO WHILE, dan FOREACH. Berikutnya, kita akan masuk ke materi function PHP.

# 10. Function

Function atau *fungsi* merupakan inti dari konsep pemrograman yang dikenal sebagai **procedural programming**. Dalam bab ini kita akan mempelajari cara membuat dan menggunakan function di PHP.

## 10.1 Pengertian Function

Dalam merancang kode program, kadang kita sering membuat kode yang melakukan tugas yang sama secara berulang-ulang, seperti membaca tabel dari database, menampilkan penjumlahan, dan lain-lain. Tugas yang sama ini akan jauh lebih efektif jika dipecah dari program utama dan dijadikan sebuah *function*.

**Function** (atau *fungsi*) adalah kode program yang dirancang untuk menyelesaikan sebuah tugas tertentu, dan merupakan bagian dari program utama. *Function* diperlukan untuk memecah alur program yang besar menjadi beberapa kode program kecil agar mudah di kelola.

Di dunia programming, function kadang disebut sebagai **subroutine** (*subrutin*) atau **procedure** (*prosedur*). Implementasinya bisa berbeda-beda. Sebagai contoh, dalam bahasa pemrograman **Pascal**, *procedure* dan *function* adalah 2 hal yang berbeda, tapi tujuannya tetap sama, yakni membuat program ‘kecil’ di dalam program utama.

PHP sendiri menyediakan ratusan, bahkan ribuan *function* yang bisa kita gunakan untuk membantu membuat kode program. Mulai dari pemrosesan string, pencarian array, hingga pengaksesan database, semuanya dilakukan melalui *function*. Mengenai fungsi bawaan PHP ini akan saya bahas dalam bab berikutnya. Kali ini kita akan mempelajari cara membuat function dan bagaimana cara kerja function PHP .



Istilah **procedural programming** berasal dari function ini. Procedural programming berarti kita membuat kode program yang terdiri dari berbagai fungsi yang saling bekerja sama. Selain itu, terdapat istilah **object oriented programming**. Disini, object-lah yang dirancang untuk saling bekerja sama.

## 10.2 Membuat Function di PHP

Dalam contoh-contoh kode program pada bab sebelumnya, beberapa kali kita telah menggunakan fungsi bawaan PHP. Perintah seperti `var_dump()`, `define()` dan `phpinfo()` adalah **function**. Selain itu masih banyak fungsi lain bawaan PHP. Sebelum kesana, kita akan mempelajari dulu cara membuat *function* sendiri, atau istilah kerennya: ***user defined function***.

Sesuai dengan pengertiannya, *function* atau *fungsi* digunakan untuk membuat program kecil yang bisa menjalankan tugas-tugas tertentu. Dengan memecah program besar menjadi fungsi-fungsi kecil, kode kita menjadi lebih rapi, efisien, dan mudah dikelola.

Berikut adalah struktur dasar pembuatan fungsi di dalam PHP:

```
function nama_fungsi ($arg1, $arg2) {  
    statement;  
    statement;  
}
```

Untuk membuat *fungsi*, harus di dahului oleh *keyword function*, kemudian diikuti dengan nama *fungsi* tersebut. Untuk nama *function*, aturannya sama dengan *variabel*, seperti tidak boleh mengandung spasi, tidak boleh diawal angka, dst (anda bisa baca kembali aturan pembuatan *variabel* di bab 6).

Nama function juga bersifat **case insensitive**. Artinya tidak membedakan huruf besar dan huruf kecil. Tapi agar konsisten, sebaiknya tetap ditulis dengan huruf kecil dan menggunakan underscore sebagai pemisah kata (*snake\_case*).

Setelah nama function, berikutnya adalah penulisan **argumen**. Sederhananya, **argumen** adalah *inputan untuk function*. Sebuah function bisa terdiri dari 1, 2, 5 atau lebih argumen tergantung kebutuhan. Kita juga bisa membuat sebuah fungsi tanpa argumen sama sekali. Penulisan setiap argument dipisah dengan tanda koma.

**Statement** adalah kode program penyusun *function*. Statement ini harus berada di dalam *block function* yang ditandai dengan kurung kurawal '{' dan '}'.

Baik, cukup tentang teorinya, kita akan langsung praktik. Berikut adalah contoh sebuah *function* sederhana:

```
1 <?php  
2     function salam(){  
3         echo "<p>Selamat Pagi</p>";  
4     }  
5 ?>
```

Fungsi diatas bernama **salam** dan dibuat **tanpa argumen**. Isinya hanya sebuah perintah **echo** "Selamat Pagi". Jika anda menjalankan kode program tersebut, tidak akan tampil apa-apa di web browser. Kenapa? Karena kita harus 'memanggil' function ini.

## 10.3 Memanggil Function

Ketika mendefenisikan sebuah *function*, ini sama seperti mengisi sesuatu ke dalam *variabel*. Untuk menampilkan isi *variabel* ini, kita harus memberikan instruksi lanjutan, seperti perintah **echo**.

Demikian juga dengan *function*, kita harus 'memanggilnya' terlebih dahulu, atau dikenal dengan istilah '**calling a function**'. Ini dilakukan dengan menulis nama *function*, kemudian diikuti argumennya. Jika fungsi tersebut tidak memiliki argument, tetap ditulis dengan tanda kurung kosong. Berikut cara memanggil fungsi **salam**:

```

1 <?php
2     function salam(){
3         echo "<p>Selamat Pagi</p>";
4     }
5
6     salam(); // Selamat Pagi
7 ?>

```

Sekarang, di web browser akan tampil string “Selamat Pagi”. Perhatikan cara pemanggilan fungsi **salam** yang tanpa argumen. Kita tetap harus menulisnya dengan tanda kurung. Karena itulah penulisan nama function di buku atau website ditulis dengan tanda kurung, seperti **salam()**.

Keuntungan dari menggunakan fungsi adalah, kita bisa menjalankan perintah yang sama beberapa kali hanya dengan memanggil fungsi tersebut:

```

1 <?php
2     function salam(){
3         echo "<p>Selamat Pagi</p>";
4     }
5
6     salam(); // Selamat Pagi
7     salam(); // Selamat Pagi
8     salam(); // Selamat Pagi
9 ?>

```

Kali ini saya memanggil fungsi **salam()** sebanyak 3 kali. Hasilnya terlihat kalimat “Selamat Pagi” sebanyak 3 kali yang merupakan isi dari function **salam()**.

Bayangkan jika fungsi **salam()** ini sebenarnya menampilkan isi tabel dari database. Untuk menampilkan tabel yang sama, kita tinggal memanggil fungsi ini tanpa harus membuat ulang seluruh kode yang diperlukan. Jauh lebih efisien.

## 10.4 Argumen Function

Agar lebih fleksibel, sebuah fungsi bisa diinput dengan ‘sesuatu’. Sesuatu ini dikenal dengan **argumen**. Langsung saja kita lihat contoh penggunaannya:

```

1 <?php
2     function salam($nama){
3         echo "<p>Selamat Pagi, $nama</p>";
4     }
5 ?>

```

Kali ini saya menambahkan 1 argumen untuk fungsi **salam()**. *Argumen* pada sebuah fungsi adalah *variabel input* untuk fungsi tersebut. Di dalam function, kita bisa menggunakan variabel **\$nama** sesuai dengan kebutuhan. Kali ini saya ingin menampilkan argumen **\$nama** ke dalam perintah **echo**.

Ketika function **salam()** dipanggil, saya bisa menginput sesuatu, seperti contoh berikut:

```
1 <?php
2     function salam($nama){
3         echo "<p>Selamat Pagi, $nama</p>";
4     }
5
6     salam("Andi"); // Selamat Pagi, Andi
7 ?>
```

Perhatikan bagaimana string "Andi" dipindahkan ke dalam variabel argumen `$nama` di dalam function `salam()`. Dengan mengubah-ubah nilai argumen, kita bisa membuat salam kepada siapa saja, seperti contoh berikut:

```
1 <?php
2     function salam($nama){
3         echo "<p>Selamat Pagi, $nama</p>";
4     }
5
6     salam("Andi");           // Selamat Pagi, Andi
7     salam("Jojo");          // Selamat Pagi, Jojo
8     salam("Indonesia");     // Selamat Pagi, Indonesia
9     salam("Dunia...");      // Selamat Pagi, Dunia...
10 ?>
```

Bagaimana dengan fungsi `salam()` dengan 2 argumen? Berikut modifikasinya:

```
1 <?php
2     function salam($waktu, $nama){
3         echo "<p>Selamat $waktu, $nama </p>";
4     }
5
6     salam("Malam", "Andi");      // Selamat Malam, Andi
7     salam("Siang", "Jojo");      // Selamat Siang, Jojo
8     salam("Pagi", "Indonesia..."); // Selamat Pagi, Indonesia...
9 ?>
```

Kali ini fungsi `salam()` menerima 2 argumen, yakni `$waktu` dan `$nama`. Kedua argumen ini saya tampilkan menggunakan perintah `echo`. Perhatikan juga cara pemanggilan fungsi `salam()`, karena fungsi ini sekarang membutuhkan 2 argumen, penulisannya juga HARUS menyertakan 2 argumen.

Bagaimana jika saya memanggil fungsi `salam()` hanya dengan 1 argumen?

```
1 <?php
2     function salam($waktu,$nama){
3         echo "<p>Selamat $waktu, $nama </p>";
4     }
5
6     salam("Andi"); // Warning: Missing argument 2 for salam()
7 ?>
```

Seperti yang terlihat, PHP akan mengeluarkan *error*. Inilah syarat dari pemanggilan *function*. **Setiap fungsi, harus dipanggil sesuai dengan jumlah argumennya.**

Tapi, saya bisa melakukan hal seperti ini:

```
1 <?php
2     function salam($waktu,$nama){
3         echo "<p>Selamat $waktu, $nama </p>";
4     }
5
6     salam("Pagi",null); // Selamat Pagi,
7     salam("", "Joko"); // Selamat, Joko
8 ?>
```

Pemanggilan fungsi **salam()** diatas tetap dihitung sebagai 2 argumen, walaupun nilai yang diinput adalah *null* dan string kosong.

Nilai yang diinput ini juga tidak harus ditulis langsung pada saat pemanggilan, kita bisa meyimpannya ke dalam variabel:

```
1 <?php
2     function salam($waktu,$nama){
3         echo "<p>Selamat $waktu, $nama </p>";
4     }
5
6     $event="Belajar PHP";
7     $user="Andi";
8
9     salam($event, $user); // Selamat Belajar PHP, Andi
10 ?>
```

Sekarang, saya menginput argumen untuk fungsi **salam()** sebagai variabel. Yang juga patut dicermati, variabel **\$event** berbeda dengan variabel **\$waktu**, walaupun isinya sama-sama string "Belajar PHP". Variabel **\$waktu** adalah variabel lokal di dalam function. Sedangkan variabel **\$event** adalah variabel global yang berada di luar fungsi.

## Argumen vs Parameter

Jika anda mempelajari buku teks algoritma dan pemrograman, terdapat istilah **argumen** dan **parameter**. Keduanya sama, tetapi sedikit berbeda.

**Parameter** adalah sebutan untuk inputan fungsi pada saat pendefenisian fungsi tersebut. Variabel `$waktu` dan `$nama` dari fungsi `salam()` merupakan *parameter*.

Sedangkan **Argumen** adalah sebutan untuk inputan fungsi pada saat pemanggilan fungsi tersebut. Variabel `$event` dan `$user` dari pemanggilan fungsi `salam()` merupakan *argumen*.

Dalam penggunaan sehari-hari, kedua istilah ini sering dipertukarkan. Di dalam dokumentasi resmi PHP, istilah yang sering dipakai adalah **argumen**. Oleh karena itu pula dalam buku ini saya lebih sering memanggil variabel inputan fungsi sebagai *argumen*. Pada dasarnya, baik *argumen* maupun *parameter* berisi data yang sama.

## 10.5 Mengembalikan Nilai Function

Agar lebih fleksibel, sebuah fungsi seharusnya mengembalikan nilai, bukan langsung menampilkan nilai dengan perintah `echo`. Sebagai contoh, ketika kita memanggil fungsi `salam()`, akan langsung tampil string “Selamat Pagi” ke dalam web browser.

Perhatikan contoh fungsi berikut:

```
1 <?php
2     function tambah($satu,$dua){
3         $hasil = $satu + $dua;
4         echo $hasil;
5     }
6
7     tambah(6,10);    // 16
8     echo "<br>";
9     tambah(100,99); // 199
10    ?>
```

Kali ini saya membuat fungsi `tambah()` dengan 2 argumen. Sesuai dengan namanya, fungsi ini menambahkan argumen pertama dengan argumen kedua dan menampilkan hasilnya. Bagaimana jika saya ingin memproses dulu nilai ini sebelum ditampilkan? Atau saya ingin menyimpannya dulu kedalam variabel? Berikut perubahannya:

```
1 <?php
2     function tambah($satu,$dua){
3         $hasil = $satu + $dua;
4         return $hasil;
5     }
6
7     $a=tambah(6,10);
8     echo $a; // 16
9 ?>
```

Sekarang, di dalam function **tambah()**, saya tidak lagi menggunakan perintah *echo*, tapi menggantinya dengan perintah **return**. Perintah **return** menginstruksikan sebuah agar nilai akhir fungsi ‘*dikembalikan*’. Dalam contoh diatas, nilai yang dikembalikan adalah variabel **\$hasil**.

Ketika saya memanggil fungsi **tambah(6,10)**, fungsi ini akan mengembalikan nilai. Selanjutnya, nilai kembalian ini saya simpan ke dalam variabel **\$a**. Kemudian hasilnya di tampilkan dengan perintah *echo*.

Dengan cara seperti ini, nilai atau output dari sebuah fungsi menjadi lebih fleksibel. Kita bisa memutuskan ingin ‘diapakan’ nilai ini, apakah disimpan di dalam variabel (seperti contoh diatas), atau digunakan dalam operasi lain. Seperti contoh berikut:

```
1 <?php
2     function tambah($satu,$dua){
3         $hasil = $satu + $dua;
4         return $hasil;
5     }
6
7     $a=tambah(6,10);
8     $b=tambah($a,9);
9
10    echo $b; // 25
11 ?>
```

Kali ini, variabel **\$a** yang berisi hasil dari fungsi **tambah(6,10)** saya gunakan sebagai input untuk fungsi **tambah(\$a,9)**. Hasilnya kemudian disimpan ke dalam variabel **\$b** dan ditampilkan dengan perintah *echo*.

Nilai kembalian ini juga tidak harus disimpan ke dalam variabel, tetapi juga bisa langsung ditampilkan pada saat pemanggilan perintah *echo*:

```
1 <?php
2     function tambah($satu,$dua){
3         $hasil = $satu + $dua;
4         return $hasil;
5     }
6
7     echo tambah(5,7);
8     echo "<br>";
9
10    echo "Hasil dari 6 + 9 adalah: ". tambah(6,9);
11    echo "<br>";
12
13    echo tambah(99,1).", Bisa didapat dari 99 + 1";
14 ?>
15
16 /* Output-----
17 12
18 Hasil dari 6 + 9 adalah: 15
19 100, Bisa didapat dari 99 + 1
20 -----*/
```

Cara pemanggilan fungsi seperti ini akan banyak kita gunakan untuk fungsi-fungsi singkat PHP. Selain mengembalikan nilai, perintah **return** juga langsung **memberhentikan fungsi**:

```
1 <?php
2     function tambah($satu,$dua){
3         $hasil = $satu + $dua;
4         return $hasil;
5         echo "Kalimat ini tidak akan pernah dijalankan...";
6     }
7
8     echo tambah(5,7); // 12
9 ?>
```

Perintah **echo** di dalam fungsi **tambah()** tidak akan pernah dijalankan, karena ketika di dalam sebuah fungsi ditemukan perintah **return**, proses fungsi langsung berhenti dan seluruh perintah lain setelahnya akan diabaikan. Ini mirip seperti perintah **break** pada perulangan.

Selain itu, perintah **return** juga hanya bisa mengembalikan 1 nilai atau 1 variabel. Tapi bagaimana jika saya tetap ingin mengembalikan 2 nilai atau lebih? Gunakan **array**:

```

1 <?php
2  function tambah_kurang($satu,$dua){
3      $hasil[] = $satu + $dua;
4      $hasil[] = $satu - $dua;
5      return $hasil;
6  }
7
8  $a=tambah_kurang(100,30);
9
10 echo "100 tambah 30 = ".$a[0]; // 100 tambah 30 = 130
11 echo "<br>";
12
13 echo "100 kurang 30 = ".$a[1]; // 100 kurang 30 = 70
14 ?>

```

Fungsi `tambah_kurang()` saya rancang untuk mengembalikan 2 nilai, yakni penjumlahan dan pengurangan kedua argumen-nya. Karena perintah return hanya bisa mengembalikan 1 varibel, saya harus membuatnya sebagai *array*.

## 10.6 Variable Scope

Pembahasan tentang **variable scope** berkaitan dengan *sejauh mana sebuah variabel masih dapat diakses*. Ketika sebuah variabel di definisikan di dalam function, variabel tersebut berstatus sebagai **local variable**. Variabel ini TIDAK BISA diakses dari luar fungsi tersebut. Berikut contoh kasusnya:

```

1 <?php
2  function test(){
3      $a="variabel lokal";
4  }
5
6  test();
7  echo $a; // Notice: Undefined variable: a
8 ?>

```

Kode program diatas akan menghasilkan **error**, karena variabel `$a` merupakan *local variable* yang tidak bisa diakses dari luar fungsi.

Jika terdapat *local variable*, tentu juga ada *global variable*. *Global variable* adalah sebutan untuk variabel yang di definisikan diluar fungsi. Tapi di dalam fungsi, variabel ini juga TIDAK BISA diakses, berikut contohnya:

```

1 <?php
2   $a="variabel global";
3
4   function test(){
5     echo $a;
6   }
7
8   test(); // Notice: Undefined variable: a
9 ?>

```

Dari kedua contoh kasus tentang **local variable** dan **global variable**, dapat disimpulkan bahwa sebuah *function* memiliki ‘dunianya’ sendiri. Variabel yang didefinisikan di dalam fungsi tidak bisa diakses dari luar, begitu juga sebaliknya. Bagaimana jika nama variabel itu memiliki nama yang sama? Mari kita coba:

```

1 <?php
2   $a=100;
3
4   function test(){
5     $a=500;
6     echo $a;
7   }
8
9   echo $a;
10  echo "<br>";
11
12  test();
13  echo "<br>";
14
15  echo $a;
16  echo "<br>";
17 ?>

```

Sebelum saya menjalankan kode program diatas, bisakah anda menebak bagaimana hasilnya? perhatikan bahwa variabel \$a di definisikan di 2 tempat, sebelum function, dan setelah function. Berikut hasil dari kode program diatas:

```

100
500
100

```

Walaupun saya telah menimpa nilai variabel \$a di dalam *function test()*, tetap tidak berefek apa-apa ke variabel \$a diluar function. Kesimpulannya, walaupun memiliki nama sama, efek *variable scope* membuat kedua variabel ini berbeda.

Untuk situasi khusus, kita bisa memaksa *function* untuk membaca *global variabel*, yakni dengan menambahkan keyword **global** di depan variabel tersebut:

```
1 <?php
2     $a=100;
3
4     function test(){
5         global $a;
6         $a=500;
7         echo $a;
8     }
9
10    echo $a;           // 100
11    echo "<br>";
12
13    test();           // 500
14    echo "<br>";
15
16    echo $a;           // 500
17    echo "<br>";
18 ?>
```

Keyword **global \$a** di dalam function **test()** akan memaksa fungsi tersebut menggunakan *variabel global*. Sekarang, ketika saya mengubah isi variabel **\$a** menjadi 500 dari dalam fungsi, nilai variabel **\$a** diluar fungsi juga akan ikut berubah.

Penggunaan perintah **global** seperti ini tidak terlalu banyak digunakan. Bahkan dari beberapa referensi, hal ini tidak disarankan. Mengubah variabel *global* dari dalam sebuah fungsi sering menjadi sumber bug yang susah dicari.

Sebaiknya kita menggunakan *argumen* untuk menginput sesuatu ke dalam fungsi dan mengambil nilai fungsi tersebut dengan perintah **return**. Sehingga, sebuah fungsi tetap menjadi unit yang terpisah dan tidak ada variabel yang saling tercampur.

## 10.7 Static Variable

**Static Variabel**, atau *variabel statis* adalah jenis variabel yang mempertahankan nilainya dalam setiap pemanggilan fungsi. Pada kondisi normal, nilai sebuah variabel akan otomatis dihapus pada saat fungsi selesai dijalankan dan akan dibuat ulang pada saat fungsi tersebut dipanggil kembali. Perhatikan contoh berikut:

```

1 <?php
2     function coba()
3     {
4         $a=0;
5         $a=$a+1;
6         return "Ini adalah pemanggilan ke-$a fungsi coba() <br />";
7     }
8
9     echo coba();      // Ini adalah pemanggilan ke-1 fungsi coba()
10    echo coba();     // Ini adalah pemanggilan ke-1 fungsi coba()
11    echo coba();     // Ini adalah pemanggilan ke-1 fungsi coba()
12    echo coba();     // Ini adalah pemanggilan ke-1 fungsi coba()
13 ?>

```

Dalam contoh diatas, saya membuat semacam *counter* untuk menghitung berapa kali fungsi **coba()** dipanggil. Seperti yang terlihat, variabel **\$a** akan terus di reset menjadi nol dalam setiap pemanggilan fungsi.

Jika variabel tersebut dinyatakan sebagai **static variabel**, nilainya akan tetap dipertahankan. Berikut contohnya:

```

1 <?php
2     function coba()
3     {
4         static $a=0;
5         $a=$a+1;
6         return "Ini adalah pemanggilan ke-$a fungsi coba() <br />";
7     }
8
9     echo coba();      // Ini adalah pemanggilan ke-1 fungsi coba()
10    echo coba();     // Ini adalah pemanggilan ke-2 fungsi coba()
11    echo coba();     // Ini adalah pemanggilan ke-3 fungsi coba()
12    echo coba();     // Ini adalah pemanggilan ke-4 fungsi coba()
13 ?>

```

Sekarang, nilai variabel **\$a** akan terus naik setiap kali pemanggilan fungsi **coba()**. Perhatikan, walaupun di dalam fungsi coba terdapat perintah untuk mereset variabel **\$a = 0**, tapi efek perintah **statis** akan mempertahankan nilai yang ada.

Sama seperti keyword **global**, perintah **statis** juga relatif jarang digunakan.

## 10.8 Default Argument

Ketika membahas tentang cara penulisan argumen, saya menulis bahwa setiap fungsi harus dipanggil sesuai dengan jumlah argumennya. Tapi kondisi ini bisa tidak berlaku jika di dalam fungsi tersebut terdapat *default argument*.

**Default argument** adalah nilai awal yang akan digunakan jika argument tidak ditulis. Berikut contohnya:

```
1 <?php
2     function salam($nama="Anton"){
3         echo "<p>Selamat Siang, $nama </p>";
4     }
5
6     salam();    // Selamat Siang, Anton
7 ?>
```

Saat mendefenisikan fungsi **salam()**, saya menambahkan sesuatu di argumen. Selain menulis variabel **\$nama**, saya mengisi variabel ini dengan “Anton”. Sekarang, string “Anton” akan digunakan ketika fungsi salam dipanggil tanpa argumen.

Bagaimana jika saya memanggilnya dengan argumen?

```
1 <?php
2     function salam($nama="Anton"){
3         echo "<p>Selamat Siang, $nama </p>";
4     }
5
6     salam();          // Selamat Siang, Anton
7     salam("Siska");  // Selamat Siang, Siska
8     salam("Indonesia..!"); // Selamat Siang, Indonesia..!
9 ?>
```

Ketika fungsi **salam()** ditulis dengan argument, nilai argumen inilah yang akan dipakai. Jika tidak ditulis, nilai default-lah yang akan digunakan.

Bagaimana jika ada 2 argumen? Tidak masalah:

```
1 <?php
2     function salam($waktu="Malam", $nama="Anton"){
3         echo "<p>Selamat $waktu, $nama </p>";
4     }
5
6     salam();          // Selamat Malam, Anton
7     salam("Pagi");   // Selamat Pagi, Anton
8     salam("Datang", "pak Presiden!"); // Selamat Datang, pak Presiden!
9 ?>
```

Selain itu, saya juga bisa membuat seperti berikut:

```

1  <?php
2      function salam($waktu, $nama="Anton"){
3          echo "<p>Selamat $waktu, $nama </p>";
4      }
5
6      salam("Sore");           // Selamat Sore, Anton
7      salam("Pagi");          // Selamat Pagi, Anton
8      salam("Datang", "Randy"); // Selamat Datang, Randy
9      salam();                // Warning: Missing argument 1 for salam()
10     ?>

```

Kali ini fungsi **salam()** terdiri dari 2 argumen. Argumen pertama tanpa nilai default, sedangkan argumen kedua memiliki nilai default (argumen **\$nama**). Jika didefinisikan seperti ini, fungsi **salam()** wajib ditulis sekurang-kurangnya dengan 1 argumen, yakni argumen untuk **\$waktu**.

Dengan menambahkan *default argument*, kita bisa merancang sebuah fungsi dengan berbagai keperluan. Saya bisa membuat fungsi yang menerima 4 argumen, tapi hanya 1 yang wajib ditulis, seperti berikut:

```

1  <?
2      function coba($a,$b="Sedang",$c="Belajar",$d="PHP") {
3          // isi function disini
4          // isi function disini
5      }
6  ?>

```

Fungsi **coba** diatas bisa dipanggil dengan 4 argumen, tapi sekurang-kurangnya argumen pertama yang wajib ditulis.

Ketika kita menggabung penulisan argumen yang tanpa nilai default, default argumen HARUS ditulis paling akhir. Sebagai contoh, saya tidak bisa mendefenisikan sebuah fungsi seperti ini:

```

1  <?
2      function coba($a="Belajar PHP",$b) {
3          // isi function disini
4          // isi function disini
5      }
6  ?>
7
8  <?
9      function coba2($a="Belajar PHP",$b, c="DuniaIlkom") {
10         // isi function disini
11         // isi function disini
12     }
13 ?>

```

Argumen yang tanpa nilai default, harus berada di posisi paling depan, baru diikuti dengan argumen yang memiliki nilai default. Fungsi **coba()** dan **coba2()** diatas seharusnya ditulis ulang menjadi:

```
1  <?
2  function coba($b, $a="Belajar PHP") {
3      // isi function disini
4      // isi function disini
5  }
6  ?>
7
8  <?
9  function coba2($b, $a="Belajar PHP", c="DuniaIlkom") {
10     // isi function disini
11     // isi function disini
12 }
13 ?>
```

---

Dalam bab ini saya telah membahas cara menulis fungsi yang dibuat sendiri (*user defined function*). Dasar-dasar ini diperlukan agar kita bisa dengan mudah memahami ribuan fungsi-fungsi bawaan PHP. Berikutnya, saya akan membahas beberapa fungsi bawaan PHP yang akan sering kita gunakan untuk membuat aplikasi web dengan PHP.

# 11. PHP Manual

Salah satu kunci untuk menguasai PHP adalah bisa membaca **PHP Manual**. Saya berani berkata bahwa jika seseorang mengaku paham PHP tapi tidak pernah berkenalan (atau bahkan tidak tau) dengan *PHP Manual*, akan sangat susah berkembang.

Bab kali ini saya khususkan untuk membahas dokumentasi resmi PHP ini dan membahas cara membaca fungsi bawaan PHP dari PHP Manual.

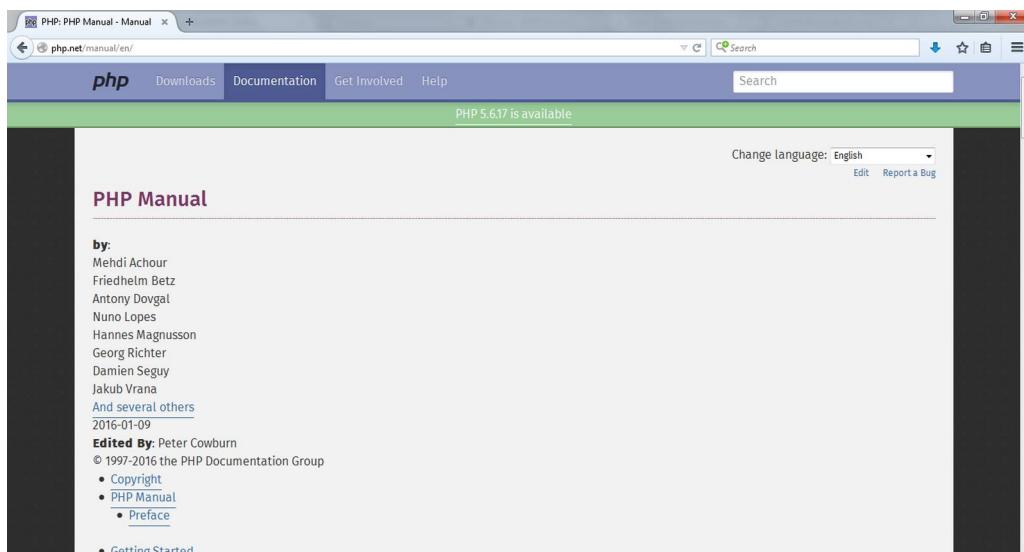
## 11.1 Mengenal PHP Manual

PHP merupakan satu dari sedikit bahasa pemrograman yang menyediakan dokumentasi resmi atau *manual* yang sangat terstruktur dan lengkap. Dokumentasi resmi PHP inilah yang disebut sebagai **PHP Manual**.

Sepanjang saya membuat program dengan PHP, *PHP Manual* sangat amat berharga. Terutama untuk mencari fungsi-fungsi PHP yang bisa memecahkan masalah yang saya hadapi.

Walaupun **PHP Manual** berbahasa Inggris, membacanya tidak terlalu sulit. Asal mengerti kosa kata dasar bahasa Inggris, anda bisa memahaminya. PHP manual juga disertai dengan banyak contoh kode program. Jika anda tidak mengerti dengan penjelasan yang ada, bisa langsung test menggunakan contoh kode program.

PHP Manual bisa diakses secara online di alamat: <http://php.net/manual/en/><sup>1</sup>, atau bisa juga di download dan diakses di komputer tanpa harus terkoneksi ke internet (offline).

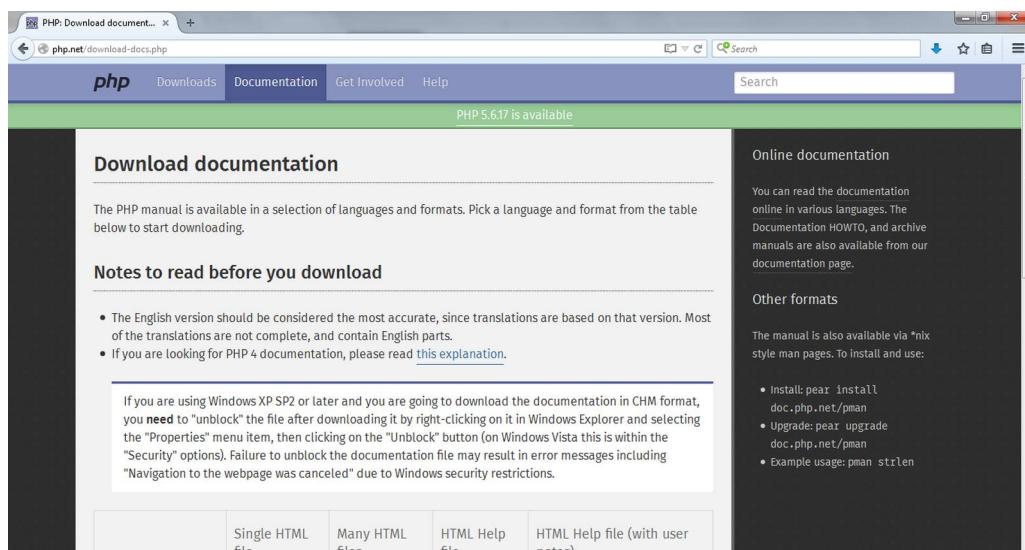


Gambar: Dokumentasi resmi PHP: "PHP Manual" di alamat <http://php.net/manual/en/>

<sup>1</sup><http://php.net/manual/en/>

## 11.2 Download PHP Manual

Selain diakses online, PHP Manual juga bisa di download dari halaman <http://php.net/download-docs.php><sup>2</sup>.



Gambar: Halaman download PHP Manual

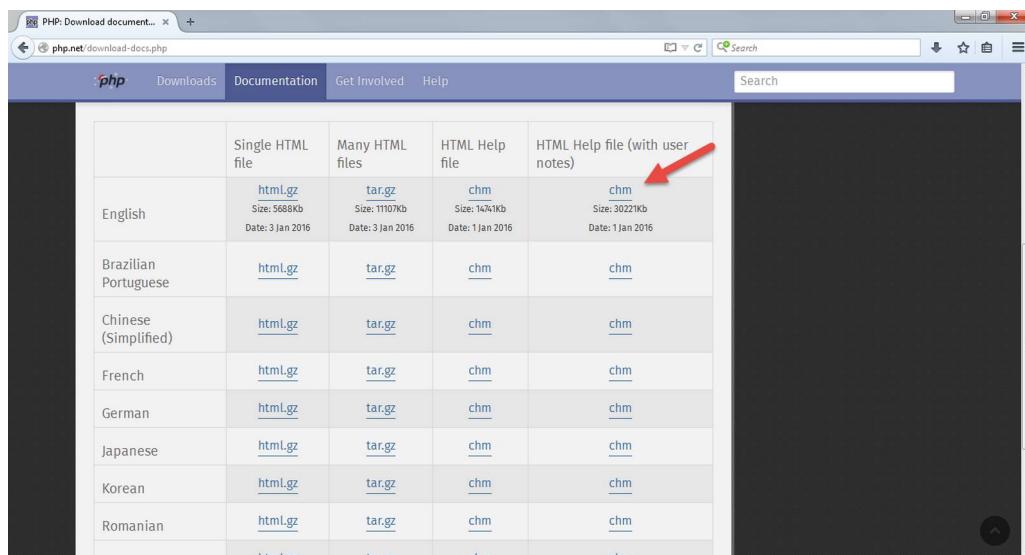
PHP menyediakan 2 jenis file dokumentasi, yakni file **HTML** dan **chm** (*HTML Help file*). Untuk format **HTML**, kita harus membukanya di web browser (mirip seperti file **HTML** biasa). Saya lebih suka menggunakan format **chm**, karena tidak perlu dibuka dari web browser dan lebih gampang digunakan.

Untuk versi **chm**, terdapat 2 pilihan, apakah hanya ‘dokumentasi asli’ saja, atau dengan komentar dari pengguna lain (*with user notes*). Ukuran file **chm with user notes** 2x lipat dari dokumentasi asli. Walaupun begitu, saya lebih menyarankan mengambil file **chm with user notes**, karena kita bisa membaca komentar-komentar dari programmer lain terkait materi yang dibahas. Komentar ini kadang berisi trik atau penjelasan tambahan yang sangat berguna.

Silahkan klik link *HTML Help file (with user notes)* dalam kolom bahasa inggris, atau klik link berikut: [http://php.net/get/php\\_enhanced\\_en.chm/from/a/mirror](http://php.net/get/php_enhanced_en.chm/from/a/mirror)<sup>3</sup>. Jika anda paham bahasa portugis, chinese, prancis, atau jerman, bisa memilih bahasa lain (sayangnya, belum tersedia versi Bahasa Indonesia).

<sup>2</sup><http://php.net/download-docs.php>

<sup>3</sup>[http://php.net/get/php\\_enhanced\\_en.chm/from/a/mirror](http://php.net/get/php_enhanced_en.chm/from/a/mirror)



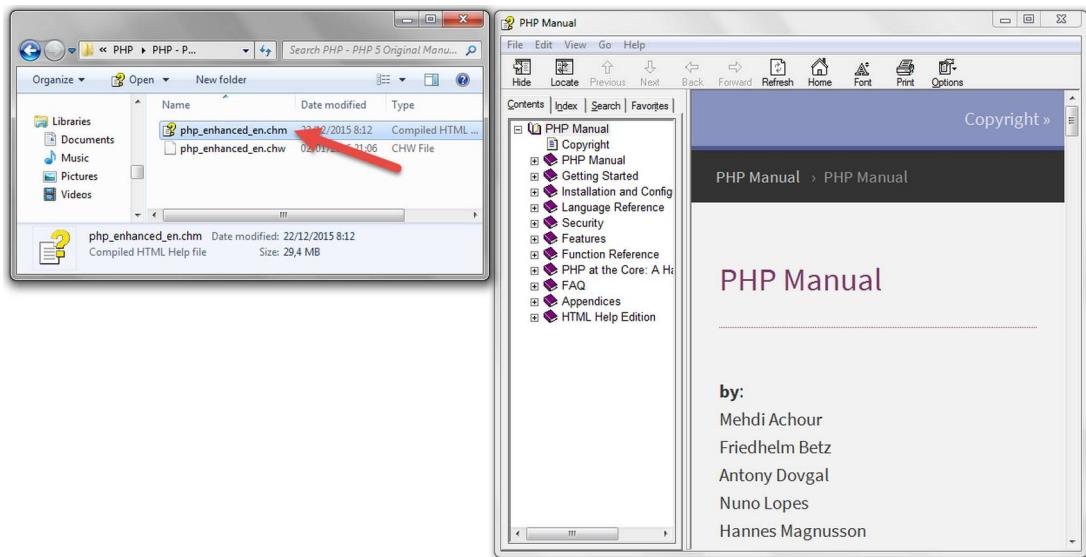
Gambar: Pilih file chm with user notes

Halaman selanjutnya adalah untuk memilih **mirror** atau lokasi server untuk download file. Secara teori, semakin dekat lokasi server dengan Indonesia, semakin cepat pula proses download. Namun efek ini tidak terlalu besar, anda bisa download dari server mana saja. Kali ini saya akan menggunakan file server **Singapore**. Klik link tersebut, dan proses download akan berjalan.



Gambar: Pilih mirror di server Singapore

Jika selesai, file **php\_enhanced\_en.chm** sudah bisa diakses. Silahkan double klik, dan file help yang berisi manual resmi PHP sudah bisa digunakan tanpa harus online.



Gambar: PHP Manual versi “offline” siap digunakan

### Mengatasi error “navigation to the webpage was canceled”

Dalam beberapa sistem operasi Windows, anda mungkin akan mendapatkan error “*navigation to the webpage was canceled*” ketika membuka file chm PHP Manual. Ini dikarenakan sistem keamanan dari Windows Vista keatas.

Solusinya, dari windows explorer klik kanan file “*php\_enhanced\_en.chm*”, lalu pilih **properties**. Dalam tab **General**, di bagian paling bawah terdapat tombol “**unblock**”. Klik tombol ini, lalu **OK**.



Gambar: Unblock file chm PHP Manual

Alternatif lain, bisa juga dengan men-copy file tersebut ke flashdisk atau drive yang tidak terhubung ke jaringan. Lebih lanjut bisa ke [stackoverflow.com](http://stackoverflow.com)<sup>a</sup>

<sup>a</sup><http://stackoverflow.com/questions/11438634/opening-a-chm-file-produces-navigation-to-the-webpage-was-canceled>

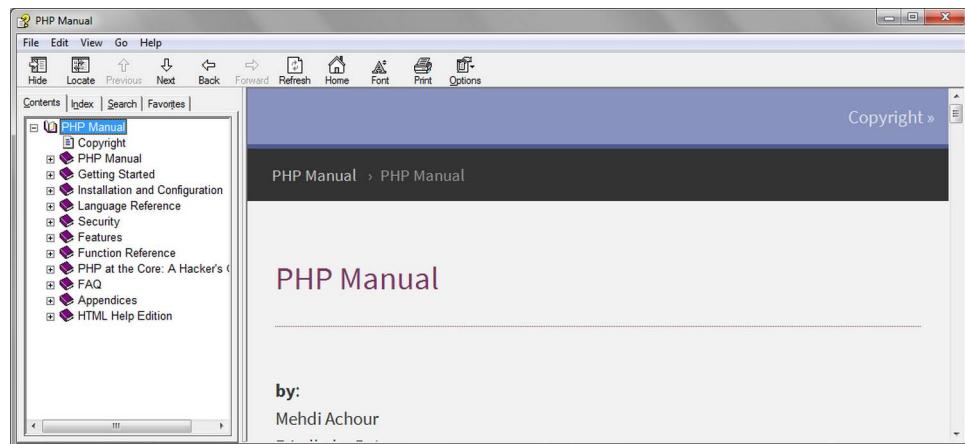
## 11.3 Cara Menggunakan PHP Manual

File **PHP Manual** yang baru saja kita download, berisi inti dari PHP. Tidak ada buku PHP manapun yang lebih lengkap dari *PHP Manual*, terutama untuk referensi fungsi dan fitur-fitur terbaru PHP. Beberapa materi yang ada dalam buku **PHP Uncover** ini juga saya ambil dari sini.

*PHP Manual* juga berisi konsep dasar PHP. Jika anda paham bahasa inggris, silahkan luangkan

waktu untuk membacanya. Namun tidak semua materi cocok untuk pemula. Kebanyakan diantaranya butuh pemahaman khusus, seperti materi tentang *object* dan *PHP security*.

Sebagaimana layaknya sebuah dokumentasi bahasa pemrograman, **PHP Manual** lebih kepada referensi. Didalamnya tidak dijelaskan cara penggunaannya untuk membuat website utuh. Tentunya anda tidak akan menemui penjelasan tentang HTML, CSS atau JavaScript disini.



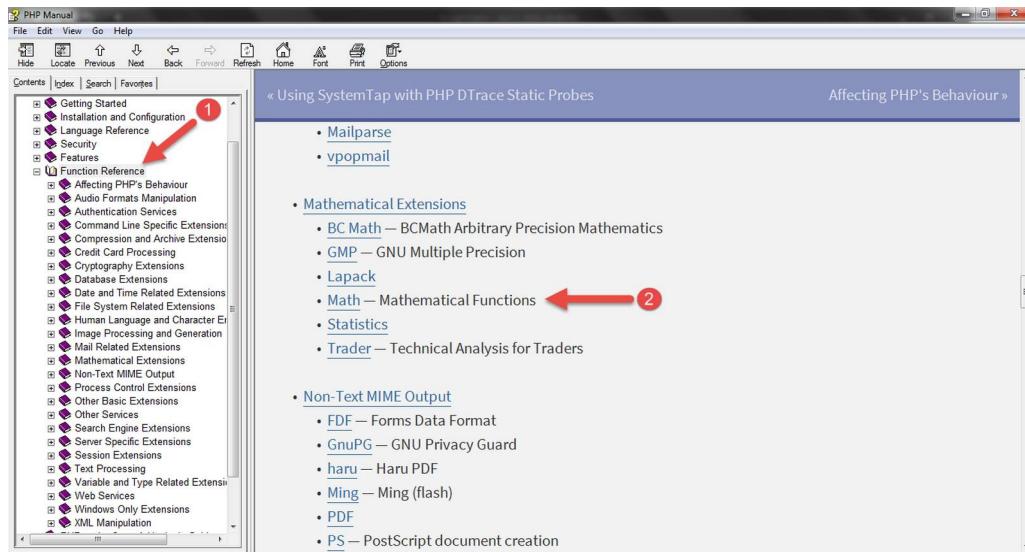
Gambar: Tampilan PHP Manual versi chm, dengan daftar isi di sisi kiri

Pada sisi kiri file **chm**, terdapat semacam daftar isi dari *PHP Manual*. Daftar isi ini dikelompokkan menurut kategorinya, mulai dari cara menginstall PHP, syntax dasar PHP (aturan penulisan kode PHP), hingga referensi fungsi-fungsi PHP.

Mulai dari bab ini hingga akhir buku, kita akan banyak menggunakan fungsi-fungsi bawaan PHP. Oleh karena itu saya akan fokus dengan bagian ini.

Silahkan klik **“Function Reference”** dari daftar isi, dan anda bisa melihat seluruh daftar katalog dari fungsi-fungsi bawaan PHP. Kategori ini diurutkan berdasarkan abjad. Sebagian besar dari fungsi-fungsi ini tidak akan kita bahas. Saya hanya fokus membahas fungsi-fungsi dasar PHP yang sering digunakan.

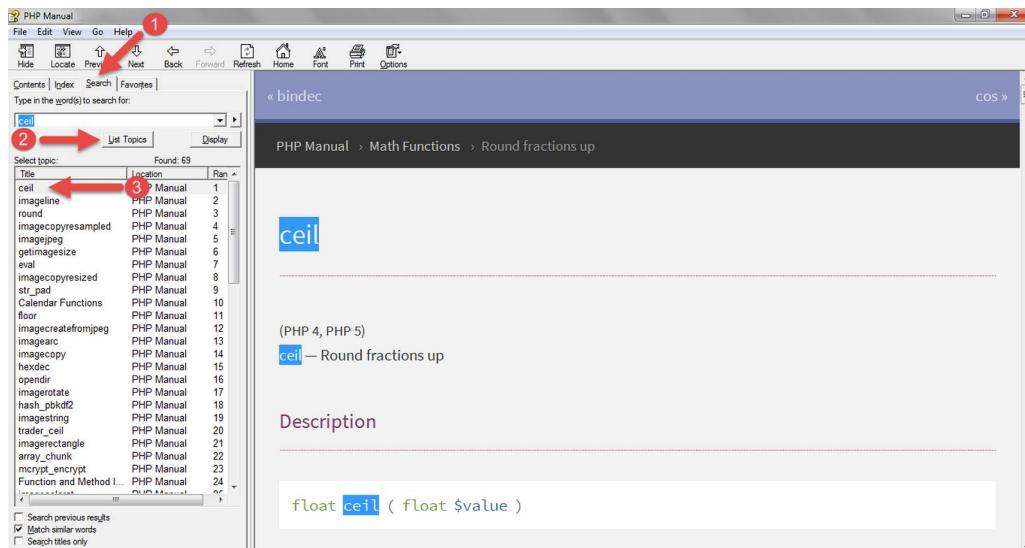
Sebagai contoh, pilih kategori **Mathematical Extensions**, lalu klik **Math – Mathematical Functions**. Kategori ini berisi fungsi-fungsi dasar yang menangani tipe data *float* dan *integer* (akan saya bahas pada bab selanjutnya).



Gambar: Mathematical Functions di PHP Manual

Cara lain menggunakan PHP Manual (yang sering saya gunakan) adalah menggunakan pencarian **index** atau **search**. Perbedaannya, pencarian *index* lebih cepat dan akurat, selama kita persis hafal nama fungsi yang ingin dicari. Namun untuk fungsi yang kurang ingat/hanya hafal sebagian namanya, pakai fitur *search*.

Misalkan anda menemukan kode yang memuat fungsi `ceil()` dan ingin tahu apa sebenarnya fungsi `ceil()` ini. Silahkan klik menu “*search*”, lalu ketik `ceil` dan klik “*list topic*”. Sesaat kemudian akan tampak hasil pencarian. Klik untuk melihat penjelasan tentang fungsi `ceil()`.



Gambar: Menggunakan fitur search di PHP Manual versi chm

## 11.4 Membaca Format Penulisan Function di PHP Manual

PHP Manual menjelaskan cara penggunaan sebuah function secara terstruktur. Sebagai contoh, mari kita lihat fungsi **ceil()**.

Penjelasan tentang fungsi *ceil()* terbagi ke dalam beberapa bagian (bisa dilihat pada halaman setelah ini). Bagian paling atas terdapat judul fungsi, yakni **ceil**. Setelahnya adalah (*PHP 4*, *PHP 5*), ini berarti fungsi *ceil()* sudah tersedia di PHP 4 dan PHP 5. Walaupun tidak dinyatakan, hampir seluruh fungsi di PHP 5 juga tersedia di PHP 7.

Di bagian *Description* ditulis sebagai berikut:

```
float ceil ( float $value )
```

Format penulisan seperti ini mengikuti aturan dari PHP Manual:

```
returned_type function_name ( parameter_type parameter_name )
```

Tanda *float* paling awal, menjelaskan bahwa nilai kembalian (*return value*) dari fungsi **ceil()** adalah ‘sesuatu’ dengan tipe data *float*. Sedangkan tanda *float* sebelum argumen *\$value* menjelaskan bahwa fungsi **ceil()** membutuhkan 1 argumen bertipe *float*.

Penjelasan tentang fungsi **ceil()** ada di baris selanjutnya: “*Returns the next highest integer value by rounding up value if necessary*”. Ini bisa diartikan: *mengembalikan nilai integer terbesar berikutnya dan akan dibulatkan jika diperlukan*.

Fungsi **ceil()** sebenarnya berfungsi untuk membulatkan sebuah angka *float* ke atas. Sebagai contoh, **ceil(5.2)** akan menghasilkan 6, begitu juga dengan **ceil(5.9)** akan mengembalikan nilai 6. Penjelasan tentang parameter dan nilai kembalian dijelaskan pada bagian *Parameters* dan *Return Values*.

Jika anda kurang paham tentang penjelasannya (yang berbahasa inggris), langsung saja lihat bagian *Examples*. Disini terdapat contoh kode program yang memperlihatkan cara penggunaan fungsi **ceil()**:

```
1 <?php
2   echo ceil(4.3);    // 5
3   echo "<br>";
4   echo ceil(9.999); // 10
5   echo "<br>";
6   echo ceil(-3.14); // -3
7 ?>
```

Seperti yang terlihat, fungsi **ceil()** akan membulatkan sebuah angka ke atas.

PHP Manual › Math Functions › Round fractions up

## ceil

---

(PHP 4, PHP 5)  
ceil — Round fractions up

### Description

```
float ceil ( float $value )
```

Returns the next highest integer value by rounding up **value** if necessary.

### Parameters

**value**  
The value to round

### Return Values

**value** rounded up to the next highest integer. The return value of **ceil()** is still of type **float** as the value range of **float** is usually bigger than that of **integer**.

### Examples

Example #1 **ceil()** example

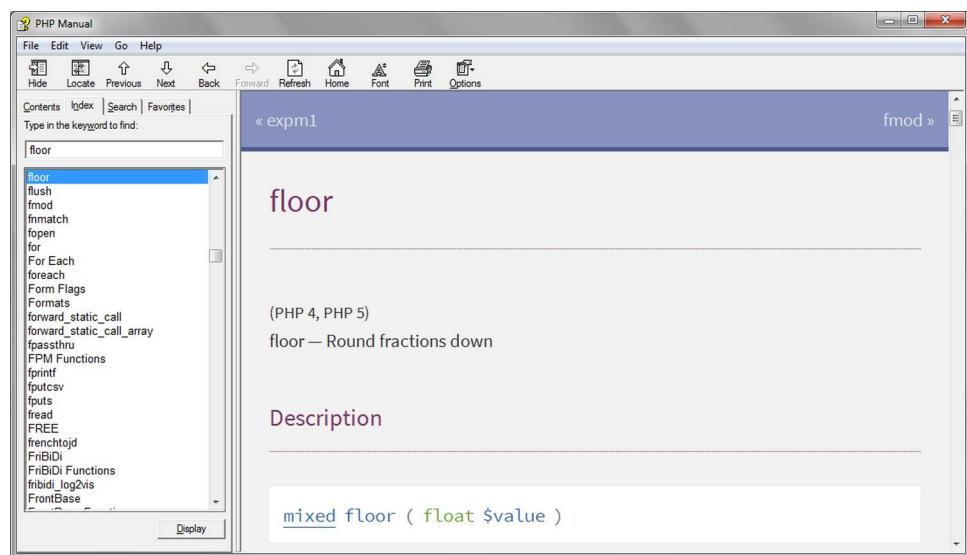
```
<?php
echo ceil(4.3);    // 5
echo ceil(9.999);  // 10
echo ceil(-3.14);  // -3
?>
```

### See Also

- [floor\(\)](#) - Round fractions down
- [round\(\)](#) - Rounds a float

### User Contributed Notes

Contoh lain, mari kita lihat penjelasan fungsi `floor()`. Anda bisa mencarinya menggunakan fitur *index* atau lihat di daftar fungsi-fungsi **Math**.



Gambar: Penjelasan fungsi `floor` dari PHP Manual

Pada bagian *Description*, fungsi ini dituliskan sebagai berikut:

```
mixed floor ( float $value )
```

**Mixed** adalah cara **PHP Manual** menyatakan bahwa nilai kembalian fungsi ini tidak hanya 1 tipe data, tapi bisa 2, 3 atau lebih. Dari penjelasannya, fungsi `floor()` akan membulatkan sebuah angka ke bawah (kebalikan dari fungsi `ceil`), sebagai contoh `floor(4.3)` akan menghasilkan 4, sedangkan `floor(9.999)` akan mengembalikan nilai 9.

Selain mengembalikan nilai integer, fungsi `floor()` juga akan menghasilkan *boolean FALSE* ketika mendapati error, seperti jika kita menginput `array: floor(array(4.2))`. Inilah yang dimaksud dengan **mixed**, dimana fungsi `floor()` bisa mengembalikan 2 jenis tipe data, yakni *float* atau *boolean*.

Berikut contoh penggunannya:

```
1 <?php
2 echo floor(4.3);      // 4
3 echo "<br>";
4 echo floor(9.999);   // 9
5 echo "<br>";
6 echo floor(-3.14);   // -4
7 echo "<br>";
8 var_dump( floor(array(4.2)) ); // bool(false)
9 ?>
```

Untuk contoh lain, mari lihat fungsi `number_format()`:

```
string number_format ( float $number [, int $decimals = 0 ] )
```

Fungsi **number\_format()** digunakan untuk menformat tampilan angka agar dikelompokkan tiap ribuan. Misalnya 12500 menjadi 12,500, 3250400 menjadi 3,250,400.

Perhatikan tanda [ dan ] di penulisan argumen kedua. Jika anda melihat cara penulisan seperti ini, artinya argumen ini bersifat opsional dan boleh tidak ditulis.

Dalam fungsi **number\_format()** diatas, jika argumen kedua tidak ditulis, nilai 0 akan digunakan (perhatikan kembali cara penulisannya: \$decimals = 0). Argumen kedua ini digunakan untuk menentukan berapa banyak angka di belakang koma.

Berikut contoh penggunaan fungsi **number\_format()**:

```
1 <?php
2     echo number_format(12500);           // 12,500
3     echo "<br>";
4     echo number_format(3250400);         // 3,250,400
5     echo "<br>";
6     echo number_format(3250400,2);       // 3,250,400.00
7     echo "<br>";
8     echo number_format(3.3333333,3);    // 3.333
9 ?>
```

Sampai disini, saya yakin anda sudah semakin paham dan bisa membaca pola penulisan function yang ada di **PHP Manual**.

Sebagai contoh terakhir, berikut format penulisan fungsi **strpos()**:

```
mixed strpos ( string $haystack , mixed $needle [, int $offset = 0 ] )
```

Terlepas dari untuk apa fungsi *strpos()* ini, perhatikan format penulisannya. Fungsi *strpos()* membutuhkan 3 argumen, dengan argumen ketiga bersifat opsional. Fungsi *strpos()* mengembalikan tipe data *mixed*, yang berarti bisa 2 atau lebih tipe data. Argumen pertama harus bertipe data *string*, argumen kedua *mixed*, dan argumen ketiga *integer*. Jika argumen ketiga tidak ditulis, nilai 0 akan digunakan.

Sepanjang sisa pembahasan dalam buku ini saya akan menampilkan format penulisan seperti ini. Sehingga anda juga akan terbiasa dan bisa memahami sendiri fungsi-fungsi dari **PHP Manual**.

---

Dalam 3 bab selanjutnya, saya akan membahas berbagai fungsi-fungsi bawaan PHP yang berasal dari **PHP Manual**. Kita akan mulai dengan fungsi-fungsi matematika di dalam PHP.

# 12. Integer & Float Function PHP

Ketika saya mengakses *PHP Manual* versi terbaru di awal Januari 2016, PHP menyediakan 49 fungsi yang berkaitan dengan angka *integer* & *float*, atau bisa juga disebut dengan fungsi-fungsi matematika. Dalam bab ini saya akan membahas beberapa diantaranya.

Jika anda berencana membuat aplikasi yang berhubungan dengan pemrosesan angka, saya sangat sarankan untuk membaca seluruh fungsi matematika PHP yang berada di bagian **Math Functions** di *PHP Manual*.



Di dalam PHP Manual, **Math Functions** bisa diakses dari *Function Reference* -> *Mathematical Extensions* -> *Math* -> *Math Functions*

## 12.1 Function abs()

Format penulisan:

```
number abs ( mixed $number )
```

Fungsi **abs()** mengembalikan nilai absolut dari sebuah angka. Angka negatif akan menjadi positif, angka positif tetap positif. Fungsi ini akan mengembalikan tipe data **number**, ini berarti hasilnya berupa angka (bisa salah satu dari *integer* atau *float*).

Fungsi **abs()** membutuhkan 1 argumen, yakni angka yang ingin diproses. Berikut contohnya:

```
1 <?php
2   echo abs(99);           // 99
3   echo "<br>";
4   echo abs(-99);          // 99
5   echo "<br>";
6   echo abs(-13.78345);    // 13.78345
7 ?>
```

## 12.2 Function ceil() dan floor()

Format penulisan:

```
float ceil ( float $value )
mixed floor ( float $value )
```

Fungsi **ceil()** dan **floor()** sama-sama digunakan untuk membulatkan angka *float*. Bedanya, fungsi **ceil()** akan membulatkan ke atas, sedangkan fungsi **floor()** akan membulatkan kebawah. Kedua fungsi ini membutuhkan 1 argumen, yakni angka yang akan diproses. Berikut contohnya:

```
1 <?php
2     echo ceil(3.4);           // 4
3     echo "<br>";
4     echo ceil(3.91);          // 4
5     echo "<br>";
6     echo ceil(-29.33);        // -29
7     echo "<br>";
8     echo floor(3.4);          // 3
9     echo "<br>";
10    echo floor(3.91);         // 3
11    echo "<br>";
12    echo floor(-29.33);       // -30
13 ?>
```

## 12.3 Function round()

Format penulisan:

```
float round ( float $val [, int $precision = 0
                      [, int $mode = PHP_ROUND_HALF_UP ]] )
```

Fungsi **round()** mirip dengan **ceil()** dan **floor()**, tetapi angka yang menjadi argumennya akan dibulatkan ke bilangan integer terdekat.

Fungsi **round()** memiliki 2 argumen opsional. Argumen opsional pertama untuk menentukan berapa digit ketelitian (*precision*), sedangkan argumen opsional kedua untuk menentukan bagaimana proses pembulatan dilakukan jika angka yang dibulatkan sama-sama kuat.

Berikut contoh penggunaan fungsi **round()** dengan 1 atau 2 argumen:

```

1 <?php
2 echo round(3.4);           // 3
3 echo "<br>";
4 echo round(3.5);           // 4
5 echo "<br>";
6 echo round(3.6);           // 4
7 echo "<br>";
8 echo round(3.6, 0);         // 4
9 echo "<br>";
10 echo round(1.95583, 2);    // 1.96
11 echo "<br>";
12 echo round(1241757, -3);   // 1242000
13 echo "<br>";
14 echo round(5.045, 2);       // 5.05
15 echo "<br>";
16 echo round(5.055, 2);       // 5.06
17 ?>

```

Bagaimana dengan argumen ke-3? Argumen ini digunakan untuk mengatur bagaimana hasil fungsi **round()** jika menemukan pembulatan untuk angka seperti 3.5, apakah dibulatkan ke angka 3 atau 4?

Argumen opsional ini bisa diisi dengan salah satu dari 4 konstanta:

- PHP\_ROUND\_HALF\_UP: Angka akan di bulatkan ke atas, 1.5 akan menjadi 2, sedangkan -1.5 akan menjadi -2.
- PHP\_ROUND\_HALF\_DOWN: Angka akan di bulatkan ke bawah, 1.5 akan menjadi 1, sedangkan -1.5 akan menjadi -1.
- PHP\_ROUND\_HALF\_EVEN: Angka akan di bulatkan ke bilangan genap
- PHP\_ROUND\_HALF\_ODD: Angka akan di bulatkan ke bilangan ganjil terdekat.

Berikut contoh fungsi **round()** dengan 3 argumen:

```

1 <?php
2 echo round(9.5, 0, PHP_ROUND_HALF_UP);    // 10
3 echo "<br>";
4 echo round(9.5, 0, PHP_ROUND_HALF_DOWN);   // 9
5 echo "<br>";
6 echo round(9.5, 0, PHP_ROUND_HALF_EVEN);   // 10
7 echo "<br>";
8 echo round(9.5, 0, PHP_ROUND_HALF_ODD);    // 9
9 echo "<br>";
10
11 echo round(8.5, 0, PHP_ROUND_HALF_UP);     // 9
12 echo "<br>";

```

```

13 echo round(8.5, 0, PHP_ROUND_HALF_DOWN); // 8
14 echo "<br>";
15 echo round(8.5, 0, PHP_ROUND_HALF_EVEN); // 8
16 echo "<br>";
17 echo round(8.5, 0, PHP_ROUND_HALF_ODD); // 9
18 ?>

```

## 12.4 Function deg2rad()

Format penulisan:

```
float deg2rad ( float $number )
```

Fungsi **deg2rad()** digunakan untuk mengkonversi nilai derajat ke radian. Jika anda tidak mengantuk saat pelajaran trigonometri,  $360^{\circ}$  sama dengan **2 pi radian**. Satuan radian banyak digunakan pada fungsi-fungsi PHP yang berkaitan dengan trigonometri. Fungsi ini membutuhkan 1 argumen, berupa angka yang ingin diproses.

Berikut contoh penggunaan fungsi **deg2rad()**:

```

1 <?php
2     echo deg2rad(45); // 0.785398163397
3     echo "<br>";
4     echo deg2rad(90); // 1.5707963267949
5     echo "<br>";
6     echo deg2rad(180); // 3.1415926535898
7     echo "<br>";
8     echo deg2rad(360); // 6.2831853071796
9 ?>

```

## 12.5 Function sin(), cos(), tan(), acos(), asin(), dan atan()

Format penulisan:

```

float sin ( float $arg )
float cos ( float $arg )
float tan ( float $arg )
float asin ( float $arg )
float acos ( float $arg )
float atan ( float $arg )

```

Ke-6 fungsi-fungsi ini digunakan untuk operasi trigonometri. Jika anda tidak tahu kegunaannya, maka fungsi ini bukanlah untuk anda. Salah satu hal yang perlu dicatat adalah, fungsi ini harus diinput dengan satuan radian, bukan derajat.

Jika butuh inputan dalam satuan derajat, pakai fungsi `deg2rad()` sebagai perantara. Berikut contohnya:

```

1 <?php
2     echo sin(0.785398163397);           // 0.70710678118623
3     echo "<br>";
4     echo sin(deg2rad(45));               // 0.70710678118623
5     echo "<br>";
6     echo cos(deg2rad(60));               // 0.5
7     echo "<br>";
8     echo tan(deg2rad(30));               // 0.57735026918963
9     echo "<br>";
10    echo asin(0.70710678118623);        // 0.785398163397
11    echo "<br>";
12    echo acos(0.5);                     // 1.0471975511966
13    echo "<br>";
14    echo atan(0.57735026918963);        // 0.5235987755983
15 ?>

```

## 12.6 Function `pow()`

Format Penulisan:

```
number pow ( number $base , number $exp )
```

Fungsi `pow()` digunakan untuk mencari hasil pangkat sebuah bilangan. Fungsi ini memerlukan 2 argumen. Argumen pertama berupa angka dasar, sedangkan argumen kedua untuk pangkat. Sebagai contoh,  $2^5$  ditulis sebagai `pow(2,5)`.

Berikut contoh penggunaan fungsi `pow()`:

```

1 <?php
2     echo pow(2,5);           // 32
3     echo "<br>";
4     echo pow(9,3);           // 729
5     echo "<br>";
6     echo pow(10,5);          // 100000
7     echo "<br>";
8     echo pow(1,200);          // 1
9 ?>

```

## 12.7 Function `log()` dan `log10()`

Format penulisan:

```
float log ( float $arg [, float $base = M_E ] )
float log10 ( float $arg )
```

Saya sengaja menyatukan pembahasan tentang fungsi **log** dan **log10**, karena ada 2 versi fungsi logaritma di PHP. Besar kemungkinan fungsi yang anda butuhkan adalah **log10()**, sebab inilah fungsi logaritma dengan basis 10.

Sedangkan fungsi **log()** secara default merupakan fungsi logaritma untuk bilangan natural e. Namun fungsi ini memiliki argumen kedua yang bisa diganti dengan basis logaritma lain. Sebagai contoh **log(100,10)** sama dengan **log10(100)**, keduanya digunakan untuk mencari hasil dari  ${}^{10}\log 100$ .

Berikut contoh penggunaan fungsi **log()** dan **log10()**:

```
1 <?php
2 echo log(100);           //4.6051701859881
3 echo "<br>";
4 echo log(100,10);        //2
5 echo "<br>";
6 echo log10(100);         //2
7 echo "<br>";
8 echo log10(10000);       //4
9 ?>
```

## 12.8 Function sqrt()

Format penulisan:

```
float sqrt ( float $arg )
```

Fungsi **sqrt()** digunakan untuk mencari akar kuadrat dari sebuah bilangan. Fungsi ini membutuh 1 arguman, yakni angka yang akan diproses. Berikut contohnya:

```
1 <?php
2 echo sqrt(25);           // 5
3 echo "<br>";
4 echo sqrt(121);          // 11
5 echo "<br>";
6 echo sqrt(1000000);       // 1000
7 ?>
```

Bagaimana dengan akar pangkat 3 keatas? Di dalam PHP tidak ada fungsi khusus untuk ini, tapi sebenarnya anda bisa menggunakan fungsi yang telah kita pelajari, yakni **pow()**.

Jika anda masih ingat, fungsi **pow()** membutuhkan 2 argumen, yakni angka dasar dan besar pangkat. Jika argumen pangkat ini ditulis sebagai pecahan, hasilnya akan menjadi akar. Sebagai contoh, **sqrt(25)** sama dengan **pow(25,1/2)**.

Berikut contoh penggunaan fungsi **pow()** untuk mencari akar:

```

1 <?php
2 echo sqrt(25);           // 5
3 echo "<br>";
4 echo pow(25,1/2);        // 5
5 echo "<br>";
6 echo pow(343,1/3);       // 7
7 echo "<br>";
8 echo pow(256,1/8);       // 2
9 ?>

```

## 12.9 Function min() dan max()

Format penulisan:

```

mixed min ( array $values )
mixed max ( array $values )

```

Fungsi **min()** dan **max()** digunakan untuk mencari nilai tertinggi dari sebuah array. Array ini diinput sebagai argumen satu-satunya fungsi **min()** dan **max()**. Berikut contohnya:

```

1 <?php
2 $arrayku=array(3,6,9,2,4,90,130,44);
3 echo max($arrayku); // 130
4 echo "<br>";
5 echo min($arrayku); // 2
6 ?>

```

## 12.10 Function rand()

Format penulisan:

```

int rand ( void )
int rand ( int $min , int $max )

```

Fungsi **rand()** digunakan untuk menghasilkan angka acak. Jika anda sedang membuat aplikasi games, angka acak dibutuhkan untuk membuat aplikasi menjadi lebih menarik. Fungsi ini memiliki 2 cara penulisan.

Jika ditulis tanpa argumen, fungsi **rand()** akan menghasilkan angka acak antara 0 hingga 32767. Nilai 32767 ini didapat dari fungsi **getrandmax()**, dan bisa berbeda-beda antara tiap komputer.

Jika ditulis dengan 2 argumen, fungsi **rand()** akan menghasilkan angka acak antara kedua argumen tersebut.

Berikut contoh penggunaan fungsi **rand()**:

```
1 <?php
2 // menampilkan nilai maksimal fungsi rand() jika tanpa argumen
3 echo getrandmax(); // 32767
4 echo "<br>";
5 echo rand(); // 3157
6 echo "<br>";
7 echo rand(); // 22122
8 echo "<br>";
9
10 // angka acak dengan batas nilai
11 echo rand(0,1); // 1
12 echo "<br>";
13 echo rand(0,100); // 89
14 echo "<br>";
15
16 //untuk membuat angka acak pecahan
17 echo rand(0,100)/100; // 0.57
18 ?>
```

Jika anda menjalankan kode program diatas, hasilnya akan selalu berbeda, karena angka acak akan digenerate setiap kali dijalankan. Yang perlu menjadi perhatian, angka acak yang dihasilkan adalah angka bulat, bukan *float*. Fungsi **rand(0,1)** hanya menghasilkan 2 kemungkinan, yakni antara 0 atau 1.

Bagaimana jika kita ingin menghasilkan angka acak yang berbentuk pecahan (*float*)? Cukup bagi hasilnya dengan pecahan, seperti contoh terakhir: **rand(0,100)/100**.

---

Dalam bab ini saya membahas beberapa fungsi bawaan PHP yang berhubungan dengan pemrosesan angka. Berikutnya, kita akan masuk ke fungsi-fungsi PHP untuk memproses tipe data string.

# 13. String Function PHP

Pengolahan data string bisa jadi merupakan hal yang paling sering kita lakukan. Mulai dari mengubah huruf besar menjadi huruf kecil, melakukan pencarian, pemotongan string, menformat tampilan, hingga memproses hasil inputan form, semuanya melibatkan string.

Dengan banyaknya hal yang dilakukan untuk string, PHP menyediakan 98 string function. Saya akan membahas beberapa diantaranya.



Di dalam PHP Manual, **String Function** bisa diakses dari *Function Reference -> Text Processing -> Strings -> String Functions*

## 13.1 Function strtolower() dan strtoupper()

Format penulisan:

```
string strtolower ( string $string )
string strtoupper ( string $string )
```

Fungsi **strtolower()** digunakan untuk menkonversi teks string menjadi huruf kecil, sedangkan fungsi **strtoupper()** untuk mengubah string menjadi huruf besar. Kedua fungsi ini hanya membutuhkan 1 argumen, yakni teks yang akan diproses. Berikut contohnya:

```
1 <?php
2   $kalimat="Sedang belajar PHP dari PHP Uncover";
3
4   echo strtolower($kalimat); // sedang belajar php dari php uncover
5   echo "<br>";
6   echo strtoupper($kalimat); // SEDANG BELAJAR PHP DARI PHP UNCOVER
7 ?>
```

Kedua fungsi ini sering digunakan untuk menyeragamkan data, misalnya kita bisa membuat aturan bahwa seluruh *username* harus menggunakan huruf kecil. Sebelum diproses, hasil inputan user bisa dikonversi dulu dengan fungsi **strtolower()**.

## 13.2 Function ucfirst() dan lcfirst()

Format penulisan:

```
string ucfirst ( string $str )
string lcfirst ( string $str )
```

Fungsi **ucfirst()** digunakan untuk membuat karakter pertama dari string menjadi huruf besar, sedangkan fungsi **lcfirst()** digunakan untuk membuat karakter pertama menjadi huruf kecil. Berikut contohnya:

```
1 <?php
2   $kalimat="sedang dalam perjalanan menjadi seorang web programmer";
3   echo ucfirst($kalimat);
4   // Sedang dalam perjalanan menjadi seorang web programmer
5   echo "<br>";
6
7   $kalimat="Sedang dalam perjalanan menjadi seorang web programmer";
8   echo lcfirst($kalimat);
9   // sedang dalam perjalanan menjadi seorang web programmer
10 ?>
```

## 13.3 Function strlen()

Format penulisan:

```
int strlen ( string $string )
```

Fungsi **strlen()** digunakan untuk mencari panjang dari sebuah string, termasuk karakter *white-space* seperti spasi. Fungsi ini sangat berguna dalam pemrosesan string. Berikut contohnya:

```
1 <?php
2   $a="abcde";
3   echo strlen($a); //5
4   echo "<br>";
5
6   $b=" ab c d  e";
7   echo strlen($b); //10
8   echo "<br>";
9
10  $kalimat="Sedang dalam perjalanan menjadi seorang web programmer";
11  echo strlen($kalimat); //54
12  echo "<br>";
13
14  $kalimat="Sedang belajar PHP dari PHP Uncover";
15  echo strlen($kalimat); //35
16 ?>
```

## 13.4 Function trim(), ltrim() dan rtrim()

Format penulisan

```
string trim ( string $str [, string $character_mask = " \t\n\r\0\x0B" ] )
string ltrim ( string $str [, string $character_mask ] )
string rtrim ( string $str [, string $character_mask ] )
```

Fungsi **trim()**, **ltrim()** dan **rtrim()** digunakan untuk menghapus spasi atau karakter lain dari kedua sisi string (sisi kiri dan sisi kanan), sisi kiri saja atau sisi kanan saja. Ketiga fungsi ini memiliki argumen opsional yang bisa diisi dengan karakter apa yang ingin dihapus. Secara default, karakter yang dihapus adalah karakter *whitespace* seperti spasi.

Berikut contohnya:

```
1 <?php
2   $a=" abcd   ";
3   echo strlen($a);           // 8
4   echo "<br>";
5
6   $trim_a = trim($a);
7   echo $trim_a;             // 'abcd';
8   echo "<br>";
9   echo strlen($trim_a);    // 4
10  echo "<br>";
11
12  $ltrim_a = ltrim($a);
13  echo $ltrim_a;           // 'abcd   ';
14  echo "<br>";
15  echo "<br>";
16  echo strlen($ltrim_a);   // 7
17
18  $rtrim_a = rtrim($a);
19  echo $rtrim_a;           // ' abcd';
20  echo "<br>";
21  echo strlen($rtrim_a);   // 5
22 ?>
```

Dalam contoh diatas, saya juga menggunakan fungsi **strlen()** untuk mencari tahu berapa panjang sebenarnya dari tiap string. Jika ditulis tanpa argumen kedua, fungsi **trim()** akan menghapus karakter *whitespace* seperti spasi dari sisi kiri dan kanan. Fungsi **rtrim()** akan menghapus spasi dari sisi kanan saja, dan fungsi **ltrim()** dari sisi kiri saja.

Fungsi **trim** ini sangat berguna ketika ingin membandingkan hasil inputan dari form, karena bisa saja pengguna tidak sengaja menambah spasi di awal atau akhir form. Berikut contoh kasusnya:

```
1 <?php
2     $username="admin "; // string admin dan sebuah spasi!
3
4     if ($username == "admin") {
5         echo "Selamat datang admin";
6     }
7     else {
8         echo "Anda bukan admin";
9     }
10 ?>
```

Hasil dari kode diatas adalah: “Anda bukan admin”. Ini terjadi karena ada sebuah spasi di ujung string `$username`. Karakter spasi ini biasanya ditambahkan ketika pengguna tidak sengaja menekan tombol spasi pada saat mengisi form.

Daripada membuat pengguna bingung, kita bisa membantunya dengan menghapus spasi dengan fungsi `trim()` sebelum operasi perbandingan dilakukan. Berikut revisi kode program diatas:

```
1 <?php
2     $username="admin "; // string admin dan sebuah spasi!
3
4     if (trim($username) == "admin") {
5         echo "Selamat datang admin";
6     }
7     else {
8         echo "Anda bukan admin";
9     }
10 ?>
```

Sekarang, hasil kodennya adalah “Selamat datang admin”.

Jika kita menambahkan argumen kedua untuk fungsi `trim()`, karakter inilah yang akan dihapus. Berikut contohnya:

```
1 <?php
2     $a=__1234__;
3
4     echo trim($a,'_'); // '1234';
5     echo "<br>";
6     echo ltrim($a,'_'); // '1234_';
7     echo "<br>";
8     echo rtrim($a,'_'); // '__1234 ';
9     echo "<br>";
10
11     $b="##-#Belajar PHP---##--";
12
```

```

13 echo trim($b, '#-'); // 'Belajar PHP';
14 echo "<br>";
15 echo ltrim($b, '#-'); // 'Belajar PHP---##---';
16 echo "<br>";
17 echo rtrim($b, '#-'); // '##-#Belajar PHP ';
18 ?>

```

Untuk contoh variabel \$b, saya menginput 2 karakter ‘#-’ ke dalam argumen kedua fungsi `trim()`. Ini berarti saya ingin menghapus karakter ‘#’ dan ‘-’ yang ada di awal dan akhir string.

## 13.5 Function `str_pad()`

Format penulisan:

```
string str_pad ( string $input , int $pad_length [, string $pad_string = " "
[, int $pad_type = STR_PAD_RIGHT ]] )
```

Fungsi `str_pad()` adalah kebalikan dari fungsi `trim()`. Kali ini kita bisa menambahkan beberapa karakter di awal string, akhir string atau keduanya. Fungsi ini membutuhkan 2 argumen utama serta 2 argumen opsional.

Argumen pertama digunakan untuk menampung string yang akan diproses. Argumen kedua untuk menentukan berapa panjang string yang diinginkan. Jika panjang string kurang dari jumlah argumen kedua ini, string akan ditambah dengan karakter di argumen ketiga (sebagai default, karakter ini adalah spasi “ ”). Argumen keempat menentukan dimana karakter akan ditambah, apakah disisi kiri, kanan, atau keduanya (nilai default berada di kanan).

Supaya mudah dipahami, mari kita lihat contohnya:

```

1 <?php
2   $a="a";
3   echo str_pad($a,8); // 'a           '
4   echo "<br>";
5
6   $a="abc";
7   echo str_pad($a,8); // 'abc          '
8   echo "<br>";
9
10  $a="abcde";
11  echo str_pad($a,8); // 'abcde         '
12  echo "<br>";
13
14  $a="abcdefghijkl";
15  echo str_pad($a,8); // 'abcdefghijkl'
16 ?>

```

Perhatikan bagaimana fungsi `str_pad($a,8)` bekerja. Ini artinya: ambil string `$a`, jika panjang string `$a` kurang dari 8, tambahkan beberapa spasi di sisi kanan hingga panjang totalnya menjadi 8. Namun jika string `$a` sudah lebih dari 8, jangan lakukan perubahan apapun.

Jika anda menjalankan kode diatas di web browser, perbedaannya tidak akan terlihat. Ini karena web browser otomatis menghapus spasi ke 2 dan seterusnya.

Agar perubahannya lebih jelas, mari kita gunakan argumen ketiga:

```
1 <?php
2     $a="a";
3     echo str_pad($a,8,'#');      // 'a#####'
4     echo "<br>";
5
6     $a="abc";
7     echo str_pad($a,8,'#');      // 'abc#####'
8     echo "<br>";
9
10    $a="abcde";
11    echo str_pad($a,8,'#');      // 'abcde###'
12    echo "<br>";
13
14    $a="abcdefghij";
15    echo str_pad($a,8,'#');      // 'abcdefghij'
16 ?>
```

Argumen ketiga dari fungsi `str_pad()` digunakan untuk menentukan karakter apa yang akan ditambahkan ke dalam string. Fungsi `str_pad($a,8,'#')` berarti: ambil string `$a`, jika panjang string `$a` kurang dari 8, tambahkan beberapa karakter '#' disisi kanan hingga panjang totalnya menjadi 8.

Sekarang, anda bisa melihat perubahannya. Selama string `$a` kurang dari 8 karakter, tanda '#' akan ditambahkan. Fitur seperti ini sering digunakan untuk membuat panjang string menjadi seragam, misalnya untuk ID dari sebuah produk, nomor urut, dll.

Argumen ke-4 bisa digunakan untuk mengatur disisi mana penambahan karakter dilakukan. Kita bisa memilih dari 1 dari 3 konstanta: `STR_PAD_RIGHT` (penambahan di sisi kanan), `STR_PAD_LEFT`(penambahan di sisi kiri), atau `STR_PAD_BOTH` (penambahan di kedua sisi). Berikut contohnya:

```
1 <?php
2   $a="123";
3   echo str_pad($a,8,'0',STR_PAD_LEFT); // '00000123'
4   echo "<br>";
5
6   $a="12345";
7   echo str_pad($a,8,'0',STR_PAD_LEFT); // '00012345'
8   echo "<br>";
9
10  $a="12345789";
11  echo str_pad($a,8,'0',STR_PAD_LEFT); // '12345789'
12 ?>
```

Kali ini saya menggunakan karakter ‘0’ dan `STR_PAD_LEFT` kedalam fungsi `str_pad()`. Hasilnya, angka akan memiliki karakter 0 di depannya. Hasil seperti ini sangat sering digunakan agar tampilan angka menjadi sama panjang (dengan tambahan karakter 0 di kanan angka, seperti 001, 002, 003, dst).



Jika anda ingat, kita pernah mempelajari cara menampilkan *menu dropdown* untuk membuat tanggal 1-31 dengan tag `<select>`. Ini saya bahas ketika kita mempelajari perulangan FOR. Bisakah anda mengubah kode tersebut agar tanggal yang tampil harus 2 digit? Yakni 01, 02, 03, ...dst hingga 31? Tipsnya, gunakan fungsi `str_pad($tanggal, 2,'0',STR_PAD_LEFT)`.

## 13.6 Function `strip_tags()`

Format penulisan:

```
string strip_tags ( string $str [, string $allowable_tags ] )
```

Fungsi `strip_tags()` digunakan untuk menghapus tag-tag HTML dari sebuah string. Biasanya fungsi ini digunakan agar pengguna tidak bisa menginput kode-kode aneh HTML ke dalam form. Saya akan kembali membahas masalah ini ketika kita membahas cara memproses form dengan PHP.

Fungsi `strip_tags()` juga menyediakan argumen kedua yang bisa diisi dengan tag HTML yang ingin ‘dilewatkan’. Berikut contohnya:

```

1 <?php
2   $komentar=<b>Nice info gan</b>, <br> <i>kunjungi blog ane ya.. </i>
3           <a href='http://www.situssspam.com'>www.situssspam.com</a>";
4   echo $komentar;
5   // <b>Nice info gan</b>, <br> <i>kunjungi blog ane ya.. </i>
6   // <a href='http://www.situssspam.com'>www.situssspam.com</a>
7   echo "<br>";
8
9   echo strip_tags($komentar);
10  // Nice info gan,  kunjungi blog ane ya..
11  // www.situssspam.com
12  echo "<br>";
13
14  echo strip_tags($komentar,"<br><i>");
15  // Nice info gan, <br> <i>kunjungi blog ane ya.. </i>
16  // www.situssspam.com
17 ?>

```

## 13.7 Function str\_shuffle()

Format penulisan:

```
string str_shuffle ( string $str )
```

Jika kita punya fungsi `rand()` untuk menghasilkan angka acak, untuk menghasilkan string acak, bisa menggunakan fungsi `str_shuffle()`. Fungsi ini akan mengacak urutan karakter string. Dalam penerapannya, ini bisa digunakan untuk membuat password acak. Berikut contohnya:

```

1 <?php
2   $a="aidueo";
3   echo str_shuffle($a); // eioua
4   echo "<br>";
5
6   $a="abcde";
7   echo str_shuffle($a); // cdaeb
8   echo "<br>";
9
10  $a="SedangBelajarPHP";
11  echo str_shuffle($a); // ePagBaSdrnjHPle
12  echo "<br>";
13
14  $a="!#$%*&(_)_+";
15  echo str_shuffle($a); // *$&(_)%#+!
16 ?>

```

## 13.8 Function number\_format()

Format penulisan:

```
string number_format ( float $number [, int $decimals = 0 ] )
string number_format ( float $number , int $decimals = 0 ,
                      string $dec_point = "." , string $thousands_sep = "," )
```

Penggunaan fungsi **number\_format()** sudah saya bahas sekilas pada bab tentang *PHP Manual*. Fungsi ini sebenarnya memiliki 2 cara penulisan. Yang pertama adalah dengan 1 atau 2 argumen. Argumen pertama untuk menginput angka asal, dan argumen kedua untuk menentukan berapa banyak angka desimal di belakang koma.

Cara penulisan kedua yakni dengan 4 argumen. Selain 2 argumen pertama, argumen ketiga dan keempat bisa digunakan untuk menukar karakter pemisah ribuan dan desimal. Sebagai contoh, untuk membuat format angka dengan aturan penulisan Indonesia, saya bisa membuatnya sebagai berikut:

```
1 <?php
2   echo number_format(39999.99,0,',','.');
3   echo "<br>";
4   echo number_format(39999.99,2,',','.');
5   echo "<br>";
6   echo number_format(39999.99,4,',','.');
7   echo "<br>";
8   echo number_format(1499999,2,',','.');
9
10  echo "<br>";
11  echo number_format(135000,2,' ',' ');
12  echo "<br>";
13 ?>
```

Dengan menggunakan fungsi **number\_format()**, kita bisa menampilkan angka nominal uang dengan lebih rapi, seperti contoh berikut:

```
1 <?php
2   $harga = 354500.89;
3   $harga_format = "Rp. ".number_format($harga,2,',','.');
4
5   echo "Harga barang ini adalah $harga_format";
6   // Harga barang ini adalah Rp. 354.500,89;
7 ?>
```

## 13.9 Function substr()

Format penulisan:

```
string substr ( string $string , int $start [, int $length ] )
```

Fungsi **substr()** berfungsi untuk mengambil sebagian karakter dari sebuah string. Fungsi ini juga cukup sering digunakan.

Fungsi **substr()** bisa diisi dengan 3 buah argumen: argumen pertama berupa string sumber, argumen kedua adalah posisi awal pengambilan, dan argumen ketiga (yang bersifat opsional) bisa diisi dengan berapa banyak karakter yang ingin diambil. Jika argumen ketiga ini tidak dituliskan, artinya akan mengambil seluruh string hingga akhir.

Yang perlu menjadi catatan, posisi perhitungan string dimulai dari 0, bukan 1. Berikut contohnya:

```
1 <?php
2   $kalimat="Belajar PHP dari PHP Uncover";
3
4   echo substr($kalimat,3);      // ajar PHP dari PHP Uncover
5   echo "<br>";
6   echo substr($kalimat,17);     // PHP Uncover
7   echo "<br>";
8   echo substr($kalimat,3,4);    // ajar
9   echo "<br>";
10  echo substr($kalimat,0,11);   // Belajar PHP
11 ?>
```

Fungsi **substr(\$kalimat,3)** berarti: ambil string **\$kalimat** mulai dari karakter urutan ke-3 hingga akhir string. Sedangkan fungsi **substr(\$kalimat,3,4)** berarti: ambil string **\$kalimat** mulai dari karakter urutan ke-3, sebanyak 4 karakter.

Agar lebih memahami, saya sangat sarankan untuk mencoba mengubah nilai yang ada, dan hitung darimana potongan string tersebut berasal.

Argumen kedua dan ketiga fungsi **substr()** juga bisa diisi dengan angka negatif. Jika dituliskan seperti ini, perhitungan akan dimulai dari akhir string, bukan dari awal string. Berikut contohnya:

```
1 <?php
2   $kalimat="Belajar PHP dari PHP Uncover";
3   echo substr($kalimat,-5);      // cover
4   echo "<br>";
5   echo substr($kalimat,-17);     // dari PHP Uncover
6   echo "<br>";
7   echo substr($kalimat,-20,12);   // PHP dari PHP
8   echo "<br>";
9   echo substr($kalimat,-20,-2);   // PHP dari PHP Unco
10 ?>
```

Fungsi **substr(\$kalimat,-5)** berarti: ambil string **\$kalimat**, mulai dari 5 karakter terakhir hingga akhir string. Fungsi **substr(\$kalimat,-20,-2)** berarti: ambil string **\$kalimat** mulai 20 karakter dari ujung string, lalu kurangi sebanyak 2 karakter dari ujung string. Sekali lagi, saya sangat sarankan untuk mengubah nilai yang ada untuk lebih memahami cara penggunaan fungsi ini.

## 13.10 Function strpos()

Format penulisan:

```
mixed strpos ( string $haystack , mixed $needle [, int $offset = 0 ] )
```

Fungsi **strpos()** digunakan untuk mencari posisi sebuah string atau karakter di dalam string lain. Fungsi ini cocok untuk operasi seperti '*find and replace*'. Fungsi **strpos()** memerlukan 3 buah argumen, dengan 1 argumen bersifat opsional.

Argumen pertama adalah string sumber, argumen kedua untuk string yang ingin dicari, dan argumen ketiga adalah posisi awal pencarian. Jika argumen ketiga tidak ditulis, pencarian akan dimulai dari awal string (posisi karakter ke-0).

Hasil dari fungsi **strpos()** adalah posisi karakter yang dicari. Posisi ini dimulai dari 0 untuk karakter pertama. Berikut contohnya:

```
1 <?php
2   $kalimat="Belajar PHP dari PHP Uncover";
3   $cari="PHP";
4   echo strpos($kalimat,$cari); // 8
5 ?>
```

Angka 8 hasil fungsi **strpos(\$kalimat,\$cari)**, berarti string **\$cari** ada di posisi ke 8 dari string **\$kalimat**.

Jika anda perhatikan, string "PHP" sebenarnya ada di 2 tempat. Bagaimana untuk mencari posisi kedua ini? Kita bisa menambahkan argumen ketiga agar fungsi **strpos()** mulai dari posisi tertentu. Berikut contohnya:

```
1 <?php
2   $kalimat="Belajar PHP dari PHP Uncover";
3   $cari="PHP";
4   echo strpos($kalimat,$cari,9); // 17
5   echo "<br>";
6   var_dump(strpos($kalimat,"CSS")); // bool(false)
7 ?>
```

Fungsi **strpos(\$kalimat,\$cari,9)** berarti saya ingin proses pencarian dimulai dari posisi ke 9. Dengan demikian, hasilnya adalah 17, yang menunjukkan string "PHP" kedua.

Bagaimana jika fungsi **strpos()** tidak menemukan string yang dicari? Hasilnya adalah boolean FALSE, seperti contoh terkahir: **strpos(\$kalimat,"CSS")**.

## 13.11 Function str\_replace()

Format penulisan:

```
mixed str_replace ( mixed $search , mixed $replace , mixed $subject  
[ , int &$count ] )
```

Fungsi **str\_replace()** berguna untuk mengganti sebuah string atau karakter yang ada di string lain. Ini mirip dengan fitur *replace* di aplikasi text processing seperti microsoft word. Fungsi **str\_replace()** membutuhkan 4 argumen (argumen terakhir adalah opsional).

Argumen pertama diisi dengan string yang ingin dicari. Argumen kedua diisi dengan string pengganti. Argumen ketiga adalah string sumber dimana proses pencarian akan dilakukan. Berikut contoh penggunaan fungsi **str\_replace()**:

```
1 <?php  
2     $kalimat="Belajar PHP dari PHP Uncover";  
3     $cari="PHP";  
4     $ganti="CSS";  
5  
6     echo str_replace($cari,$ganti,$kalimat);  
7     // Belajar CSS dari CSS Uncover  
8 ?>
```

Dalam kode diatas, saya ingin mengganti seluruh kata “PHP” menjadi “CSS”. Fungsi ini akan mengganti seluruh karakter yang ada. Kita tidak bisa mengatur berapa banyak string yang bisa diganti.

Argumen keempat dari fungsi **str\_replace()** sebenarnya bukan untuk input, tapi tempat bagi variabel penampung seberapa banyak proses replace yang berhasil dilakukan. Berikut contohnya:

```
1 <?php  
2     $kalimat="Belajar PHP dari PHP Uncover";  
3     $cari="PHP";  
4     $ganti="CSS";  
5     $hasil=0;  
6  
7     echo str_replace($cari,$ganti,$kalimat,$hasil);  
8     // Belajar CSS dari CSS Uncover  
9     echo "<br>";  
10    echo $hasil; // 2  
11 ?>
```

Perhatikan bagaimana cara penggunaan variabel **\$hasil**. Karena fungsi **str\_replace()** berhasil menukar 2 buah string “PHP” dengan “CSS”, variabel **\$hasil** berisi angka 2. Jika fungsi **str\_replace()** berhasil menukar 99 string, variabel **\$hasil** juga akan berisi angka 99.

## 13.12 Function explode()

Format penulisan:

```
array explode ( string $delimiter , string $string [, int $limit ] )
```

Fungsi **explode()** digunakan untuk mengkonversi sebuah string menjadi array. Fungsi ini membutuhkan 3 argumen dengan 1 argumen opsional.

Argumen pertama diisi dengan karakter pemisah. Karakter ini bisa berupa titik, koma, atau karakter lain. Argumen kedua diisi dengan string yang ingin dikonversi. Berikut contoh penggunaan fungsi **explode()** dengan 2 argumen:

```
1 <?php
2   $sumber="HTML,CSS,PHP,MySQL,JavaScript";
3
4   $array_hasil=explode(",",$sumber);
5   print_r($array_hasil);
6
7   //Array ([0] => HTML [1] => CSS [2] => PHP [3] => MySQL [4] => JavaScript)
8 ?>
```

Dalam contoh diatas saya membuat sebuah string **\$sumber** yang terdiri dari 5 materi web programming. Diantaranya dipisahkan dengan tanda koma. Tanda koma inilah yang bisa digunakan sebagai karakter pemisah untuk fungsi **explode()**.

Kita bisa menggunakan karakter apa saja sebagai karakter pemisah, tergantung struktur string sumber, seperti contoh berikut:

```
1 <?php
2   $sumber="ab/cd/ef/gh/ij";
3
4   $array_hasil=explode("/",$sumber);
5   print_r($array_hasil);
6
7   // Array ( [0] => ab [1] => cd [2] => ef [3] => gh [4] => ij )
8 ?>
```

Kali ini saya menggunakan karakter '/' sebagai pemisah string.

Argumen ketiga dari fungsi **explode()** digunakan untuk menentukan seberapa banyak element array yang dihasilkan, berikut contohnya:

```
1 <?php
2     $sumber="HTML,CSS,PHP,MySQL,JavaScript";
3
4     $array_hasil=explode(",",$sumber,3);
5     print_r($array_hasil);
6
7     // Array (
8     // [0] => HTML
9     // [1] => CSS
10    // [2] => PHP,MySQL,JavaScript )
11 ?>
```

Karena saya menginput angka 3 sebagai argumen ketiga, array yang dihasilkan juga akan dibatasi sebanyak 3 element.

## 13.13 Function implode()

Format penulisan:

```
string implode ( string $glue , array $pieces )
```

Fungsi **implode()** merupakan kebalikan dari fungsi **explode()**. Fungsi ini digunakan untuk menyatukan element array menjadi sebuah string. Karakter penyatu string ditempatkan sebagai argumen pertama, sedangkan array yang ingin disatukan sebagai argumen kedua. Berikut contohnya:

```
1 <?php
2     $array_sumber = array("HTML", "CSS", "PHP", "MySQL", "JavaScript");
3
4     $string_hasil=implode("#",$array_sumber);
5     echo $string_hasil; // HTML#CSS#PHP#MySQL#JavaScript
6 ?>
```

---

Dalam bab ini kita sudah membahas beberapa fungsi string di dalam PHP. Jika anda berminat, bisa mempelajari fungsi-fungsi string lain dari PHP Manual. Berikutnya, saya akan masuk ke dalam fungsi-fungsi array bawaan PHP.

# 14. Array Function PHP

Array merupakan salah satu tipe data terpenting dan sering digunakan di dalam PHP. Operasi seperti pemrosesan form, menampilkan data dari database, membuat data terstruktur, semuanya melibatkan array.

**PHP Manual** menyediakan sekitar 79 fungsi bawaan untuk memproses Array. Kali ini saya akan membahas beberapa diantaranya.



Di dalam PHP Manual, **Array Function** bisa diakses dari *Function Reference -> Variable and Type Related Extensions -> Arrays -> Array Functions*.

## 14.1 Function count()

Format penulisan:

```
int count ( mixed $array_or_countable [, int $mode = COUNT_NORMAL ] )
```

Fungsi **count()** sangat sederhana, tapi cukup sering digunakan. Fungsi ini berguna untuk menghitung jumlah element di dalam suatu array. Berikut contohnya:

```
1 <?php
2     $zoo = array("kucing", "ikan", "ayam", "bebek", "sapi");
3     echo count($zoo); // 5
4 ?>
```

Angka 5 hasil dari fungsi **count()** diatas, berarti array **\$zoo** memiliki 5 element.

Argumen kedua dari fungsi ini bisa ditambahkan jika kita memiliki **array rekursif (nested array)**, yakni ada array di dalam array. Berikut contohnya:

```
1 <?php
2     $zoo =
3         array(
4             "ayam" => array("kampung", "buras", "kate", "hutan"),
5             "kucing" => array("persia", "himalaya", "anggora", "kampung", "hutan")
6         );
7     echo count($zoo); // 2
8     echo "<br>";
9     echo count($zoo, COUNT_RECURSIVE); // 11
10 ?>
```

Kali ini array `$zoo` berisi 2 array lain, yakni array `$ayam` dan array `$kucing`. Ketika saya menggunakan fungsi `count($zoo)`, hasilnya adalah 2. Ini berarti ada 2 element di dalam array `$zoo`. Walaupun tidak salah, tetapi kedua element ini adalah array yang juga terdiri dari element lain didalamnya.

Agar semua element ikut dihitung, saya bisa menambahkan konstanta `COUNT_RECURSIVE` sebagai argumen kedua dari fungsi `count()`. Hasilnya, fungsi `count($zoo,COUNT_RECURSIVE)` akan mengembalikan angka 11. Angka ini berasal dari penjumlahan 2 element array (array `$ayam` dan array `$kucing`) + 4 (jumlah element di dalam array `$ayam`) + 5 (jumlah element di dalam array `$kucing`).

Fungsi `count()` sering digunakan jika kita ingin mengakses seluruh element array menggunakan perulangan, seperti contoh berikut ini:

```

1 <?php
2   $zoo = array("kucing", "ikan", "ayam", "bebek", "sapi");
3
4   for ($i=0; $i < count($zoo); $i++) {
5     echo $zoo[$i];
6     echo "<br>";
7   }
8 ?>

```

Yang harus selalu diingat, index atau key maksimum dari sebuah array adalah hasil fungsi `count()` dikurangi 1. Ini karena index dimulai dari angka 0, bukan 1. Array `$zoo` memiliki 5 element, dimana element terakhirnya adalah `$zoo[4]`.

## 14.2 Function array\_sum()

Format penulisan:

```
number array_sum ( array $array )
```

Fungsi `array_sum()` digunakan untuk menghitung jumlah total dari seluruh element array. Fungsi ini cocok jika anda ingin membuat aplikasi matematis atau statistik. Berikut contohnya:

```

1 <?php
2   $nilai = array(98, 59, 42, 65, 87);
3   echo array_sum($nilai); // 351
4   echo "<br>";
5
6   $nilai = array(9.8, 5.9, 4.2);
7   echo array_sum($nilai); // 19.9
8   echo "<br>";
9

```

```

10 $nilai = array("anton" => 82, "rudy" => 81, "rini" => 95);
11 $nilai_rata2 = array_sum($nilai)/count($nilai);
12 echo "Nilai rata-rata siswa : $nilai_rata2";
13 //Nilai rata-rata siswa : 86
14 ?>

```

Array `$nilai` dalam contoh terakhir merupakan *associative array*. Dapat dilihat bahwa fungsi `array_sum()` bisa digunakan untuk array biasa, maupun *associative array*. Selain itu, saya bisa menghitung nilai rata-rata dengan menggunakan fungsi `array_sum()` dan `count()`.

## 14.3 Function sort()

Format penulisan:

```
bool sort ( array &$array [, int $sort_flags = SORT_REGULAR ] )
```

Sesuai dengan namanya, fungsi `sort()` bertujuan untuk mengurutkan array. Berikut contoh penggunaannya:

```

1 <?php
2 $nilai = array(98, 59, 42, 65, 87, 66, 82, 49, 99);
3 sort($nilai);
4 print_r($nilai);
5 // Array ( [0] => 42 [1] => 49 [2] => 59 [3] => 65 [4] => 66
6 // [5] => 82 [6] => 87 [7] => 98 [8] => 99 )
7 echo "<br>";
8
9 $siswa = array("andi", "gina", "joko", "santi", "rini", "rika", "joy");
10 sort($siswa);
11 print_r($siswa);
12 // Array ( [0] => andi [1] => gina [2] => joko [3] => joy
13 // [4] => rika [5] => rini [6] => santi )
14 ?>

```

Dalam contoh diatas, array `$nilai` berisi angka integer, sedangkan array `$siswa` berisi tipe data string. Dengan menggunakan fungsi `sort()`, element kedua array ini diurutkan berdasarkan penomoran dan abjad.

Selain itu, perhatikan cara penggunaan fungsi `sort()` ini. Saya tidak menampung hasilnya ke dalam variabel. Fungsi `sort()` akan langsung merubah array asli. Jika anda ingin tetap bisa mengakses isi array asli (array sebelum proses pengurutan), silahkan copy isi variabel ini ke dalam variabel lain sebelum di urutkan.

Pada format penulisan di **PHP Manual**, fungsi `sort()` ditulis dengan tambahan karakter ‘&’ di bagian argumennya, yakni “`array &$array`” pada argumen pertama. Inilah ‘tanda baca’ yang

menjelaskan bahwa fungsi **sort()** akan mengubah variabel array asli, dan bukan mengembalikan nilai array hasil pengurutan.

Fungsi **sort()** juga memiliki argumen kedua yang bersifat opsional. Argumen kedua ini berfungsi untuk menentukan bagaimana proses pengurutan dilakukan. Sebagai contoh, perhatikan kode berikut ini:

```
1 <?php
2 echo "<br>";
3 $siswa = array("siswa1", "siswa20", "siswa2", "siswa11", "siswa5");
4 sort($siswa);
5
6 print_r($siswa);
7 // Array ( [0] => siswa1 [1] => siswa11 [2] => siswa2
8 //           [3] => siswa20 [4] => siswa5 )
9 ?>
```

Proses pengurutan dilakukan tidak seperti yang ‘seharusnya’, string “siswa11” muncul sebelum string “siswa2”. Untuk mengatasi hal ini saya bisa menambahkan konstanta **SORT\_NATURAL** sebagai argumen kedua fungsi **sort()**, seperti berikut ini:

```
1 <?php
2 echo "<br>";
3 sort($siswa, SORT_NATURAL);
4 print_r($siswa);
5 // Array ( [0] => siswa1 [1] => siswa2 [2] => siswa5
6 //           [3] => siswa11 [4] => siswa20 )
7 ?>
```

Kali ini array **\$siswa** diurutkan seperti seharusnya. Kasus seperti ini memang cukup jarang. Umumnya kita hanya butuh fungsi **sort()** ‘biasa’ tanpa argumen kedua ini.

## 14.4 Function array\_rand()

Format penulisan

```
mixed array_rand ( array $array [, int $num = 1 ] )
```

Fungsi **array\_rand()** berguna untuk mengambil satu **key** acak dari sebuah array. Langsung saja kita lihat contoh penggunaannya:

```

1 <?php
2 $siswa = array("andi", "gina", "joko", "santi", "rini", "rika", "joy");
3 $key_siswa = array_rand($siswa);
4
5 echo $key_siswa;           // 5
6 echo $siswa[$key_siswa];   // rika
7 ?>

```

Jika anda menjalankan kode diatas, hasilnya akan berbeda. Ini karena nilai kembalian dari fungsi `array_rand($siswa)` selalu random (acak). Yang perlu diperhatikan adalah, fungsi `array_rand()` mengembalikan nilai key dari array asal, bukan element-nya. Jika kita ingin menampilkan element acak tersebut, harus ditulis dengan `echo $siswa[$key_siswa]`.

Dalam contoh diatas, *key* yang dikembalikan hanya 1. Fungsi `array_rand()` juga mendukung argumen kedua yang bisa digunakan untuk menentukan berapa banyak key acak yang ingin diambil. Berikut contohnya:

```

1 <?php
2 $siswa = array("andi", "gina", "joko", "santi", "rini", "rika", "joy");
3 $key_siswa = array_rand($siswa,2);
4
5 print_r($key_siswa); // Array ( [0] => 3 [1] => 4 )
6
7 echo $siswa[$key_siswa[0]]; // santi
8 echo "<br>";
9 echo $siswa[$key_siswa[1]]; // rini
10 ?>

```

Ketika fungsi `array_rand()` dipanggil dengan argumen kedua, hasil kembalinya bukan lagi sebuah *key*, tetapi *array* yang berisi *key*. Variabel `$key_siswa` berisi sebuah array dengan 2 nilai: 3 dan 4. Angka di dalam array inilah yang merupakan *key* acak yang berasal dari array `$siswa`.

## 14.5 Function shuffle()

Format penulisan:

```
bool shuffle ( array &$array )
```

Fungsi `shuffle()` digunakan untuk mengacak urutan array. Perhatikan format penulisan dasar fungsi ini, argumen pertama menggunakan tanda ‘&’ yang mirip dengan fungsi `sort()`. Ini artinya, array asal akan langsung diacak (bukan menyembalikan hasil array yang diacak). Berikut contoh penggunaan fungsi `shuffle()`:

```

1 <?php
2   $siswa = array("andi", "gina", "joko", "santi", "rini", "rika", "joy");
3   shuffle($siswa);
4
5   print_r($siswa);
6   // Array ( [0] => santi [1] => joy [2] => gina [3] => joko
7   //           [4] => rika [5] => rini [6] => andi )
8 ?>

```

## 14.6 Function array\_push()

Format penulisan:

```
int array_push ( array &$array , mixed $value1 [, mixed $... ] )
```

Fungsi **array\_push()** digunakan untuk menambah satu atau beberapa element di akhir array. Fungsi ini juga langsung mengubah array yang menjadi argumennya. Nilai kembalian dari fungsi ini adalah seberapa banyak element array saat ini. Berikut contoh penggunaan fungsi **array\_push()**:

```

1 <?php
2   $siswa = array("andi", "gina", "joko", "santi");
3   $var = array_push($siswa, "rani");
4
5   echo $var; // 5
6   print_r($siswa);
7   // Array ( [0] => andi [1] => gina [2] => joko [3] => santi [4] => rani )
8 ?>

```

Dalam contoh ini saya menambahkan sebuah string “rani” kedalam array **\$siswa**. Seperti yang terlihat, string tersebut ditambahkan di urutan terakhir array.

Perhatikan format penulisan fungsi **array\_push()**, untuk argumen kedua ditulis dengan “**mixed \$value1 [, mixed \$... ]**”. Tanda **[...]** berarti fungsi ini bisa diinput dengan jumlah argumen yang tidak dibatasi. Dalam fungsi **array\_push()** ini berarti kita bisa menginput banyak element baru sekaligus, tidak hanya 1 saja. Mari lihat contoh kode programnya:

```

1 <?php
2   $nilai = array(98, 59, 42, 65, 87, 66);
3   $var = array_push($nilai, 82, 49, 99);
4
5   echo $var; // 9
6   print_r($nilai);
7   // Array ( [0] => 98 [1] => 59 [2] => 42 [3] => 65 [4] => 87
8   //           [5] => 66 [6] => 82 [7] => 49 [8] => 99 )
9 ?>

```

Kali ini saya menambahkan 3 element baru ke dalam array `$nilai`. Bagaimana dengan 10 element? Saya tinggal membuat 10 angka sebagai argumen fungsi `array_push()`.

Apabila anda masih ingat, sebenarnya ada cara yang lebih praktis, terutama jika ingin menambahkan 1 atau 2 element:

```

1 <?php
2   $nilai = array(98, 59, 42, 65, 87, 66);
3   $nilai[] = 82;
4   $nilai[] = 49;
5   $nilai[] = 99;
6
7   print_r($nilai);
8   // Array ( [0] => 98 [1] => 59 [2] => 42 [3] => 65 [4] => 87
9   //           [5] => 66 [6] => 82 [7] => 49 [8] => 99 )
10 ?>

```

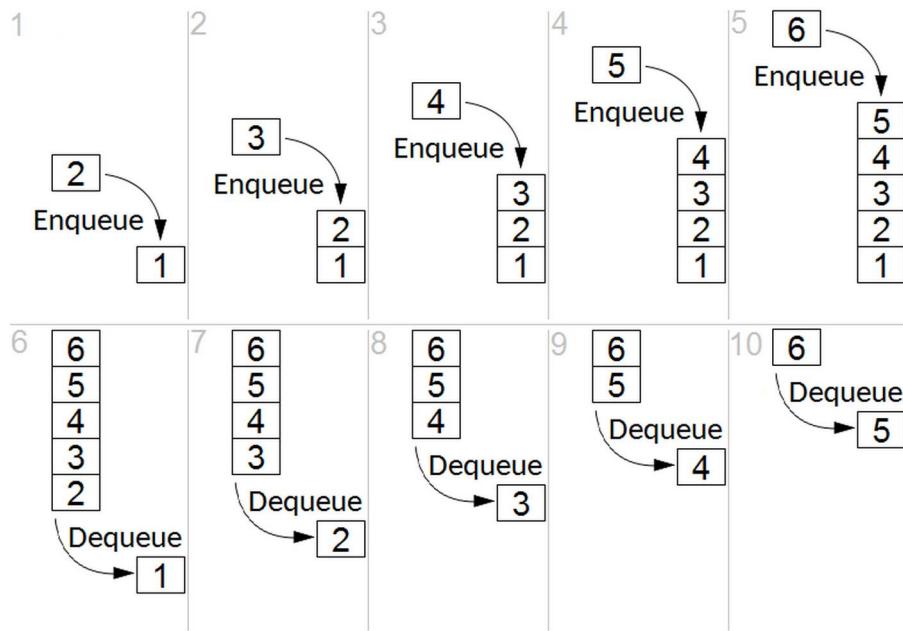
Cara penulisan seperti ini telah kita pelajari ketika membahas tipe data array. Jadi mengapa harus repot-repot menggunakan fungsi `array_push()`?

Dalam ilmu komputer, terdapat mata kuliah yang dinamakan “struktur data”. Struktur data membahas bagaimana sebuah data disusun di dalam memory komputer. Dalam “ilmu” ini, konsep array dikenal juga sebagai tumpukan (bahasa inggris: *stack*). Ketika membuat program yang cukup rumit, konsep *stack* ini bisa berguna.

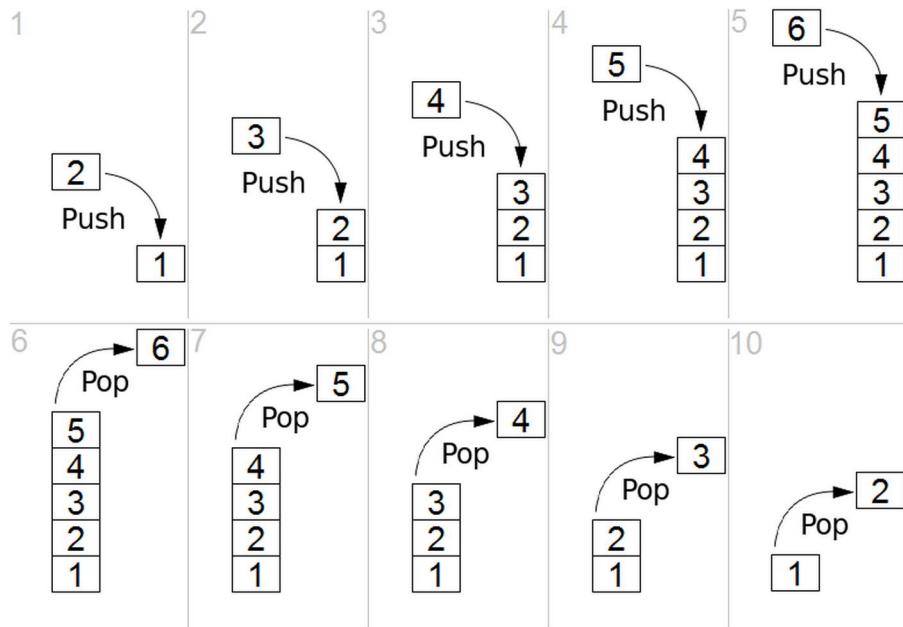
Sebagai contoh, misalkan saya ingin membuat program sistem antrian seperti di bank. Nasabah akan mengambil tiket antrian terlebih dahulu, kemudian menunggu giliran sampai dipanggil. Setiap nasabah yang mengambil tiket, akan diantrikan ke dalam *stack*.

Jika *stack* ini diibaratkan dengan array, setiap nasabah baru akan ditumpuk ke dalam array. Ketika customer service bank ingin memanggil antrian berikutnya, ia akan mengambil *stack* pada urutan paling bawah, karena inilah nasabah yang datang paling awal.

Konsep ini dikenal sebagai **FIFO** (*First In First Out*). Element yang lebih dulu antri, akan di proses terlebih dahulu. Ilustrasinya bisa digambarkan sebagai berikut:

Gambar: Konsep antrian FIFO (*First In First Out*), sumber: wikipedia

Selain FIFO, ada lagi istilah LIFO (*Last In, First Out*). Kali ini element yang terakhir datang, akan diproses paling awal:

Gambar: Konsep antrian LIFO (*Last In First Out*), sumber: wikipedia

Untuk membuat sistem seperti ini, PHP menyediakan 4 fungsi dasar: `array_push()`, `array_pop()`, `array_shift()`, dan `array_unshift()`. Fungsi `Array push()` sudah saya bahas, mari lanjut ke fungsi `array_pop()`.



Pengertian *stack*, **FIFO**, dan **LIFO** memang tidak untuk semua orang. Materi ini sebenarnya lebih cocok untuk mahasiswa IT. Namun saya merasa konsepnya cukup penting untuk dipahami. Terlebih array sangat sering digunakan di dalam PHP dan pemrograman web lain seperti JavaScript.

## 14.7 Function array\_pop()

Format penulisan:

```
mixed array_pop ( array &$array )
```

Fungsi **array\_pop()** digunakan untuk mengambil 1 element terakhir dari sebuah array. Selain itu, fungsi ini juga akan ‘memotong’ array karena 1 element sudah diambil. Langsung saja kita lihat contoh penggunaan fungsi ini:

```
1 <?php
2   $siswa = array("andi", "gina", "joko", "santi");
3
4   $satu_siswa = array_pop($siswa);
5   echo $satu_siswa; // santi
6   print_r($siswa); // Array ( [0] => andi [1] => gina [2] => joko )
7 ?>
```

Saya membuat array **\$siswa** dengan 4 element. Kemudian menggunakan fungsi **array\_pop(\$siswa)** dan menyimpan hasilnya ke variabel **\$satu\_siswa**. Perhatikan urutan array **\$siswa**, string “santi” berada diurutan terakhir. Oleh karena itu, fungsi **array\_pop(\$siswa)** akan ‘mengambil’ string ini.

Efek lain dari **array\_pop()** adalah, array **\$siswa** juga akan dikurangi dengan element yang sudah diambil tadi. Hal ini bisa dilihat dari perintah **print\_r(\$siswa)**, string “santi” sudah tidak ada lagi.

Jika saya terus melanjutkan beberapa kali pemanggilan fungsi **array\_pop()**, saya bisa mengeluarkan satu-satu element array, hingga array **\$siswa** menjadi kosong. Ketika fungsi **array\_pop()** mendapati element array sudah kosong, fungsi ini akan mengembalikan nilai **NULL**. Berikut contohnya:

```
1 <?php
2     $siswa = array("andi", "gina", "joko", "santi");
3
4     $satu_siswa = array_pop($siswa);
5     echo $satu_siswa;      // santi
6     print_r($siswa);      // Array ( [0] => andi [1] => gina [2] => joko )
7     echo "<br>";
8
9     $satu_siswa = array_pop($siswa);
10    echo $satu_siswa;     // joko
11    print_r($siswa);      // Array ( [0] => andi [1] => gina )
12    echo "<br>";
13
14    $satu_siswa = array_pop($siswa);
15    echo $satu_siswa;     // gina
16    print_r($siswa);      // Array ( [0] => andi )
17    echo "<br>";
18
19    $satu_siswa = array_pop($siswa);
20    echo $satu_siswa;     // andi
21    print_r($siswa);      // Array ( )
22    echo "<br>";
23
24    $satu_siswa = array_pop($siswa);
25    var_dump($satu_siswa); // NULL
26    print_r($siswa);      // Array ( )
27 ?>
```

Perhatikan, setiap kali fungsi `array_pop()` dipanggil, array `$siswa` juga berkurang sebanyak 1 element. Pemanggilan fungsi `array_pop()` saya lakukan 5 kali hingga seluruh element array `$siswa` kosong.

Jika anda sudah bisa memahami cara kerja fungsi `array_pop()`, saya akan membahas konsep perulangan array yang sering digunakan di dalam PHP. Ini banyak digunakan untuk menampilkan hasil tabel dari database seperti MySQL (pembahasan mengenai MySQL akan saya bahas dalam bab terpisah).

Dengan menggunakan fungsi `array_pop()`, saya bisa membuat sebuah perulangan untuk menampilkan seluruh data array. Berikut kode programnya:

```

1 <?php
2 $siswa = array("andi", "gina", "joko", "santi");
3 while ($satu_siswa = array_pop($siswa)) {
4     echo $satu_siswa;
5     echo "<br>";
6 }
7 ?>
8
9 /* Output-----
10 santi
11 joko
12 gina
13 andi
14 -----*/

```

Agar dapat memahami perulangan diatas, bandingkan dengan kode program sebelumnya. Kondisi `while ($satu_siswa = array_pop($siswa))` akan terus dijalankan selama array `$siswa` berisi ‘sesuatu’.

Ketika array `$siswa` sudah kosong, fungsi `array_pop()` akan mengembalikan nilai `NULL`. Kondisi ini akan membuat perulangan while menjadi `while (NULL)`. `NULL` secara otomatis di konversi menjadi boolean `FALSE`. Dengan kondisi `while (FALSE)`, perulangan akan berhenti.

Saya sangat sarankan anda untuk bisa memahami alur kerja kode program diatas, karena konsep ini hampir selalu digunakan untuk mengakses data dari tabel MySQL.

Tapi, bukankah ada perulangan **FOREACH**? Betul, dengan perulangan FOREACH, hasilnya juga akan sama. Tetapi terdapat kasus dimana kita tidak bisa memakai FOREACH. Salah satunya adalah untuk menampilkan data tabel dari database.



Hasil dari perulangan dengan fungsi `array_pop()` akan menampilkan element array dengan posisi terbalik (masih ingat konsep **FIFO (First In First Out?)**). Jika kita ingin element pertama ditampilkan terlebih dahulu (**LIFO/ Last In First Out**), bisa menggunakan fungsi `array_shift()` yang akan saya bahas sesaat lagi.

## 14.8 Function `array_unshift()`

Format penulisan:

```
int array_unshift ( array &$array , mixed $value1 [ , mixed $... ] )
```

Fungsi `array_unshift()` hampir sama dengan `array_push()`. Bedanya, kali ini element array ditambahkan di awal, bukan di akhir. Selain perbedaan ini, cara penggunaannya sama seperti `array_push()`. Berikut contohnya:

```

1 <?php
2   $siswa = array("andi", "gina", "joko", "santi");
3   $var = array_unshift($siswa, "rani");
4
5   echo $var; // 5
6   print_r($siswa);
7   // Array ( [0] => rani [1] => andi [2] => gina [3] => joko [4] => santi )
8 ?>

```

Kali ini string “rani” ditempatkan di posisi pertama array `$siswa`. Dengan penambahan ini, seluruh element array lain akan digeser posisinya kebelakang.

Saya juga bisa menginput beberapa element sekaligus dengan fungsi `array_unshift()`:

```

1 <?php
2   $nilai = array(98, 59, 42, 65, 87, 66);
3   $var = array_unshift($nilai, 82, 49, 99);
4
5   echo $var; // 9
6   print_r($nilai);
7   // Array ( [0] => 82 [1] => 49 [2] => 99 [3] => 98 [4] => 59
8   //           [5] => 42 [6] => 65 [7] => 87 [8] => 66 )
9 ?>

```

## 14.9 Function array\_shift()

Format penulisan:

```
mixed array_shift ( array &$array )
```

Fungsi `array_shift()` juga mirip dengan fungsi `array_pop()`. Bedanya, kali ini element akan diambil dari urutan pertama array, bukan dari urutan terakhir seperti fungsi `array_pop()`. Berikut contohnya:

```

1 <?php
2   $siswa = array("andi", "gina", "joko", "santi");
3
4   $satu_siswa = array_shift($siswa);
5   echo $satu_siswa; // andi
6   print_r($siswa); // Array ( [0] => gina [1] => joko [2] => santi )
7 ?>

```

Sekarang yang diambil terlebih dahulu adalah string “andi” yang berada di posisi pertama array `$siswa`.

Sama seperti fungsi `array_pop()`, saya juga bisa membuat perulangan untuk mengeluarkan seluruh element di dalam array dengan fungsi `array_shift()`:

```

1 <?php
2 $siswa = array("andi", "gina", "joko", "santi");
3   while ($satu_siswa = array_shift($siswa)) {
4     echo $satu_siswa;
5     echo "<br>";
6   }
7 ?>
8
9 /* Output-----
10 andi
11 gina
12 joko
13 santi
14 -----*/

```

Bedanya, kali ini element array akan ditampilkan mulai dari element pertama.

Pembahasan tentang fungsi `array_shift()` ini melengkapi 4 fungsi ‘stack’ di dalam PHP : `array_push()`, `array_pop()`, `array_shift()`, dan `array_unshift()`.

## 14.10 Function `current()`, `next()`, `prev()`, `end()` dan `reset()`

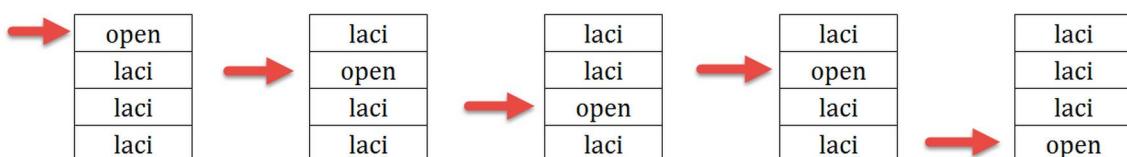
```

mixed current ( array &$array )
mixed next ( array &$array )
mixed prev ( array &$array )
mixed end ( array &$array )
mixed reset ( array &$array )

```

Kelima fungsi yang akan kita bahas kali ini digunakan untuk merubah **pointer** penunjuk array. Materi ini masih berkaitan dengan array sebagai *stack*.

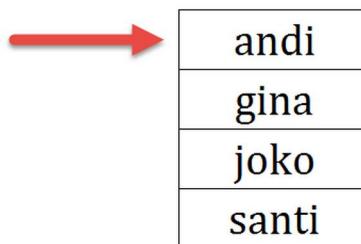
Agar mudah dipahami, mari bayangkan array sebagai sebuah lemari yang terdiri dari tumpukan laci-laci. Syaratnya, kita hanya bisa membuka 1 laci pada satu waktu. Selain itu, laci yang bisa dibuka harus memiliki label “open”. Jika ingin membuka laci lain, label “open” harus dilepaskan dari laci lama, kemudian pindahkan ke laci baru. Ilustrasinya bisa digambarkan sebagai berikut:



Gambar: Ilustrasi array sebagai tumpukan laci dengan 1 label ‘open’

Label “open” inilah yang bisa disamakan dengan sebuah **pointer** di dalam array. Pointer merupakan jarum penunjuk ke baris mana yang saat ini sedang di akses oleh PHP.

Sebagai contoh, jika saya membuat sebuah array `$siswa = array("andi", "gina", "joko", "santi")`, 'lemari'nya bisa digambarkan sebagai berikut:



Gambar: 'Lemari' dari array `$siswa`

Tanda panah merah menandakan posisi pointer. Ketika kode program dijalankan, pointer secara default menunjuk ke element pertama array: "andi".

Untuk mengambil string "andi", kita bisa menggunakan fungsi `current()`. Fungsi `current()` akan mengambil element array pada posisi pointer saat ini. Dalam contoh diatas, pointer berada di awal array, hasilnya adalah "andi". Berikut contoh kode program PHPnya:

```

1 <?php
2   $siswa = array("andi", "gina", "joko", "santi");
3
4   $satu_siswa = current($siswa);
5   echo $satu_siswa;    // andi
6 ?>

```

Untuk "menggerakkan" pointer, PHP menyediakan fungsi `next()` dan `prev()`. Seperti yang bisa anda tebak, fungsi `next()` digunakan untuk menggeser pointer ke element selanjutnya, sedangkan fungsi `prev()` bertujuan untuk menggeser pointer mundur 1 element.

Selain menggeser posisi pointer, fungsi `next()` dan `prev()` juga mengembalikan nilai element dimana pointer berada. Berikut contohnya:

```

1 <?php
2   $siswa = array("andi", "gina", "joko", "santi");
3
4   $satu_siswa = current($siswa);
5   echo $satu_siswa;    // andi
6   echo "<br>";
7
8   $satu_siswa = next($siswa);
9   echo $satu_siswa;    // gina
10  echo "<br>";
11
12  $satu_siswa = next($siswa);
13  echo $satu_siswa;    // joko

```

```

14 echo "<br>";
15
16 $satu_siswa = prev($siswa);
17 echo $satu_siswa; // gina
18 ?>

```

Perhatikan bagaimana saya menggeser posisi pointer menggunakan fungsi **next()** dan **prev()**. Alur pointer dalam kode program tersebut bisa diilustrasikan melalui gambar berikut:



Gambar: Perpindahan posisi pointer

Setiap pemanggilan fungsi **next()**, pointer akan maju 1 element, sedangkan ketika pemanggilan fungsi **prev()**, posisi pointer akan mundur sebanyak 1 element.

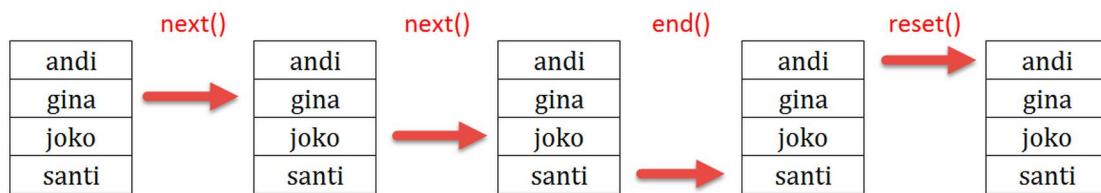
Untuk menggeser pointer ke element terakhir, tersedia fungsi **end()**. Sedangkan untuk mengembalikan posisi pointer ke element pertama array, bisa menggunakan fungsi **reset()**, berikut contohnya:

```

1 <?php
2   $siswa = array("andi", "gina", "joko", "santi");
3
4   $satu_siswa = next($siswa);
5   echo $satu_siswa; // gina
6   echo "<br>";
7
8   $satu_siswa = next($siswa);
9   echo $satu_siswa; // joko
10  echo "<br>";
11
12  $satu_siswa = end($siswa);
13  echo $satu_siswa; // santi
14  echo "<br>";
15
16  $satu_siswa = reset($siswa);
17  echo $satu_siswa; // andi
18 ?>

```

Berikut ilustrasi dari kode program diatas:



Gambar: Perpindahan posisi pointer

Dengan menggunakan kelima fungsi ini, kita bisa mengakses element array sesuai dengan kebutuhan. Jika anda masih kurang paham sistem kerjanya, silahkan latihan dengan membuat array lain dan gunakan fungsi-fungsi pointer ini .

Dalam implementasinya, kita relatif jarang mengakses satu-satu element array seperti ini. Walaupun begitu, pengetahuan tentang materi pointer array cukup penting untuk dipahami.

## 14.11 Function in\_array()

Format penulisan:

```
bool in_array ( mixed $needle , array $haystack [, bool $strict = FALSE ] )
```

Fungsi **in\_array()** cukup sederhana, tapi sangat berguna. Fungsi ini digunakan untuk mengecek apakah sebuah nilai ada di dalam array atau tidak. Hasil dari fungsi **in\_array()** adalah boolean **TRUE** atau **FALSE**.

Argumen pertama fungsi **in\_array()** diinput dengan string yang ingin dicari, sedangkan argumen kedua adalah array sumber. Berikut contoh penggunaannya:

```
1 <?php
2   $siswa = array("andi", "gina", "joko", "santi");
3   $cek = in_array("joko", $siswa);
4   var_dump($cek); // bool(true)
5 ?>
```

Saya mencari apakah string "joko" ada di dalam array **\$siswa**. Hasilnya? boolean **TRUE**. Ini berarti string "joko" ada di dalam array. Hasil seperti ini sangat cocok digunakan dalam kondisi IF, seperti contoh berikut:

```

1 <?php
2   $absensi = array("andi", "gina", "joko", "santi");
3   if (in_array("joko",$absensi)) {
4     echo "joko hadir";
5   }
6   else {
7     echo "joko tidak hadir";
8   }
9   // joko hadir
10 ?>

```

Argumen ketiga fungsi `in_array()` bersifat opsional. Fungsi argumen ini adalah untuk menentukan apakah proses pencarian akan melibatkan jenis tipe datanya atau tidak. Argumen ini bisa diisi dengan nilai boolean true atau false (nilai default). Berikut contohnya:

```

1 <?php
2   $nilai = array(98, 59, 42, 65, 87, 66, 82, 49, 99);
3
4   var_dump(in_array("66",$nilai));      // bool(true)
5   echo "<br>";
6   var_dump(in_array("66",$nilai,true)); // bool(false)
7   echo "<br>";
8   var_dump(in_array(66,$nilai,true));  // bool(true)
9 ?>

```

Sebagaimana yang kita ketahui, nilai 66 bertipe data *integer*, sedangkan "66" adalah *string*. Tanpa argumen ketiga, nilai 66 dan "66" akan dianggap sama (hasilnya TRUE). Namun jika ditambahkan argument ketiga dengan nilai TRUE, proses pencarian akan melibatkan tipe datanya, sehingga 66 dan "66" dianggap tidak sama (hasilnya FALSE).

## 14.12 Function array\_key\_exists()

```
bool array_key_exists ( mixed $key , array $array )
```

Fungsi `array_key_exists()` mirip dengan fungsi `in_array()`. Bedanya, di dalam fungsi `array_key_exists()` yang diperiksa adalah key array, bukan isi element array. Fungsi ini cocok digunakan untuk *associative array*, yakni array yang bagian key-nya berisi 'sesuatu'. Berikut contoh penggunaan fungsi `array_key_exists()`:

```
1 <?php
2     $nilai = array("anton" => 82, "rudy" => 81, "rini" => 95);
3
4     $cek = array_key_exists("rudy", $nilai);
5     var_dump($cek); // bool(true)
6
7     $cek = array_key_exists("joko", $nilai);
8     var_dump($cek); // bool(false)
9 ?>
```

Fungsi `array_key_exists()` tidak memiliki argumen ketiga seperti fungsi `in_array()`.

## 14.13 Function `array_search()`

Format penulisan:

```
mixed array_search ( mixed $needle , array $haystack
                    [, bool $strict = false ] )
```

Sesuai dengan namanya, fungsi `array_search()` digunakan untuk mencari suatu nilai di dalam array. Hasil kembalian fungsi ini adalah posisi index dari nilai yang ditemukan, atau boolean FALSE jika nilai yang dicari tidak ada. Berikut contoh penggunaannya:

```
1 <?php
2     $siswa = array("andi", "gina", "joko", "santi");
3
4     $key = array_search("joko", $siswa);
5     var_dump($key); // int(2)
6     echo "<br>";
7
8     $key = array_search("alex", $siswa);
9     var_dump($key); // bool(false)
10 ?>
```

Dengan menggunakan `array_search()`, kita bisa mendapatkan informasi yang lebih baik dari pada fungsi `in_array()`, yakni posisi dari array yang ditemukan. Ini sangat cocok untuk associative array:

```

1 <?php
2   $nilai = array("anton" => 82, "rudy" => 81, "rini" => 95);
3   $key = array_search(95,$nilai);
4
5   if ($key!==false) {
6     echo "Siswa yang dapat nilai 95 adalah $key";
7   }
8   else {
9     echo "Tidak ada siswa yang dapat nilai 95";
10  }
11 // Siswa yang dapat nilai 95 adalah rini
12 ?>

```

Saya ingin mengajak anda untuk ‘memikirkan’ sedikit kode diatas. Kenapa saya menggunakan kondisi `if ($key!==false)`? Kenapa tidak menggunakan kondisi `if ($key!=false)`? Dimana letak perbedaannya? padahal fungsi `array_search()` juga akan mengembalikan nilai `false` ketika tidak menemukan apa yang dicari.

Jawabannya, karena terdapat kemungkinan hasil dari fungsi `array_search()` adalah 0, dan ini akan menjadi `false`. Perhatikan contoh kasus berikut ini:

```

1 <?php
2   $siswa = array("andi", "gina", "joko", "santi");
3   $key = array_search("andi",$siswa);
4
5   if ($key!=false) {
6     echo "Siswa ada di index ke $key";
7   }
8   else {
9     echo "Siswa tidak ditemukan";
10  }
11 ?>

```

Dapatkah anda menebak apa hasilnya? Yup, hasilnya adalah “Siswa tidak ditemukan”. Inilah efek dari **type juggling** di PHP. Jika anda masih ingat, *type juggling* adalah konsep dimana PHP secara otomatis mengubah sebuah tipe data menjadi tipe data lain ketika dibutuhkan.

String “andi” berada di posisi pertama array `$siswa`, index dari element ini adalah 0. Dengan demikian, kondisi yang dijalankan adalah `if (0!=false)`. Karena ini adalah operasi perbandingan, integer 0 akan dikonversi PHP menjadi tipe data boolean. Di dalam PHP, 0 akan menjadi boolean `FALSE`. Sehingga kondisi `if (false!=false)` akan menghasilkan `false`, yang berarti string “andi” tidak ada di array `$siswa`.

Kesalahan logika seperti ini sangat sangat sulit di deteksi, karena PHP tidak mengeluarkan error. Bagaimana solusinya? Cukup dengan mengganti kondisi `if ($key!=false)` menjadi `if ($key!==false)`:

```

1 <?php
2 $siswa = array("andi", "gina", "joko", "santi");
3 $key = array_search("andi", $siswa);
4
5 if ($key !== false) {
6     echo "Siswa ada di index ke $key";
7 }
8 else {
9     echo "Siswa tidak ditemukan";
10 }
11 ?>

```

Operator perbandingan `!==` hanya akan menghasilkan `false` ketika kedua operandnya `false` dan harus tipe data yang sama. Kondisi `if (0 !== false)` akan menghasilkan `true`, karena integer 0 berbeda dengan boolean `false`. Dengan demikian, yang akan dijalankan adalah “Siswa ada di index ke 0”.

## 14.14 Function list()

Format penulisan:

```
array list ( mixed $var1 [, mixed $... ] )
```

Fungsi `list()` berguna untuk menginput value dari array ke variabel. Fungsi `list()` akan lebih mudah dijelaskan langsung dengan contoh:

```

1 <?php
2 $kegiatan = array("andi", "belajar", "PHP");
3
4 list($siapa, $mengapa, $apa) = $kegiatan;
5 echo "$siapa sedang $mengapa $apa";
6 // andi sedang belajar PHP
7 ?>

```

Perhatikan bagaimana cara fungsi `list()` memindahkan isi array `$kegiatan` ke variabel lain. Setiap element array akan berpasangan dengan variabel yang menjadi argumen fungsi ini.

Fungsi `list()` juga sering digunakan untuk menampilkan data tabel dari database. Sebagai contoh, perhatikan kode berikut:

```
1 <?php
2     $mahasiswa = array(
3         array("01", "Joko", "Akuntansi"),
4         array("02", "Nona", "Psikologi"),
5         array("03", "Lestya", "Matematika"),
6         array("04", "Tasya", "Kedokteran")
7     );
8
9     while(list($no, $nama, $jurusan) = array_shift($mahasiswa))
10    {
11        echo "$no $nama $jurusan";
12        echo "<br>";
13    }
14 ?>
15
16 /* Output-----
17 01 Joko Akuntansi
18 02 Nona Psikologi
19 03 Lestya Matematika
20 04 Tasya Kedokteran
21 -----*/
```

Saya mendefenisikan sebuah nested array **\$mahasiswa** yang berisi data-data mahasiswa. Setiap data ini berada di dalam array. Perhatikan bagaimana saya bisa menggunakan perulangan **while**, fungsi **list()** dan fungsi **array\_shift()** untuk mengeluarkan satu-satu element yang ada di dalam array **\$mahasiswa**. Teknik seperti ini cukup sering digunakan untuk menampilkan tabel dari database MySQL.

---

Dalam bab ini kita telah membahas berbagai jenis fungsi bawaan MySQL yang berkaitan dengan array. Selain itu, konsep array sebagai *stack* dan array *pointer* juga cukup penting dipahami.

# 15. Latihan: Menggabungkan HTML dan PHP

Dalam tiga bab terakhir, kita telah mempelajari fungsi-fungsi bawaan PHP. Kali ini saya akan mengajak anda ‘latihan’ berbagai konsep dasar PHP yang telah kita pelajari hingga saat ini. Latihan ini belum berupa sebuah website utuh, namun lebih kepada cara penggunaan PHP untuk menghasilkan kode HTML.

Materi ini juga sebagai pengantar untuk masuk ke bab-bab tentang praktik PHP seperti pemrosesan form atau untuk membuat koneksi dengan database MySQL.

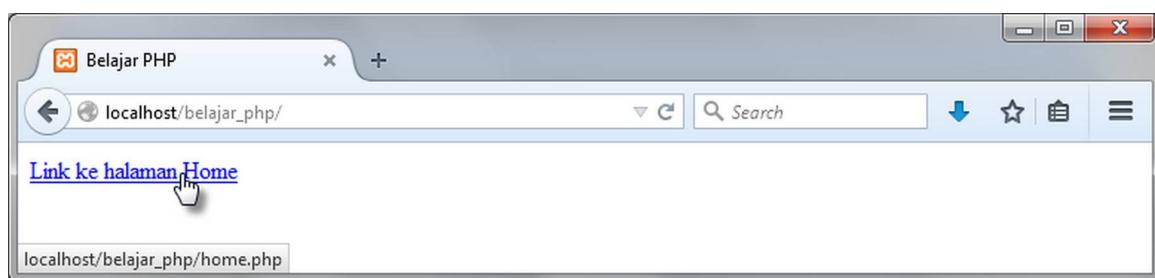
## 15.1 Membuat tag HTML dari PHP

Sebagai latihan pertama, saya akan membahas cara membuat tag `<a>` atau link dari PHP. Ini sebenarnya cukup sederhana, namun saya rasa penting untuk memastikan pemahaman anda tentang PHP dan HTML.

Perhatikan kode program berikut:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar PHP</title>
6 </head>
7 <body>
8   <a href="home.php">Link ke halaman Home</a>
9 </body>
10 </html>
```

Kode diatas adalah murni HTML dan tidak ada kode PHP sama sekali. Halaman ini akan menampilkan sebuah link ke halaman `home.php`.



Gambar: Sebuah link dari HTML

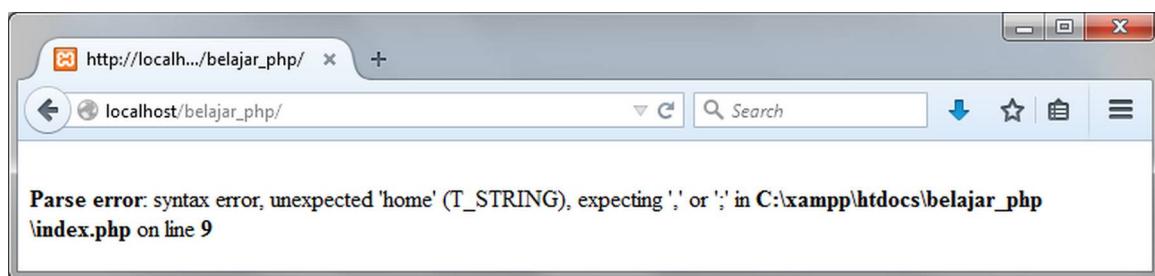
Selanjutnya, saya ingin link tersebut di generate dari PHP, bukan dari HTML. Berikut perubahannya:

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6  </head>
7  <body>
8  <?php
9      echo "<a href=\"home.php\">Link ke halaman Home</a>"; 
10 ?>
11 </body>
12 </html>

```

Kali ini saya ‘membawa’ tag `<a>` ke dalam perintah `echo`. Seperti yang terlihat, link sudah berada di dalam PHP. Namun ketika kode diatas dijalankan, hasilnya adalah:



Gambar: Parse error: syntax error

Dapatkah anda mencari dimana letak kesalahannya?

Error pada kode program diatas disebabkan oleh tanda kutip dua ( “ ) yang tertulis sebanyak 4 kali di dalam perintah `echo`. Hal seperti ini cukup sering terjadi, karena sebuah atribut HTML biasanya memang ditulis di dalam tanda kutip dua, yakni bagian : `<a href="home.php">`. Karena saya membuat `echo` juga dengan tanda kutip dua, tanda ini ‘bentrok’ dengan tanda kutip dua dari atribut `href`.

Terdapat beberapa solusi untuk masalah ini. Pertama, saya bisa menghapus tanda kutip di bagian atribut. Dengan demikian, perintah `echo` menjadi:

```

1  <?php
2      echo "<a href=home.php>Link ke halaman Home</a>"; 
3  ?>

```

Di dalam HTML5, penulisan atribut seperti ini valid dan tidak salah. Penambahan tanda kutip untuk bagian atribut berasal dari pendahulu HTML5, yakni dari **XHTML**. Walaupun begitu, menambahkan tanda kutip juga tidak salah. Bagi beberapa orang (termasuk saya), tanda kutip untuk atribut HTML sudah menjadi kebiasaan dan terkesan lebih rapi.

Karena saya tetap ingin menambahkan tanda kutip, saya bisa menggunakan alternatif penulisan kedua:

```

1 <?php
2 echo "<a href='home.php'>Link ke halaman Home</a>";
3 ?>

```

Tanda kutip satu di dalam tanda kutip dua tidak akan menghasilkan error. Di sisi HTML, tanda kutip satu ( ' ) maupun tanda kutip dua ( " ) dianggap sama dan juga tidak menjadi masalah.

Tapi bagaimana jika saya tetap ingin menggunakan tanda kutip dua? Saya terpaksa harus menggunakan *escape character* ( \ ), agar tanda kutip tersebut tidak diproses oleh PHP, inilah alternatif penulisan ketiga:

```

1 <?php
2 echo "<a href=\"home.php\">Link ke halaman Home</a>";
3 ?>

```

Penulisan seperti ini akan sangat sering anda jumpai. Walaupun terkesan ‘kurang rapi’, tapi inilah cara yang umum digunakan untuk membuat tag HTML di dalam perintah **echo**.

Tapi tunggu dulu, bukankah di PHP tersedia tanda kutip satu? Sehingga seharusnya saya bisa menggunakan alternatif penulisan keempat:

```

1 <?php
2 echo '<a href="home.php">Link ke halaman Home</a>';
3 ?>

```

Kali ini hasilnya lebih rapi, tanpa ada penambahan *escape character* ( \ ). Namun, ketika menggunakan perintah **echo** dengan tanda kutip satu seperti ini, saya kehilangan kemampuan untuk menampilkan variabel. Misalnya, saya tidak bisa membuat:

```

1 <?php
2 echo '<a href="home.php">Link ke halaman $nama_halaman</a>';
3 ?>

```

Variabel **\$nama\_halaman** tidak akan diproses dan ditampilkan sebagai bagian dari *string*. Jika menggunakan tanda kutip dua, variabel ini akan diproses oleh PHP.



Perbedaan antara penggunaan tanda kutip satu dan tanda kutip dua untuk **echo**, telah saya bahas pada materi tentang tipe data string PHP. Jika butuh ‘penyegaran’, anda bisa membacanya kembali.

Baik, saya memutuskan untuk tetap menggunakan tanda kutip dua pada perintah **echo**. Tapi, masih terdapat alternatif penulisan ke lima, yakni dengan memisahkan tag HTML dengan bagian yang harus dibuat dari PHP, berikut contoh yang saya maksud:

```
1 <a href="<?php echo "home.php"; ?>"><?php echo "Link ke halaman Home" ?></a>
```

Dapatkah anda menentukan mana kode HTML dan mana kode PHP?

Dalam contoh ini, saya hanya masuk ke ‘PHP Mode’ pada saat dibutuhka saja, yakni ketika menampilkan string “`home.php`” dan “**Link ke halaman Home**”. Tampilan seperti ini bisa dibilang sebagai ‘*kode PHP di dalam HTML*’. Kode PHP seolah-olah berada di dalam sebuah tag HTML.

Walaupun terlihat sedikit ‘kacau’, penggunaan kode HTML dan PHP seperti ini juga sangat sering digunakan. Jika anda iseng melihat kode theme **WordPress** atau template **Joomla** (yang juga dibuat dengan PHP), anda akan menemukan cara penulisan seperti ini.

Sebagai contoh, berikut potongan kode theme *twentyfifteen* bawaan WordPress 4:

```
1 <!DOCTYPE html>
2 <html <?php language_attributes(); ?> class="no-js">
3 <head>
4     <meta charset="<?php bloginfo( 'charset' ); ?>">
5     <meta name="viewport" content="width=device-width">
6     <link rel="profile" href="http://gmpg.org/xfn/11">
7     <link rel="pingback" href="<?php bloginfo( 'pingback_url' ); ?>">
```

Sekali lagi, dapatkah anda membedakan mana kode HTML dan mana kode PHP? di dalam contoh ini terdapat 3 fungsi PHP: `language_attributes()`, `bloginfo('charset')` dan `bloginfo('pingback_url')`. Saya tidak akan membahas apa fungsinya, tapi cukup perhatikan bagaimana cara penulisan ‘*PHP di dalam HTML*’.

Kembali ke pembahasan tentang cara menampilkan tag link dari PHP, setidaknya terdapat 5 alternatif penulisan, yakni:

```
1 <?php
2     echo "<a href=home.php>Link ke halaman Home</a>";
3     echo "<a href='home.php'>Link ke halaman Home</a>";
4     echo "<a href=\"home.php\">Link ke halaman Home</a>";
5     echo '<a href="home.php">Link ke halaman Home</a>';
6 ?>
7 <a href="<?php echo "home.php"; ?>"><?php echo "Link ke halaman Home" ?></a>
```

Yang manakah sebaiknya digunakan? Tidak ada jawaban pasti, karena semuanya valid dan tidak ada yang salah. Ini lebih kepada kesukaan kita masing-masing. Saya pribadi cenderung memilih alternatif penulisan ketiga dan kelima, yakni antara:

```
1 <?php
2     echo "<a href=\"home.php\">Link ke halaman Home</a>";
3 ?>
```

Dan

```
1 <a href="<?php echo "home.php"; ?>"><?php echo "Link ke halaman Home" ?></a>
```

Saya menggunakannya secara bergantian tergantung situasi. Kuncinya, gunakan mana yang membuat kode program lebih mudah ditulis dan mudah dikelola.

## 15.2 Membuat tag HTML dari Variabel PHP

Latihan selanjutnya, kita akan membuat ‘isi’ tag HTML dari sebuah variabel PHP. Ini sebenarnya hanya memodifikasi sedikit contoh kasus kita sebelumnya. Perhatikan kode program berikut ini:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar PHP</title>
6 </head>
7 <body>
8 <?php
9   $alamat_link="home.php";
10  $judul_link="Link ke halaman home";
11
12  echo "<a href=\"$alamat_link\">$judul_link</a>";
13 ?>
14 </body>
15 </html>
```

Kali ini saya membuat 2 buah variabel, yakni `$alamat_link` dan `$judul_link`. Keduanya kemudian ditampilkan dengan perintah `echo`. Tidak ada hal yang baru disini. Kedua variabel langsung diproses oleh perintah `echo` karena saya menggunakan tanda kutip dua ( “ ).

Sebagai alternatif penulisan, saya bisa membuat seperti ini:

```
1 <?php
2   $alamat_link="home.php";
3   $judul_link="Link ke halaman home";
4 ?>
5 <a href="<?php echo $alamat_link ?>"><?php echo $judul_link ?></a>
```

Ini adalah gaya penulisan ‘*PHP di dalam HTML*’, dimana saya menggunakan 2 kali perintah `echo` untuk menampilkan setiap variabel secara terpisah.

Ketika kode PHP bergabung dengan HTML seperti ini, biasanya bagian PHP akan ditempatkan di sisi paling atas halaman, yakni sebelum tag pembuka HTML: `<!DOCTYPE html>`. Berikut contohnya:

```

1  <?php
2      $alamat_link="home.php";
3      $judul_link="Link ke halaman home";
4  ?>
5  <!DOCTYPE html>
6  <html>
7  <head>
8      <meta charset="UTF-8">
9      <title>Belajar PHP</title>
10 </head>
11 <body>
12     <a href="<?php echo $alamat_link ?>"><?php echo $judul_link ?></a>
13 </body>
14 </html>

```

Dalam contoh ini, kode PHP saya pindahkan ke bagian atas halaman. Saat ini kode PHP tersebut hanya terdiri dari 2 baris (untuk membuat variabel `$alamat_link` dan `$judul_link`). Seiring dengan kompleksitas website, kode PHP di bagian ini bisa mencapai ratusan atau bahkan ribuan baris.

Dengan pemisahan seperti ini, kode PHP tampak ‘rapi’ dan terpisah dari HTML. Di bagian HTML, kita cukup menampilkan hasil dari pemrosesan kode PHP hanya ketika diperlukan.

## 15.3 Membuat tag HTML dari Array PHP

Baik, tidak ada masalah dengan menampilkan variabel ke dalam tag HTML. Tapi bagaimana dengan menampilkannya dari sebuah array? Kita akan masuk ke latihan berikutnya.

Sebagai contoh, saya memiliki array berikut:

```

1  <?php
2      $alamat_link=array("home.php",
3                          "Kategori.php",
4                          "artikel.php",
5                          "shop.php",
6                          "login.php");
7      $judul_link=array("Link ke halaman Home",
8                          "Link ke halaman Kategori",
9                          "Link ke halaman Artikel",
10                         "Link ke halaman Shop",
11                         "Link ke halaman Login");
12 ?>

```

Sebagai tantangan pertama, bisakah anda menampilkan isi salah satu element array diatas ke dalam sebuah link?

Berikut salah satu alternatif penulisan yang saya gunakan:

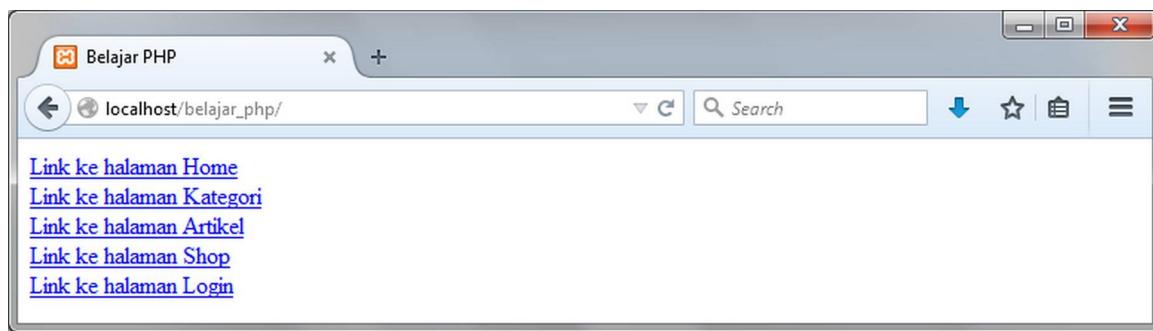
```
1 <?php
2 echo "<a href=\"$alamat_link[0]\">$judul_link[0]</a>";
3 ?>
```

Kita sudah membahas cara pengaksesan array pada bab tentang tipe data. Oleh karena itu saya tidak akan membahas kode diatas. Jika anda lupa, silahkan membaca kembali materi tersebut.

Tantangan selanjutnya, bisakah anda menampilkan seluruh isi array menjadi 5 link?

Berikut kode yang saya gunakan:

```
1 <?php
2 $alamat_link=array("home.php", "Kategori.php", "artikel.php", "shop.php",
3                     "login.php");
4 $judul_link=array("Link ke halaman Home", "Link ke halaman Kategori",
5                   "Link ke halaman Artikel", "Link ke halaman Shop",
6                   "Link ke halaman Login");
7 ?>
8 <!DOCTYPE html>
9 <html>
10 <head>
11   <meta charset="UTF-8">
12   <title>Belajar PHP</title>
13 </head>
14 <body>
15 <?php
16 echo "<a href=\"$alamat_link[0]\">$judul_link[0]</a>";
17 echo "<br>";
18 echo "<a href=\"$alamat_link[1]\">$judul_link[1]</a>";
19 echo "<br>";
20 echo "<a href=\"$alamat_link[2]\">$judul_link[2]</a>";
21 echo "<br>";
22 echo "<a href=\"$alamat_link[3]\">$judul_link[3]</a>";
23 echo "<br>";
24 echo "<a href=\"$alamat_link[4]\">$judul_link[4]</a>";
25 echo "<br>";
26 ?>
27 </body>
28 </html>
```



Gambar: 5 link yang berasal dari element array PHP

Juga tidak ada yang baru disini. Saya hanya mengulang penulisan array sebanyak 5 kali, tentunya dengan index array yang berbeda-beda.

Selanjutnya, dapatkan anda mengubah kode program diatas menjadi ‘*PHP di dalam HTML*’? Maksudnya, kita masuk ke ‘PHP Mode’ hanya ketika diperlukan, seperti contoh kita sebelumnya:

```
1 <a href="php echo $alamat_link ?&gt;"<?php echo $judul_link ?></a>
```

Namun kali ini, variabel `$alamat_link` dan `$judul_link` harus diganti dengan array. Saya sangat sarankan anda mencobanya terlebih dahulu.

Berikut kode yang saya gunakan:

```
1 <a href="php echo $alamat_link[0] ?&gt;"<?php echo $judul_link[0] ?></a>
2 <br>
3 <a href="php echo $alamat_link[1] ?&gt;"<?php echo $judul_link[1] ?></a>
4 <br>
5 <a href="php echo $alamat_link[2] ?&gt;"<?php echo $judul_link[2] ?></a>
6 <br>
7 <a href="php echo $alamat_link[3] ?&gt;"<?php echo $judul_link[3] ?></a>
8 <br>
9 <a href="php echo $alamat_link[4] ?&gt;"<?php echo $judul_link[4] ?></a>
```

Untuk membuat 5 link diatas, saya hanya perlu memisahkan mana bagian HTML dan mana bagian untuk PHP.

Jika anda perhatikan, cara pengaksesan array seperti ini terkesan diulang-ulang. Jadi, bukankah sangat tepat untuk sebuah perulangan?

Inilah tantangan anda berikutnya, bisakah anda membuat sebuah perulangan untuk menampilkan kelima link? Anda bisa menggunakan perulangan apa saja, apakah itu `for`, `while`, maupun `foreach`.

Baik, sudah? Berikut kode yang saya gunakan:

```

1 <?php
2   for ($i=0; $i<count($alamat_link); $i++)
3   {
4     echo "<a href=\"$alamat_link[$i]\">$judul_link[$i]</a>";
5     echo "<br>";
6   }
7 ?>

```

Jika anda tidak berhasil membuat perulangan diatas, silahkan pelajari sejenak. Seluruh konsepnya sudah kita bahas pada bab tentang perulangan dan juga ketika membahas fungsi **count()**.

Masih terkait dengan perulangan, bisakah anda menjelaskan maksud kode berikut:

```

1 <?php
2   $alamat_link=array("home.php", "Kategori.php", "artikel.php", "shop.php",
3                      "login.php");
4   $judul_link=array("Link ke halaman Home", "Link ke halaman Kategori",
5                      "Link ke halaman Artikel", "Link ke halaman Shop",
6                      "Link ke halaman Login");
7 ?>
8 <!DOCTYPE html>
9 <html>
10 <head>
11   <meta charset="UTF-8">
12   <title>Belajar PHP</title>
13 </head>
14 <body>
15 <?php
16   for ($i=0; $i<count($alamat_link); $i++)
17   {
18 ?>
19   <a href="<?php echo $alamat_link[$i] ?>"><?php echo $judul_link[$i] ?></a>
20   <br>
21 <?php
22   }
23 ?>
24 </body>
25 </html>

```

Kali ini saya menggunakan perulangan dengan model ‘*PHP di dalam HTML*’. Perhatikan bagaimana penulisannya. Saya membuat awal perulangan dengan:

```

1 <?php
2   for ($i=0; $i<count($alamat_link); $i++)
3   {
4 ?>

```

Walaupun kita sudah keluar dari ‘PHP mode’, tetapi perulangan ini masih tetap jalan. Oleh karena itulah saya harus menutupnya kembali dengan tanda:

```

1 <?php
2   }
3 ?>

```

Penulisan seperti ini banyak digunakan dalam pembuatan template seperti theme WordPress. Sebagai alternatif, saya juga bisa menulisnya seperti berikut:

```

1 <?php
2   for ($i=0; $i<count($alamat_link); $i++):
3 ?>
4   <a href="php echo $alamat_link[$i] ?&gt;"&gt;?<php echo $judul_link[$i] ?&gt;&lt;/a&gt;
5   &lt;br&gt;
6 &lt;?php
7   endfor;
8 ?&gt;
</pre

```

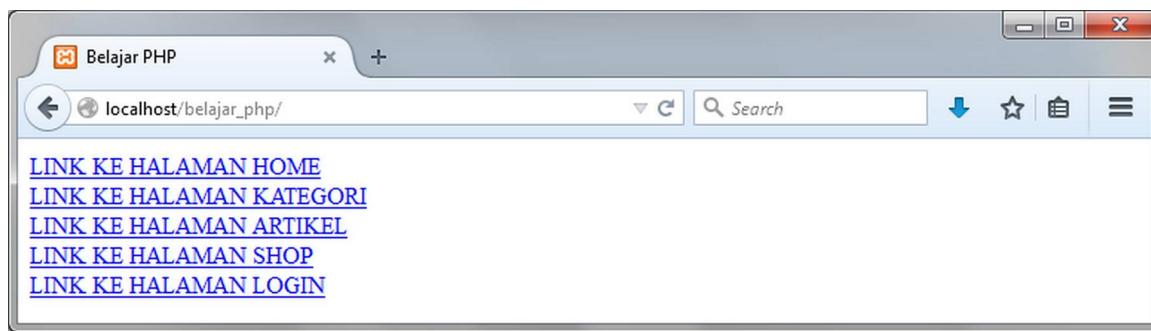
Disini saya mengganti blok perulangan **for** bukan dengan kurung kurawal, melainkan dengan blok ‘:’ dan ‘**endfor**’. Alternatif penulisan seperti ini juga telah saya bahas pada materi tentang struktur perulangan PHP.

Tantangan berikutnya, saya ingin agar judul link menjadi huruf besar tanpa harus mengubah array asli. Bagaimana caranya? Fungsi **strtoupper()** siap membantu:

```

1 <?php
2   for ($i=0; $i<count($alamat_link); $i++)
3   {
4 ?>
5   <a href="php echo $alamat_link[$i] ?&gt;"&gt;
6   &lt;?php echo strtoupper($judul_link[$i]) ?&gt;&lt;/a&gt;
7   &lt;br&gt;
8 &lt;?php
9   }
10 ?&gt;
</pre

```



Gambar: Teks untuk link ditampilkan dengan huruf besar

Disinilah fungsi-fungsi string PHP berperan. Kita bisa menformat tampilan teks sesuai dengan kebutuhan, apakah menjadi huruf besar, huruf kecil, menambahkan beberapa karakter, dll.

Jika anda perhatikan, kumpulan link diatas lebih pas sebagai menu navigasi untuk sebuah website. Jadi apa yang harus ditambahkan?

Saatnya CSS yang mengambil alih. Di dalam buku **CSS Uncover**, saya telah membahas cara membuat menu horizontal dan vertikal. Bagi anda yang lupa-lupa ingat (atau belum membeli buku CSS yang sangat bagus itu, hehe..) Untuk membuat menu, kita menempatkan link di dalam *unordered list*, yakni tag `<a>` di dalam tag `<ul>` dan `<li>`, lalu tambahkan sedikit ‘bumbu’ CSS. Berikut contohnya:

```

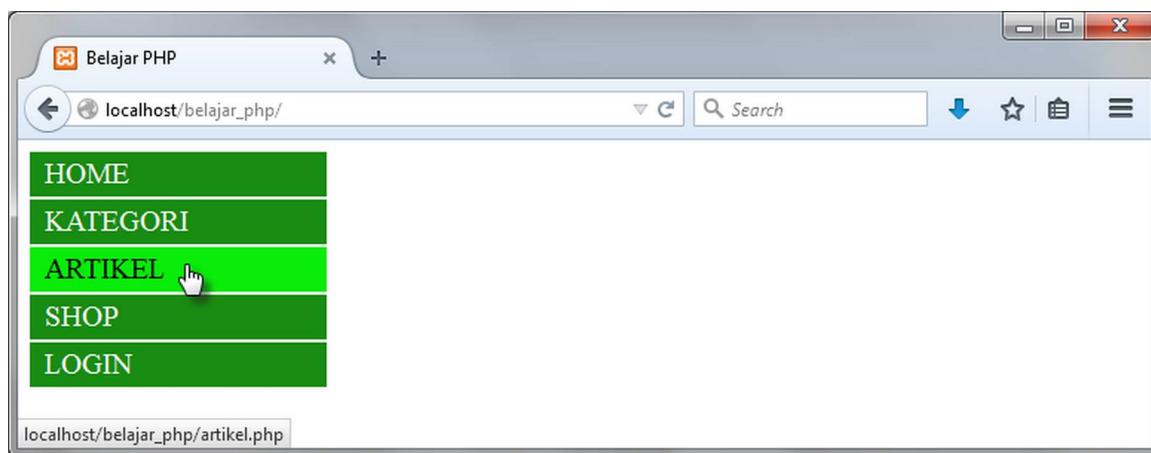
1 <?php
2   $alamat_link=array("home.php", "Kategori.php", "artikel.php", "shop.php",
3                      "login.php");
4   $judul_link=array("Home", "Kategori", "Artikel", "Shop", "Login");
5 ?>
6 <!DOCTYPE html>
7 <html>
8 <head>
9   <meta charset="UTF-8">
10  <title>Belajar PHP</title>
11 </head>
12 <style>
13   ul {
14     list-style: none;
15     padding: 0;
16     margin: 0;
17     width: 200px;
18   }
19   li a {
20     display: block;
21     background-color: #198C13;
22     color: white;
23     text-decoration: none;
24     font-size: 20px;
25     height: 30px;

```

```

26     line-height: 30px;
27     padding-left: 10px;
28     border-bottom: 2px solid #FFF;
29 }
30 li a:hover {
31     background-color: #0AED0A;
32     color: black;
33 }
34 </style>
35 <body>
36 <ul>
37 <?php
38     for ($i=0; $i<count($alamat_link); $i++)
39     {
40     ?>
41     <li>
42         <a href="<?php echo $alamat_link[$i] ?>">
43             <?php echo strtoupper($judul_link[$i]) ?></a>
44     </li>
45 <?php
46     }
47 ?>
48 </ul>
49 </body>
50 </html>

```



Gambar: Link Navigasi yang dibuat dari HTML, CSS dan PHP

Khusus untuk array, saya mengubah sedikit isi array `$judul_link` agar lebih singkat. Selebihnya, inilah kolaborasi dari 3 bahasa pemrograman web: **HTML**, **CSS**, dan **PHP**. Silahkan anda pelajari sesaat untuk dapat memahami peran dari masing-masing bahasa tersebut.

## 15.4 Menampilkan Associative Array PHP

Setelah ‘bermain-main’ dengan link untuk membuat menu navigasi, kita akan masuk ke latihan selanjutnya. Tantangan kali ini adalah bagaimana cara menampilkan data dari *associative array* PHP.

Baik, arraynya adalah sebagai berikut:

```

1 <?php
2 $siswa=array(
3     "siswa1" => array ("Joko","Medan","12 Agustus 1998"),
4     "siswa2" => array ("Rini","Jakarta","22 Juli 1999"),
5     "siswa3" => array ("Alex","Bandung","9 Januari 2000"),
6     "siswa4" => array ("Joy","Samarinda","4 Maret 1998"),
7     "siswa5" => array ("Santi","Palembang","12 Desember 1999")
8 );
9 ?>
10 <!DOCTYPE html>
11 <html>
12 <head>
13   <meta charset="UTF-8">
14   <title>Belajar PHP</title>
15 </head>
16 <body>
17 <?php
18 // kode PHP disini
19 ?>
20 </body>
21 </html>

```

Tantangan pertama, bisakah anda menampilkan salah satu nama siswa? Tidak perlu menambahkan HTML, tapi cukup dengan perintah **echo** PHP. Yang penting nama siswa bisa tampil di web browser. Jika anda sedikit lupa, silahkan pelajari sebentar pembahasan tentang tipe data array di bab tentang tipe data.

Berikut salah satu perintah yang saya gunakan:

```

1 <?php
2 echo $siswa["siswa1"][0]; // Joko
3 echo $siswa["siswa1"][1]; // Medan
4 echo $siswa["siswa1"][2]; // 12 Agustus 1998
5 ?>

```

Perintah **echo \$siswa["siswa1"][0]**, berarti saya ingin menampilkan data index ke-0 dari array **\$siswa1**, yang berada di dalam array **\$siswa**.

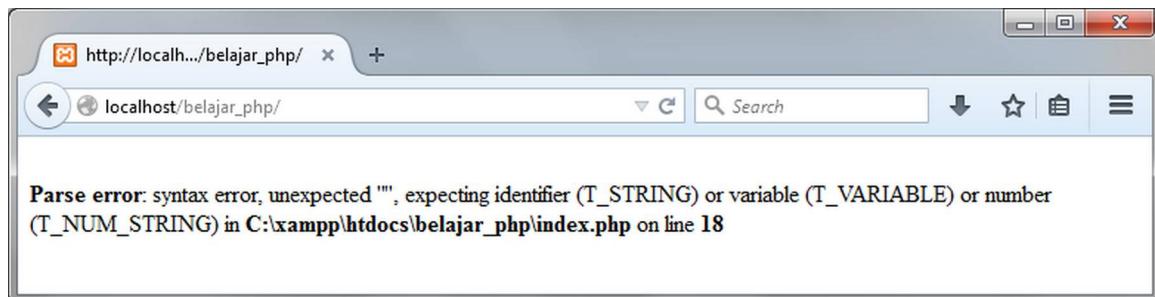
Selanjutnya, dapatkah anda menyatukan 3 baris ini menjadi 1 perintah **echo**? Mungkin anda akan mencobanya seperti berikut ini:

```

1 <?php
2 echo "$siswa["siswa1"] [0] $siswa["siswa1"] [1] $siswa["siswa1"] [2]";
3 ?>

```

Ketika dijalankan, hasilnya:



Gambar: Parse error: syntax error

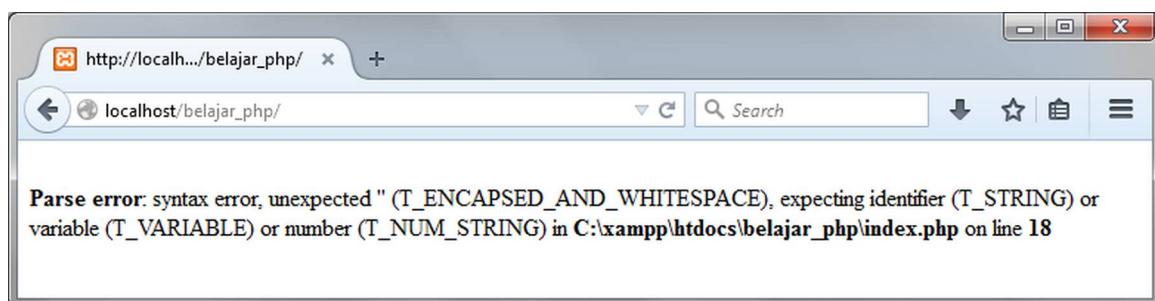
Bisakah anda menebak apa yang salah? Yup, terdapat banyak tanda kutip dua yang saling bertumpuk. Ini berasal dari cara kita mengakses *associative array*. Baiklah, bagaimana jika mengubahnya menjadi tanda kutip satu?

```

1 <?php
2 echo "$siswa['siswa1'] [0] $siswa['siswa1'] [1] $siswa['siswa1'] [2]";
3 ?>

```

Saya mengganti cara pengaksesan array dari `$siswa["siswa1"] [0]` menjadi `$siswa['siswa1'] [0]`, agar tidak ‘bentrok’ dengan tanda kutip dua. Hasilnya?



Gambar: Parse error: syntax error (masih error..??)

Apa yang terjadi? Inilah salah satu ‘jebakan betmen’ jika kita mengakses *associative array* di dalam sebuah string (menggunakan tanda kutip dua dan perintah `echo`). PHP tidak membolehkan hal ini. Jadi bagaimana solusinya? Terdapat 2 cara.

Cara pertama, adalah dengan mengeluarkan *associative array* dari string, seperti contoh berikut:

```

1 <?php
2 echo $siswa["siswa1"] [0] . $siswa["siswa1"] [1] . $siswa["siswa1"] [2];
3 ?>

```

Kali ini saya tidak menempatkan *associative array* di dalam tanda kutip dua. Saya menggunakan operator titik untuk menyambung ketiga array. Bagaimana jika ingin ada karakter pembatas diantara ketiga element array ini?

```

1 <?php
2   echo $siswa["siswa1"][0] . "-" . $siswa["siswa1"][1] . "-" . $siswa["siswa1"][2];
3 ?>

```

Materi ini penyambungan string (*string concatenate*) seperti ini telah kita pada bab tentang operator string.

Selain ‘mengeluarkan array’, terdapat cara kedua untuk menampilkan *associative array*, yakni menggunakan tanda kurung kurawal { dan } untuk membedakan antara array dengan string. Berikut contohnya:

```

1 <?php
2   echo "{$siswa["siswa1"]}[0]-{$siswa["siswa1"]}[1]-{$siswa["siswa1"]}[2]";
3 ?>

```

Penggunaan tanda { dan } untuk menampilkan variabel pernah saya singgung ketika mendapati kasus seperti ini:

```

1 <?php
2   $kata="sepak";
3   echo "{$kata}bola"; // sepakbola
4 ?>

```

Solusi ini juga bisa digunakan untuk menampilkan *associative array* PHP yang ‘menempel’ dengan string lain.

Baik, sekarang kita akan menambahkan unsur ‘HTML’, jadi, bagaimana cara menampilkan kelima data siswa ini? Tentunya cukup dengan mengulang 5 kali perintah **echo**:

```

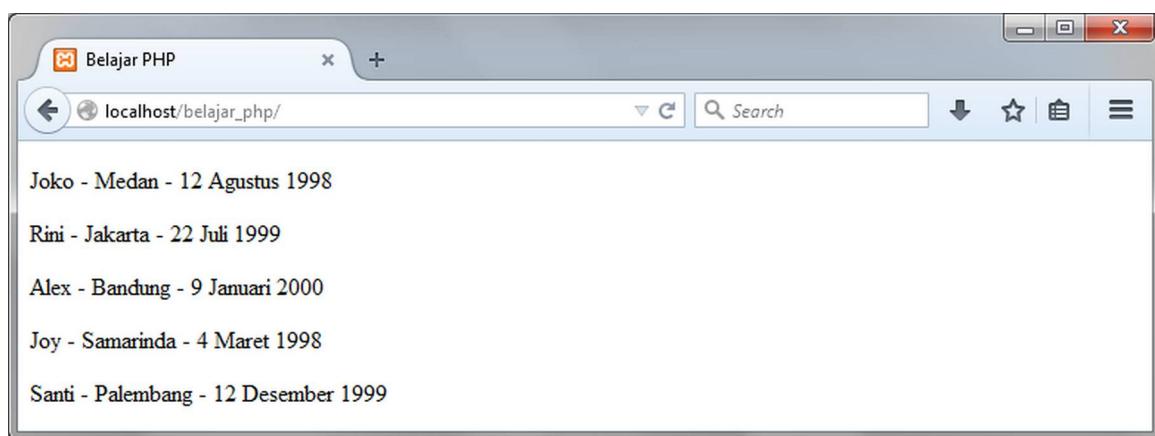
1 <?php
2 $siswa=array("siswa1" => array ("Joko", "Medan", "12 Agustus 1998"),
3             "siswa2" => array ("Rini", "Jakarta", "22 Juli 1999"),
4             "siswa3" => array ("Alex", "Bandung", "9 Januari 2000"),
5             "siswa4" => array ("Joy", "Samarinda", "4 Maret 1998"),
6             "siswa5" => array ("Santi", "Palembang", "12 Desember 1999")
7 );
8 ?>
9 <!DOCTYPE html>
10 <html>
11 <head>
12   <meta charset="UTF-8">
13   <title>Belajar PHP</title>

```

```

14  </head>
15  <body>
16  <?php
17  echo "<p>{$siswa["siswa1"]}[0] - {$siswa["siswa1"]}[1] - "
18  "{$siswa["siswa1"]}[2] </p>";
19  echo "<p>{$siswa["siswa2"]}[0] - {$siswa["siswa2"]}[1] - "
20  "{$siswa["siswa2"]}[2] </p>";
21  echo "<p>{$siswa["siswa3"]}[0] - {$siswa["siswa3"]}[1] - "
22  "{$siswa["siswa3"]}[2] </p>";
23  echo "<p>{$siswa["siswa4"]}[0] - {$siswa["siswa4"]}[1] - "
24  "{$siswa["siswa4"]}[2] </p>";
25  echo "<p>{$siswa["siswa5"]}[0] - {$siswa["siswa5"]}[1] - "
26  "{$siswa["siswa5"]}[2] </p>";
27 ?>
28 </body>
29 </html>

```



Gambar: Kelima isi array \$siswa sukses ditampilkan

Sekarang, seluruh isi array sudah tampil di web browser. Saya sengaja memberi contoh kasus sebuah *associative array*, karena inilah bentuk umum data yang berasal dari database.

Ketika ada kode program yang diulang-ulang seperti diatas, tentunya lebih pas untuk sebuah perulangan. Bisakah anda mengubah kode tersebut dan menampilkannya dari sebuah perulangan?

Berikut kode yang saya gunakan:

```

1  <?php
2  foreach ($siswa as $value) {
3      echo "<p>{$value[0]} - {$value[1]} - {$value[2]}</p>";
4  }
5 ?>

```

Kali ini saya menggunakan perulangan **foreach**. Saya yakin saat ini anda sudah bisa memahami apa yang dilakukan oleh kode PHP diatas.

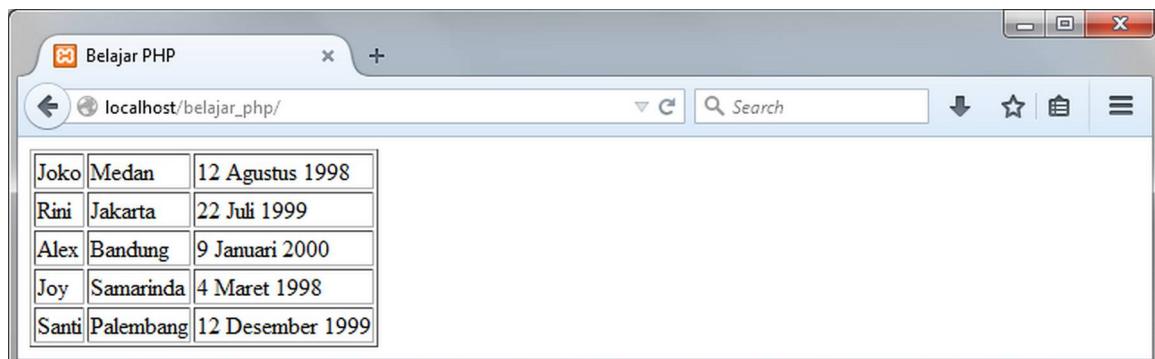
Jika anda perhatikan, data siswa ini akan lebih pas jika ditampilkan ke dalam bentuk tabel. Dan, inilah tantangan berikutnya. Bisakah anda menampilkan data siswa ke dalam bentuk tabel HTML? Agar lebih menantang, tabel ini juga digenerate menggunakan perulangan, jadi kita tidak perlu menulis setiap baris.

Berikut kode program yang saya gunakan untuk membuat tabel siswa:

```

1 <table border="1">
2 <?php
3   foreach ($siswa as $value) {
4     echo "<tr><td>{$value[0]}</td><td>{$value[1]}</td>
5       <td>{$value[2]}</td></tr>";
6   }
7 ?>
8 </table>

```



Gambar: Array siswa ditampilkan dalam bentuk tabel

Saya memecah penulisan **echo** menjadi 2 baris karena keterbatasan panjang halaman. Tapi sebenarnya itu adalah 1 baris string.

Jika anda sudah paham cara pembuatan tabel di HTML, saya yakin anda juga paham maksud kode ini. Jika tidak, silahkan baca kembali buku HTML Uncover, bab tentang tabel :)

Sebagai alternatif penulisan, saya bisa memecahnya menjadi beberapa perintah echo:

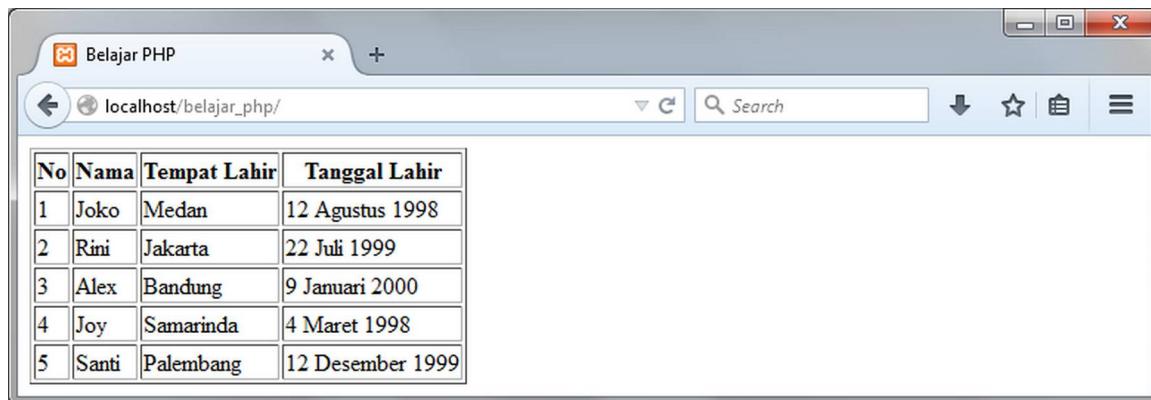
```

1 <table border="1">
2 <?php
3   foreach ($siswa as $value) {
4     echo "<tr>";
5     echo "<td>{$value[0]}</td>";
6     echo "<td>{$value[1]}</td>";
7     echo "<td>{$value[2]}</td>";
8     echo "</tr>";
9   }
10 ?>
11 </table>

```

Untuk data tabel yang cukup kompleks, saya sering menggunakan penulisan seperti ini. Memang sedikit lebih panjang, tapi kode PHPnya lebih mudah dikelola.

Latihan selanjutnya, dapatkah anda memodifikasi kode diatas agar tampil seperti berikut?



No	Nama	Tempat Lahir	Tanggal Lahir
1	Joko	Medan	12 Agustus 1998
2	Rini	Jakarta	22 Juli 1999
3	Alex	Bandung	9 Januari 2000
4	Joy	Samarinda	4 Maret 1998
5	Santi	Palembang	12 Desember 1999

Gambar: Tabel siswa dengan judul kolom dan penomoran

Kali ini saya ingin tabel tersebut memiliki judul kolom: **No**, **Nama**, **Tempat Lahir** dan **Tanggal Lahir**. Selain itu, penomoran otomatis berisi angka 1, 2, 3 dst, sesuai dengan banyak element di dalam array (bukan dibuat manual).

Berikut kode yang saya gunakan untuk tampilan seperti ini:

```

1  <?php
2  $siswa=array("siswa1" => array ("Joko","Medan","12 Agustus 1998"),
3  "siswa2" => array ("Rini","Jakarta","22 Juli 1999"),
4  "siswa3" => array ("Alex","Bandung","9 Januari 2000"),
5  "siswa4" => array ("Joy","Samarinda","4 Maret 1998"),
6  "siswa5" => array ("Santi","Palembang","12 Desember 1999")
7  );
8  ?>
9  <!DOCTYPE html>
10 <html>
11 <head>
12   <meta charset="UTF-8">
13   <title>Belajar PHP</title>
14 </head>
15 <body>
16 <table border="1">
17 <tr>
18   <th>No</th>
19   <th>Nama</th>
20   <th>Tempat Lahir</th>
21   <th>Tanggal Lahir</th>
22 </tr>
23 <?php
24   $i=1;

```

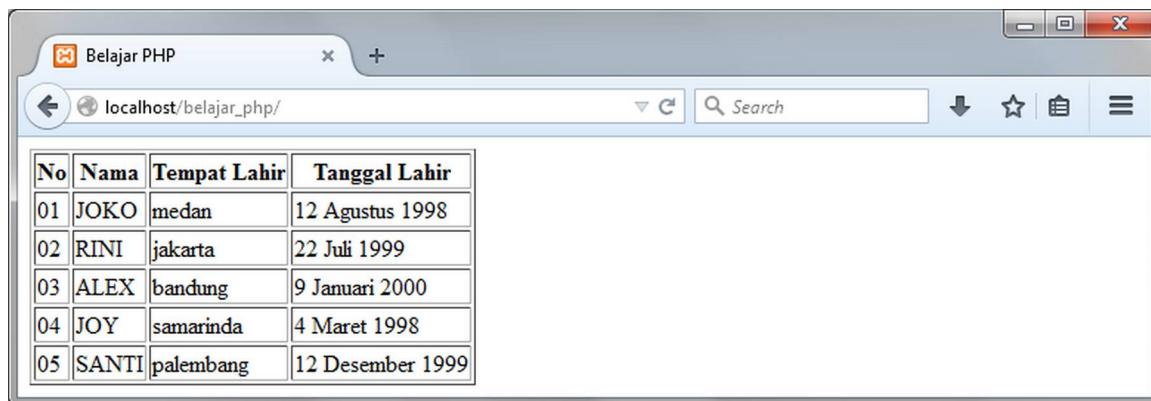
```

25  foreach ($siswa as $value) {
26      echo "<tr>";
27      echo "<td>$i</td>";
28      echo "<td>{$value[0]}</td>";
29      echo "<td>{$value[1]}</td>";
30      echo "<td>{$value[2]}</td>";
31      echo "</tr>";
32      $i++;
33  }
34 ?>
35 </table>
36 </body>
37 </html>

```

Perhatikan bagaimana saya membuat nomor urut menggunakan variabel `$i`. Variabel ini akan terus naik (*increment*) sebanyak 1 angka pada setiap perulangan. Dengan demikian berapapun isi dari array siswa, nomor urut ini akan terus ditampilkan.

Tantangan selanjutnya, bagaimana dengan tampilan seperti ini?



The screenshot shows a web browser window titled 'Belajar PHP'. The address bar displays 'localhost/belajar\_php/'. The content of the browser is a table with the following data:

No	Nama	Tempat Lahir	Tanggal Lahir
01	JOKO	medan	12 Agustus 1998
02	RINI	jakarta	22 Juli 1999
03	ALEX	bandung	9 Januari 2000
04	JOY	samarinda	4 Maret 1998
05	SANTI	palembang	12 Desember 1999

Gambar: Array `$siswa` disajikan dalam bentuk tabel

Ada 3 hal yang saya inginkan:

1. Penomoran sekarang dimulai dari 01, 02, 03,.. 10, 11, dst.
2. Nama ditampilkan dengan huruf besar.
3. Tempat Lahir ditampilkan dengan huruf kecil.

Untuk mendapatkan hasil seperti ini anda tidak boleh merubah isi array asli. Petunjuknya, gunakan fungsi-fungsi bawaan PHP, yakni fungsi `str_pad()` untuk penomoran, `strtoupper()` untuk kolom *Nama*, dan `strtolower()` untuk kolom *Tempat Lahir*.

Berikut kode yang saya gunakan:

```

1  <?php
2  $siswa=array("siswa1" => array ("Joko","Medan","12 Agustus 1998"),
3  "siswa2" => array ("Rini","Jakarta","22 Juli 1999"),
4  "siswa3" => array ("Alex","Bandung","9 Januari 2000"),
5  "siswa4" => array ("Joy","Samarinda","4 Maret 1998"),
6  "siswa5" => array ("Santi","Palembang","12 Desember 1999")
7  );
8  ?>
9  <!DOCTYPE html>
10 <html>
11 <head>
12   <meta charset="UTF-8">
13   <title>Belajar PHP</title>
14 </head>
15 <body>
16 <table border="1">
17 <tr>
18   <th>No</th>
19   <th>Nama</th>
20   <th>Tempat Lahir</th>
21   <th>Tanggal Lahir</th>
22 </tr>
23 <?php
24   $i=1;
25   foreach ($siswa as $value) {
26     echo "<tr>";
27     echo "<td>".str_pad($i,2,0,STR_PAD_LEFT)."</td>";
28     echo "<td>".strtoupper($value[0])."</td>";
29     echo "<td>".strtolower($value[1])."</td>";
30     echo "<td>{$value[2]}</td>";
31     echo "</tr>";
32     $i++;
33   }
34 ?>
35 </table>
36 </body>
37 </html>

```

Selamat, jika anda berhasil merancang tampilannya tanpa melihat kode yang saya buat. Kodennya tidak harus sama persis, yang penting tampilan dan fiturnya sama.

Dapat terlihat bagaimana saya menggunakan fungsi **str\_pad()**, **strtoupper()** dan **strtolower()** untuk mengubah tampilan array. Trik seperti ini WAJIB hukumnya anda pahami, karena sangat sering digunakan.

Terakhir, saya bisa menambahkan sedikit bumbu CSS untuk membuat tabel ini menjadi sedikit lebih cantik:

```
1  <?php
2  $siswa=array("siswa1" => array ("Joko","Medan","12 Agustus 1998"),
3  "siswa2" => array ("Rini","Jakarta","22 Juli 1999"),
4  "siswa3" => array ("Alex","Bandung","9 Januari 2000"),
5  "siswa4" => array ("Joy","Samarinda","4 Maret 1998"),
6  "siswa5" => array ("Santi","Palembang","12 Desember 1999")
7  );
8  ?>
9  <!DOCTYPE html>
10 <html>
11 <head>
12   <meta charset="UTF-8">
13   <title>Belajar PHP</title>
14   <style>
15   table {
16     border-collapse:collapse;
17     border-spacing:0;
18     font-size:18px;
19   }
20   table th {
21     padding:10px;
22     color:#fff;
23     background-color:#2A72BA;
24     border-top:1px black solid;
25     border-bottom:1px black solid;
26   }
27   table td {
28     padding:10px;
29     border-top:1px black solid;
30     border-bottom:1px black solid;
31   }
32   tr:nth-child(even) {
33     background-color: #DFEBF8;
34   }
35   </style>
36 </head>
37 <body>
38 <table border="1">
39 <tr>
40   <th>No</th>
41   <th>Nama</th>
42   <th>Tempat Lahir</th>
43   <th>Tanggal Lahir</th>
44 </tr>
45 <?php
46   $i=1;
```

```

47  foreach ($siswa as $value) {
48      echo "<tr>";
49      echo "<td>".str_pad($i,2,0,STR_PAD_LEFT)."</td>";
50      echo "<td>".strtoupper($value[0])."</td>";
51      echo "<td>".strtolower($value[1])."</td>";
52      echo "<td>{$value[2]}</td>";
53      echo "</tr>";
54      $i++;
55  }
56 ?>
57 </table>
58 </body>
59 </html>

```

Gambar: Tampilan tabel siswa dengan penambahan CSS

Kode diatas menutup bab latihan kita kali ini. Saya sangat sarankan anda mencoba-coba hal lain untuk mengkombinasikan HTML, PHP, dan CSS seperti apa yang sudah kita lakukan disini. Misalnya pada latihan diatas saya menggunakan perulangan **foreach**. Bisakah anda membuatnya dengan perulangan **for** atau **while**?

Atau bisakah anda menampilkan baris untuk siswa ‘ALEX’ tampil dengan huruf tebal, sedangkan yang lain tampil normal? Untuk membuat tampilan seperti ini anda bisa menggunakan logika IF, yakni jika nama siswa = “alex”, tambahkan tag **<b>** dan **</b>**. Jika bukan alex, tampilkan secara normal.

Selanjutnya, saya akan membahas tentang beberapa fungsi PHP untuk memproses halaman HTML.

# 16. HTML Entity dan Include Function

Bab kali ini masih berkaitan dengan fungsi-fungsi bawaan PHP. Saya akan mengajak anda untuk mempelajari cara PHP men-generate HTML Entity dan membahas fungsi *include()* untuk memecah file HTML dan PHP.

## 16.1 Sekilas tentang HTML Entity

Di dalam HTML, terdapat beberapa karakter yang memiliki fungsi khusus. Karakter khusus ini tidak bisa ditulis langsung. Sebagai contoh, saya tidak bisa menampilkan tulisan berikut di HTML:

```
"tag <table> digunakan untuk membuat tabel di HTML"
```

Alasannya, tanda (<) dan (>) merupakan karakter khusus HTML yang digunakan sebagai tanda pembuka tag. Kata '<table>' dalam kalimat diatas akan diproses oleh web browser sebagai tag tabel, bukan sebagai teks biasa. Hasil yang akan ditampilkan adalah:

```
"tag digunakan untuk membuat tabel di HTML"
```

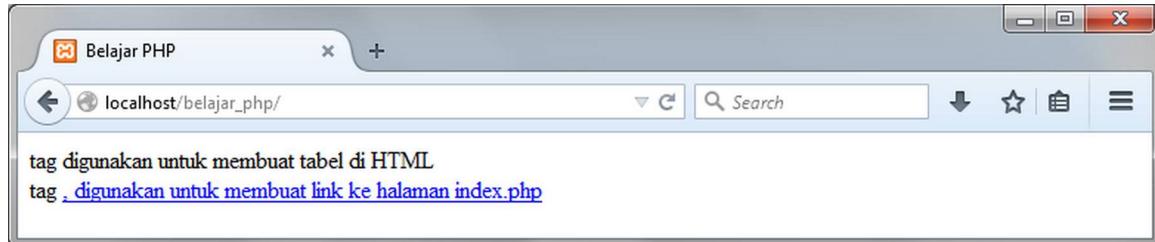
Masalah seperti ini juga terjadi ketika saya menampilkan teks tersebut dari PHP. Berikut contohnya:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <meta charset="UTF-8">
5  <title>Belajar PHP</title>
6  </head>
7  <body>
8  <?php
9  $kalimat1="tag <table> digunakan untuk membuat tabel di HTML";
10 echo $kalimat1;
11
12 echo "<br>";
13
14 $kalimat2="tag <a href=\"index.php\">, digunakan untuk membuat
15         link ke halaman index.php";
```

```

16 echo $kalimat2;
17 ?>
18 </body>
19 </html>

```



Gambar: Karakter ‘<’ dan ‘>’ tidak bisa ditampilkan di HTML

Agar karakter “<” dan “>” bisa ditampilkan seperti biasa, kita harus mengubahnya menjadi **HTML Entity**. *HTML Entity* adalah cara penulisan karakter dengan tanda khusus. Sebagai contoh, tanda ( < ) bisa diganti menjadi: ( &lt; ), tanda ( > ) menjadi: ( &gt; ), dan tanda ( & ) menjadi ( &amp; ).



Penjelasan lebih lengkap tentang **HTML Entity** telah saya bahas di eBook **HTML Uncover**. Anda juga bisa membacanya di tutorial duniaIlkom: [Cara Menampilkan Karakter Khusus HTML](#)<sup>1</sup>.

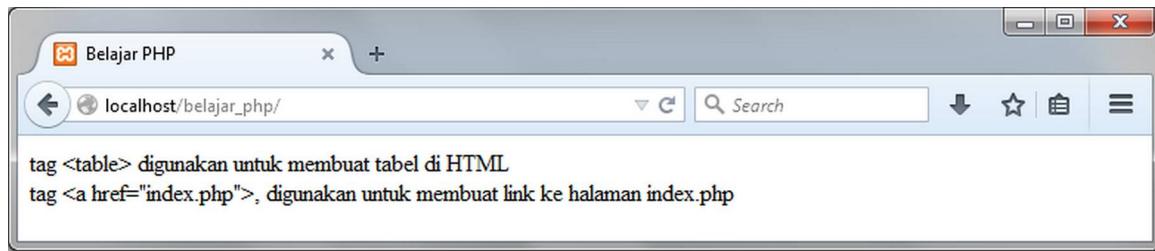
Contoh kita sebelumnya akan tampil baik jika saya mengubah karakter khusus tersebut menjadi **HTML Entity**:

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar PHP</title>
6 </head>
7 <body>
8 <?php
9   $kalimat1="tag &lt;table&gt; digunakan untuk membuat tabel di HTML";
10  echo $kalimat1;
11
12  echo "<br>";
13
14  $kalimat2="tag &lt;a href="index.php"&gt;, digunakan untuk
15      membuat link ke halaman index.php";
16  echo $kalimat2;
17 ?>
18 </body>
19 </html>

```

<sup>1</sup><http://www.duniaIlkom.com/tutorial-text-html-cara-menampilkan-dan-memasukkan-karakter-khusus-ke-dalam-html/>



Gambar: Karakter ‘<’ dan ‘>’ sukses ditampilkan menggunakan HTML Entity

Daripada mengubah manual karakter-karakter khusus ini, PHP menyediakan fungsi **htmlspecialchars()** dan **htmlentities()**.

## 16.2 Function htmlspecialchars()

Fungsi **htmlspecialchars()** bisa digunakan untuk mengkonversi 4 karakter khusus yang sering menjadi ‘masalah’. Karakter tersebut adalah:

- ‘&’ (ampersand) akan dikonversi menjadi ‘&amp;’
- “” (double quote) akan dikonversi menjadi ‘&quot;’
- ‘<’ (less than) akan dikonversi menjadi ‘&lt;’
- ‘>’ (greater than) akan dikonversi menjadi ‘&gt;’

Berikut contoh penggunaannya:

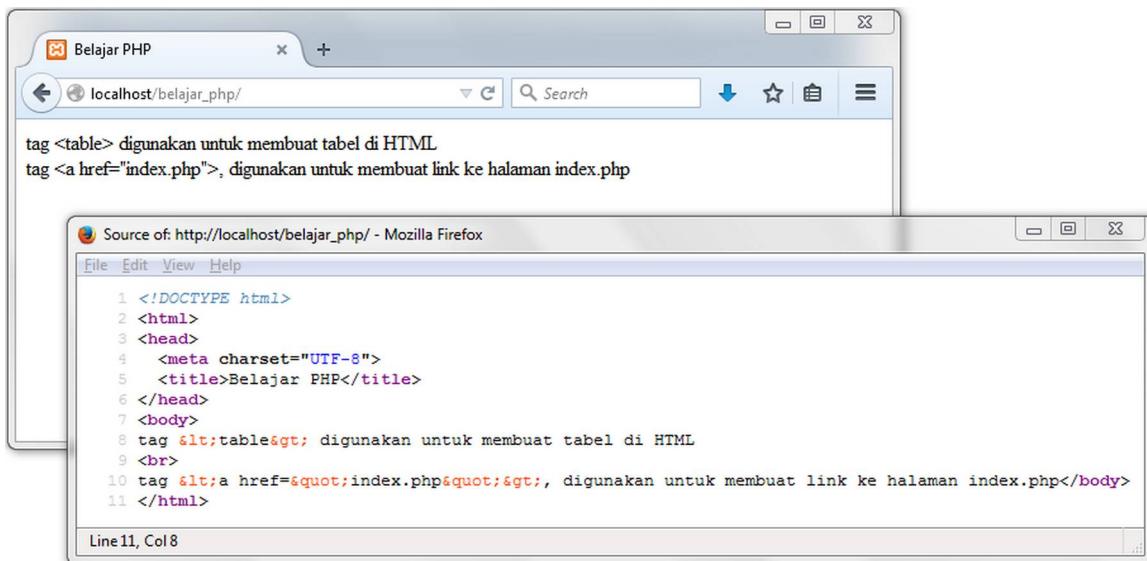
```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6  </head>
7  <body>
8  <?php
9      $kalimat1=htmlspecialchars("tag <table> digunakan untuk membuat
10                      tabel di HTML");
11  echo $kalimat1;
12
13  echo "<br>";
14
15  $kalimat2=htmlspecialchars("tag <a href=\"index.php\">, digunakan
16                      untuk membuat link ke halaman index.php");
17  echo $kalimat2;
18 ?>
19 </body>
20 </html>

```

Agar hasil dari fungsi `htmlspecialchars()` ini dapat terlihat, kita harus memeriksanya dari *source code* HTML yang dijalankan oleh web browser. Ini karena karakter ( < ) dan ( &lt; ) sama-sama ditampilkan sebagai ( < ).

Berikut penampakan dari source code web browser:



Gambar: Fungsi `htmlspecialchars()` akan mengkonversi karakter khusus HTML

Seperti yang terlihat, karakter ( < ), ( > ) dan ( & ) secara otomatis diubah menjadi karakter HTML entity.

## 16.3 Function `htmlspecialchars_decode()`

Walaupun jarang digunakan, tersedia fungsi untuk membalik karakter *HTML entity* kembali ke karakter asalnya, yakni dengan fungsi `htmlspecialchars_decode()`. Berikut contoh penggunaanya:

```

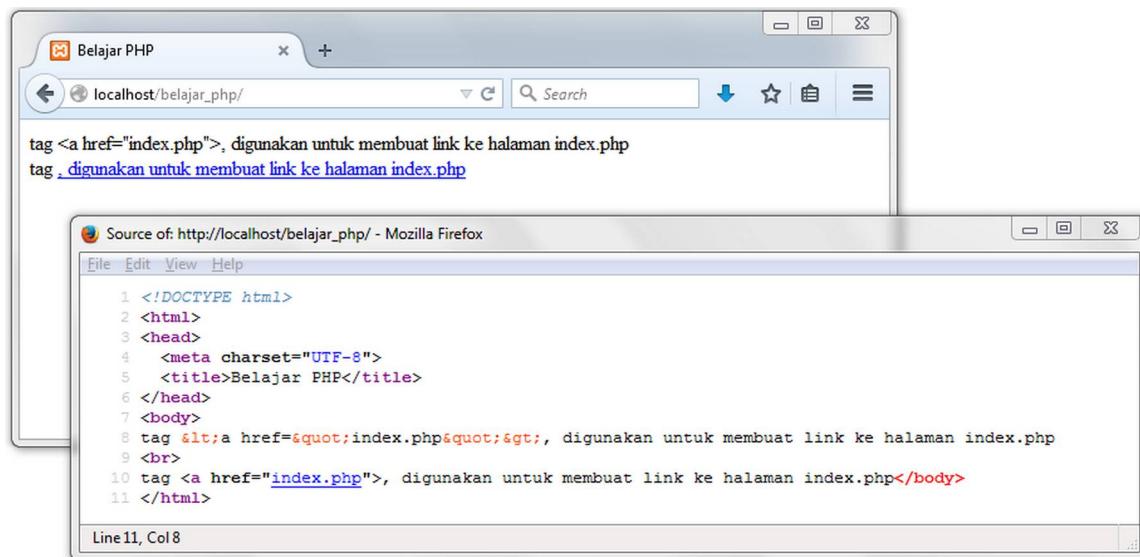
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6  </head>
7  <body>
8  <?php
9      $kalimat1=htmlspecialchars("tag <a href=\"index.php\">, digunakan untuk
10                                membuat link ke halaman index.php");
11      echo $kalimat1;
12
13      echo "<br>";
14

```

```

15 $kalimat1_decode=htmlspecialchars_decode($kalimat1);
16 echo $kalimat1_decode;
17 ?>
18 </body>
19 </html>

```



Gambar: Fungsi htmlspecialchars\_decode() akan mengembalikan HTML Entity ke karakter asalnya

Terlihat bahwa fungsi **htmlspecialchars\_decode()** akan menkonversi HTML Entity menjadi karakter ‘biasa’.

## 16.4 Function htmlentities()

Selain fungsi **htmlspecialchars()**, PHP juga menyediakan fungsi **htmlentities()**. Keduanya sama-sama bertujuan untuk mengkonversi karakter khusus HTML menjadi *HTML Entity*. Jadi dimana perbedaannya?

Fungsi **htmlspecialchars()** hanya akan mengkonversi 4 karakter saja, yakni ( & ), ( < ), ( > ) dan ( “ ). Sedangkan fungsi **htmlentities()** akan mengubah seluruh karakter khusus HTML. Berikut contoh perbedaannya:

```

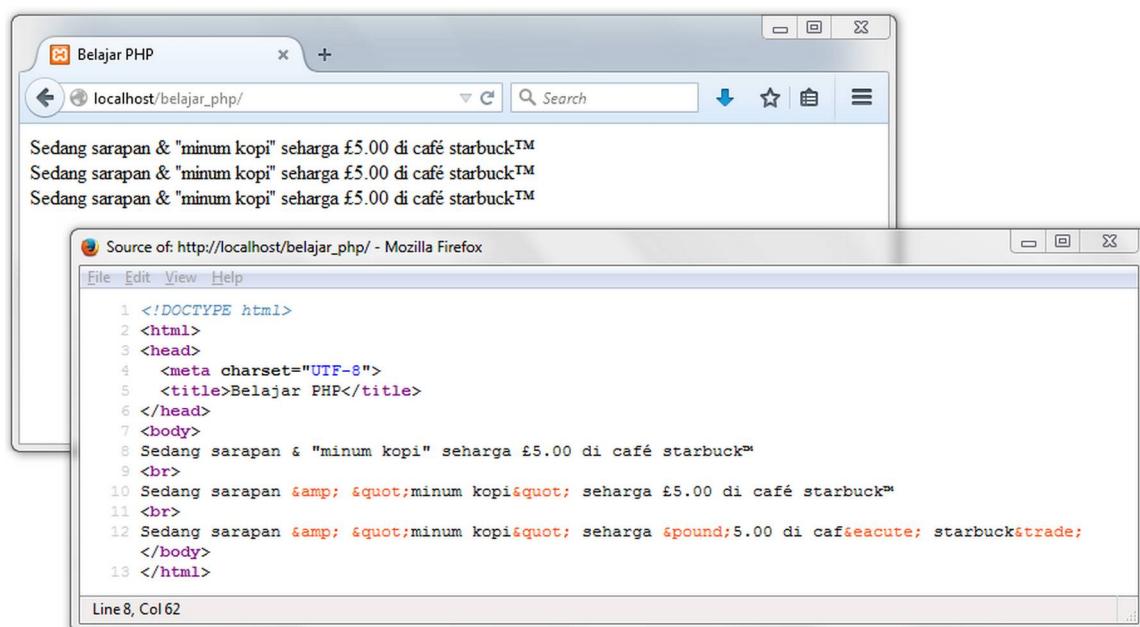
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar PHP</title>
6 </head>
7 <body>
8 <?php
9   $kalimat="Sedang sarapan & \"minum kopi\" seharga £5.00 di café starbuck™";

```

```

10 echo $kalimat;
11 echo "<br>";
12
13 echo htmlspecialchars($kalimat);
14 echo "<br>";
15
16 echo htmlentities($kalimat);
17 ?>
18 </body>
19 </html>

```



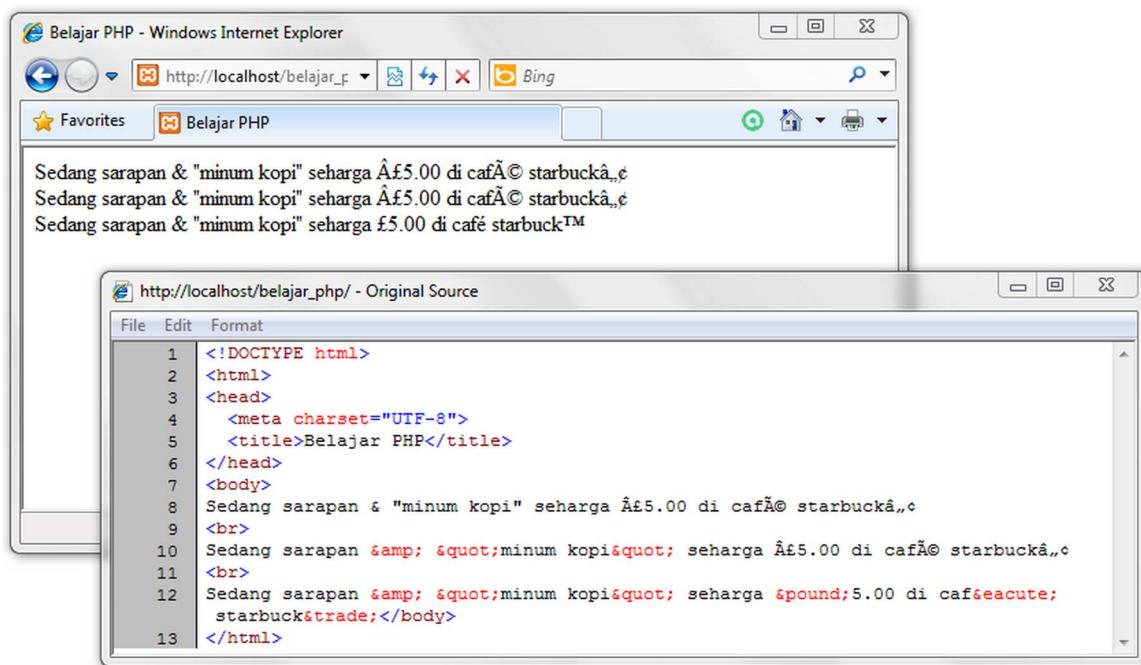
Gambar: Perbedaan dari fungsi htmlspecialchars() dengan htmlentities()

Dapatkah anda membedakan ketiga kalimat ini?

- **Kalimat pertama** ditampilkan ‘apa adanya’.
- **Kalimat kedua** adalah hasil dari fungsi **htmlspecialchars()**, terlihat bahwa karakter ( “ ) dan ( & ) dikonversi menjadi *HTML entity*.
- **Kalimat ketiga** adalah hasil dari fungsi **htmlentities()**, kali ini selain karakter ( “ ) dan ( & ), karakter khusus lain seperti ( £ ), ( é ) dan ( ™ ) juga akan dijadikan *HTML entity*.

Tapi, kenapa kita harus repot-repot menggunakan fungsi ini? Bukankah ketiganya tampil sama?

Ini karena web browser *Mozilla Firefox* yang saya gunakan cukup update dan bisa menampilkan karakter-karakter khusus ini secara langsung. Untuk web browser lawas dan belum mendukung karakter Unicode UTF 8, hasilnya akan tampak berbeda:



Gambar: Tampilan di web browser Internet Explorer 8

Tampilan diatas saya ambil dari web browser *Internet Explorer 8* dengan sedikit pengubahan settingan *Charater Encoding*. Perhatikan bagaimana kalimat pertama dan kedua tampil ‘kacau’. Ini karena IE tidak bisa membaca karakter khusus yang ditulis langsung. Kasus seperti ini memang cukup jarang, karena mayoritas web browser sudah update dan mendukung karakter khusus yang ditulis langsung.

**i** Jika anda bertanya bagaimana cara saya menulis karakter ( € ), ( é ) dan ( ™ ), karena tidak bisa diketik dari keyboard, saya mencarinya dari situs [ev.w3.org](http://ev.w3.org)<sup>2</sup>. Di halaman tersebut terdaftar sebagian besar karater HTML Entity dan cara penulisannya.

Karakter seperti ( £ ), ( é ) dan ( ™ ) dikenal juga sebagai “high ASCII character”, karena karakter ini tidak ditemukan di dalam daftar 128 karakter ASCII ‘normal’.

## 16.5 Function `html_entity_decode()`

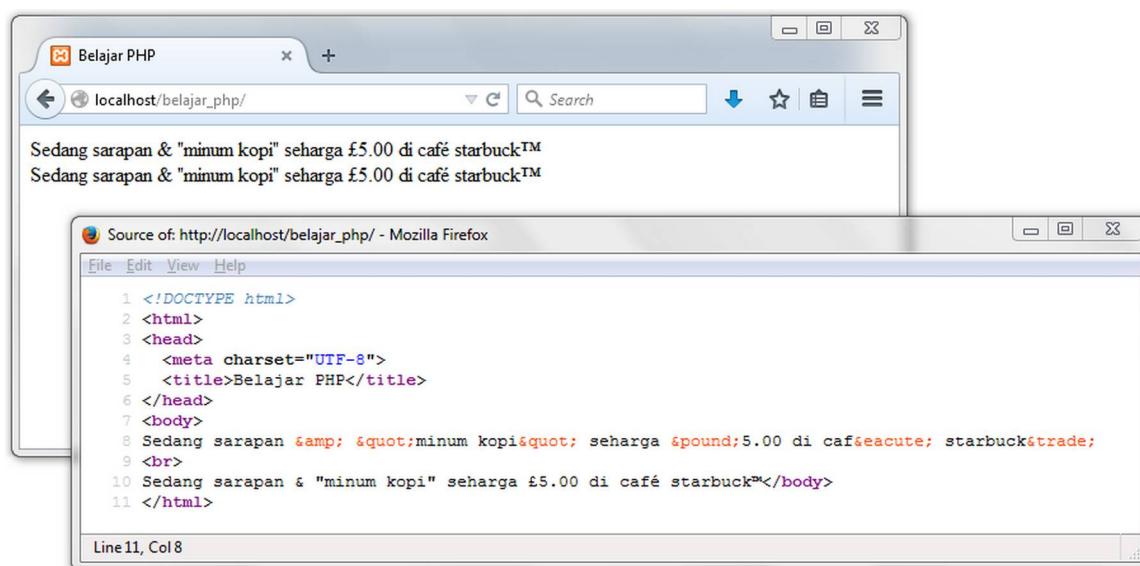
Fungsi `htmlentities()` juga memiliki pasangan fungsi `html_entity_decode()` yang bertujuan untuk mengembalikan karakter yang sudah di konversi kembali ke bentuknya semula:

<sup>2</sup><http://dev.w3.org/html5/html-author/charref>

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6  </head>
7  <body>
8  <?php
9      $kalimat="Sedang sarapan & \"minum kopi\" seharga £5.00 di café starbuck™";
10     $kalimat_encode=htmlentities($kalimat);
11
12     echo $kalimat_encode;
13     echo "<br>";
14
15     $kalimat_dencode=html_entity_decode($kalimat);
16     echo $kalimat_dencode;
17 ?>
18 </body>
19 </html>

```



Gambar: Fungsi `html_entity_decode()` untuk mengembalikan karakter HTML Entity

Jadi, yang mana sebaiknya digunakan untuk memproses karakter-karakter khusus HTML? Dari referensi<sup>3</sup> yang saya temukan, fungsi `htmlspecialchars()` sebenarnya sudah cukup, karena umumnya hanya karakter ( & ), ( < ), ( > ) dan ( " ) saja yang perlu dikonversi menjadi HTML Entity. Karakter khusus lain biasanya sudah bisa ditampilkan langsung oleh web browser.

<sup>3</sup><http://stackoverflow.com/questions/46483/htmlentities-vs-htmlspecialchars>

## 16.6 Memecah Halaman HTML

Apabila anda pernah membuat website dengan HTML dan CSS saja (tanpa PHP), tentunya akan menyadari bahwa dalam setiap halaman, terdapat bagian yang selalu sama dan diulang-ulang, yakni bagian seperti *header*, *sidebar*, dan *footer*. Pada tiap-tiap halaman, yang bertukar hanyalah konten utamanya. Selain itu, jika kita ingin menambah 1 menu, setiap halaman juga harus diubah satu-persatu.

PHP menyediakan solusi untuk hal ini, yaitu dengan memecah atau membagi halaman menggunakan fungsi **include()** dan **required()**.

## 16.7 Function include()

Fungsi **include()** digunakan untuk memasukkan kode program dari file lain ke dalam halaman saat ini. Agar lebih mudah dipahami, saya akan membuat contoh kasusnya. Perhatikan kode HTML berikut:

index.php

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <meta charset="UTF-8">
5  <title>Belajar PHP</title>
6  <style>
7  /* kode CSS disini */
8  </style>
9  <script>
10 // kode JavaScript disini
11 </script>
12 </head>
13 <body>
14
15 <!-- Header dari website-->
16 <header>
17 <h1>Menu Navigasi</h1>
18 <h2>Logo</h2>
19 </header>
20
21 <article>
22 <p>Konten Halaman Home</p>
23 </article>
24
25 <!-- Footer dari website-->
26 <footer>
27 <p>Copyright 2016 DuniaIlkom</p>
```

```
28 </footer>
29
30 </body>
31 </html>
```

---

Kode diatas merupakan contoh struktur halaman HTML ‘lengkap’. Lengkap disini maksudnya sudah memiliki bagian *header*, *logo*, *footer*, serta tag `<style>` dan `<script>` untuk tempat kode CSS dan JavaScript (walaupun hanya berupa kerangka saja).

Misalkan kode ini saya gunakan untuk membuat halaman `index.php`. Tentunya halaman lain seperti `artikel.php`, `shop.php`, atau `contact_us.php` juga menggunakan struktur yang sama. Yang akan berubah hanyalah bagian ‘konten’ atau isi dari website. Bagian *header* dan *footer* akan selalu tetap untuk tiap halaman.

Daripada menulis ulang bagian *header* dan *footer* ini, saya bisa ‘melepasnya’ dari halaman `index.php` menjadi bagian terpisah kemudian disatukan kembali menggunakan PHP.

Langkah pertama adalah memisahkan kode HTMLnya. Karena website yang saya rancang akan memiliki struktur header yang sama, saya bisa memisahkan bagian atas halaman ke dalam sebuah file: `header.html`. Isi dari file `header.html` ini adalah sebagai berikut:

`header.html`

---

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar PHP</title>
6   <style>
7     /* kode CSS disini */
8   </style>
9   <script>
10    // kode JavaScript disini
11 </script>
12 </head>
13 <body>
14
15 <!-- Header dari website-->
16 <header>
17   <h1>Menu Navigasi</h1>
18   <h2>Logo</h2>
19 </header>
```

---

Kemudian, saya akan memisahkan bagian *footer* kedalam file `footer.html`. Isinya adalah:

**footer.html**


---

```

1 <!-- Footer dari website-->
2 <footer>
3 <p>Copyright 2016 DuniaIlkom</p>
4 </footer>
5
6 </body>
7 </html>

```

---

Sekarang, saya sudah memiliki 3 file: **index.php**, **header.html**, dan **footer.html**. Di halaman **index.php**, saya bisa menuliskan kode berikut:

**index.php**


---

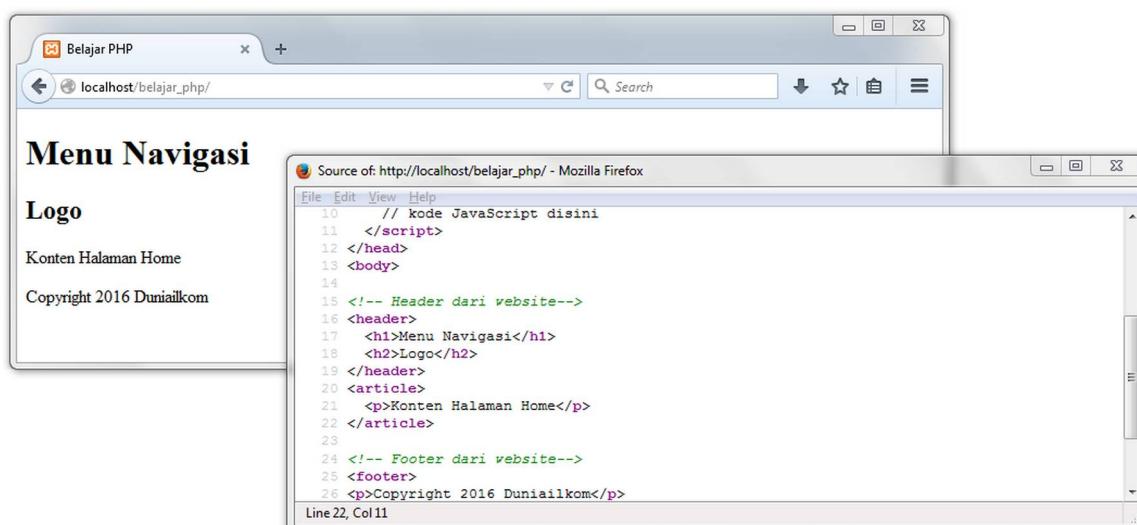
```

1 <?php
2     include("header.html");
3 ?>
4
5 <article>
6     <p>Konten Halaman Home</p>
7 </article>
8
9 <?php
10    include("footer.html");
11 ?>

```

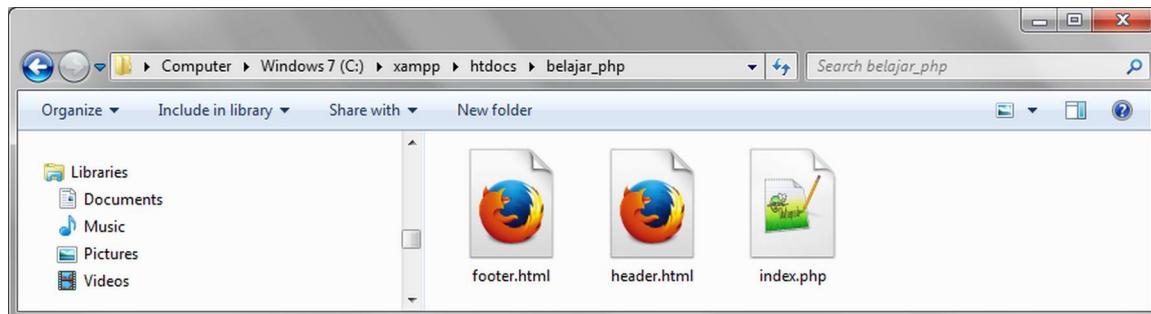
---

Ketika dijalankan, tampilan **index.php** tampak normal, lengkap dengan bagian header dan footer:



Gambar: Tampilan halaman index.php

Inilah kegunaan dari fungsi **include()** PHP. **include()** berfungsi untuk memasukkan file lain ke dalam halaman PHP. Argumen dari fungsi ini adalah alamat dan nama file yang ingin diinput. Alamat ini harus sesuai dengan struktur folder. Misalnya dalam contoh diatas, file **header.html** dan file **footer.html** harus berada di dalam folder yang sama dengan file **index.php**.



Gambar: Struktur folder belajar\_php

Secara teknis, kita bisa menggunakan **alamat absolut** seperti: “<http://www.duniaIlkom/header.html>” atau **alamat relatif** seperti “`folderku/filehtml/header.html`”. Argumen ini harus ditulis dalam tanda kutip (tipe data *string*).

Selain itu, perhatikan bahwa file yang saya *include* adalah sebuah file HTML (**header.html** dan file **footer.html**). Saya menggunakan contoh file HTML untuk menunjukkan bahwa fungsi **include()** juga bisa bekerja untuk file HTML.

Pada umumnya file yang di-*include* adalah file PHP, karena besar kemungkinan kita juga ingin menyisipkan kode PHP di bagian *header* dan *footer* ini.

Silahkan buka kembali file **header.html**, kemudian tambahkan sedikit kode PHP:

#### header.php

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6      <style>
7          /* kode CSS disini */
8      </style>
9      <script>
10         // kode JavaScript disini
11     </script>
12 </head>
13 <body>
14
15 <!-- Header dari website-->
16 <header>
17 <?php
18     echo "<h1>Menu Navigasi</h1>";
19     echo "<h2>Logo</h2>";

```

```
20 ?>
21 </header>
```

---

Karena terdapat kode PHP, file ini harus disimpan sebagai file PHP: **header.php**.

Begitu juga untuk halaman **footer.html**, saya akan menambahkan kode PHP:

#### footer.php

```
1 <!-- Footer dari website-->
2 <footer>
3 <?php
4     echo "<p>Copyright 2016 DuniaIlkom</p>";
5 ?>
6 </footer>
7 </body>
8 </html>
```

---

Kali ini save sebagai **footer.php**.

Tentu saja saya juga harus mengubah kode di file **index.php** agar men-include file **header.php** dan **footer.php** (karena kedua file ini sebelumnya merupakan file HTML):

#### index.php

```
1 <?php
2     include("header.php");
3 ?>
4
5 <article>
6     <p>Konten Halaman Home</p>
7 </article>
8
9 <?php
10    include("footer.php");
11 ?>
```

---

Dengan memecah kode-kode yang ada, saya bisa menggunakan metode yang sama untuk membuat file-file lain, seperti halaman **kategori.php**:

**kategori.php**

---

```
1 <?php
2     include("header.php");
3 ?>
4
5 <article>
6     <p>Konten Halaman Kategori</p>
7 </article>
8
9 <?php
10    include("footer.php");
11 ?>
```

---

Jauh lebih efisien daripada harus mencopy seluruh bagian *header* dan *footer* ke tiap-tiap halaman. Keuntungan lain, jika saya ingin menukar gambar logo, saya tinggal mengubah file *header.php*. Halaman lain yang men-include *header.php* juga otomatis berubah. Tanpa menggunakan fitur include ini, saya terpaksa harus mengubah satu-satu halaman yang ada.

PHP juga tidak membatasi berapa banyak file yang di include. Untuk halaman yang lebih kompleks, saya bisa membuat seperti ini:

**index.php**

---

```
1 <?php
2     include("header.php");
3     include("menu_navigasi.php");
4     include("sidebar.php");
5 ?>
6
7 <article>
8     <p>Konten Halaman Home</p>
9 </article>
10
11 <?php
12    include("iklan.php");
13    include("footer.php");
14 ?>
```

---

Fungsi **include()** juga sering digunakan untuk menginput kode PHP yang berisi **function()**. Sebagai contoh, saya akan membuat sebuah file *function.php*. Isi dari file ini adalah:

**function.php**


---

```

1 <?php
2 function salam($nama){
3   return "Selamat Siang $nama";
4 }
5 ?>

```

---

File **function.php** berisi sebuah fungsi **salam()** sederhana. Dalam prakteknya, file ini bisa berisi puluhan atau ratusan fungsi-fungsi siap dipakai untuk digunakan oleh halaman-halaman lain.

Di halaman **index.php** saya bisa mengakses fungsi **salam()** tanpa harus menulisnya lagi. Caranya, dengan men-include file **function.php** ke dalam file **index.php**:

**index.php**


---

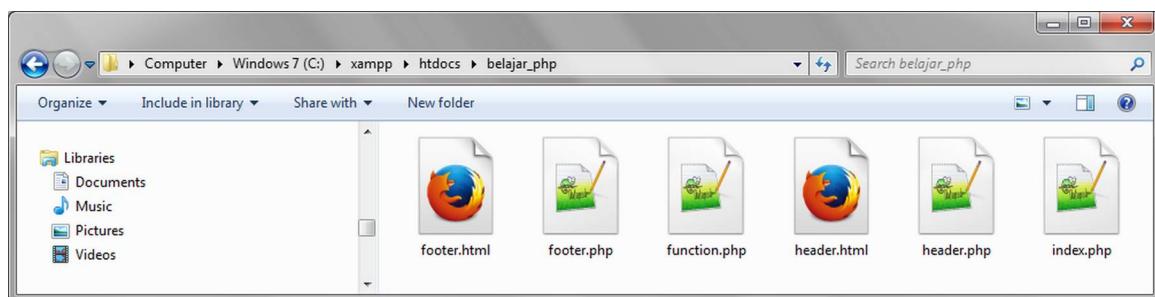
```

1 <?php
2   include("function.php");
3   include("header.php");
4 ?>
5
6 <article>
7   <p>Konten Halaman Home</p>
8 </article>
9
10 <?php
11   echo salam("Ronny"); // Selamat Siang Ronny
12
13   include("footer.php");
14 ?>

```

---

Sekarang, setiap halaman yang meng-include **function.php** bisa mengakses fungsi **salam()**.



Gambar: Agar bisa diakses, seluruh file harus ditempatkan pada folder yang sama

Trik paling umum, saya bisa membuat sebuah fungsi untuk koneksi ke database, misalnya fungsi **koneksi()**, dan menyimpannya di halaman **function.php**. Jika ada halaman lain yang ingin mengakses database, tinggal men-include file ini dan langsung bisa terhubung ke database.

Mari kita coba, silahkan buka kembali file **function.php**, lalu tambahkan fungsi **koneksi()**:

**function.php**

---

```
1 <?php
2 function salam($nama){
3     return "Selamat Siang $nama";
4 }
5
6 function koneksi($user){
7     return "Koneksi ke MySQL dengan user $user Berhasil";
8 }
9 ?>
```

---

Fungsi **koneksi()** ini sebenarnya hanya mengembalikan string seperti fungsi **salam()**. Fungsi asli untuk koneksi ke database akan kita bahas di bab lain. Ini semata-mata agar anda bisa melihat konsep penggunaan **include()**.

Di halaman **index.php**, saya bisa menulis kode berikut:

**index.php**

---

```
1 <?php
2     include("function.php");
3     include("header.php");
4 ?>
5
6 <article>
7     <p>Konten Halaman Home</p>
8 </article>
9
10 <?php
11     echo koneksi("root"); // Koneksi ke MySQL dengan user root Berhasil
12
13     include("footer.php");
14 ?>
```

---

## 16.8 Function **include\_once()**

Selain fungsi **include()**, PHP juga menyediakan fungsi **include\_once()**. Perbedaan mendasar pada fungsi **include\_once()** adalah, fungsi ini akan memastikan file yang di include hanya 1 kali. Perhatikan contoh kasus berikut:

**index.php**

---

```
1 <?php
2     include("function.php");
3     include("header.php");
4 ?>
5
6 <article>
7     <p>Konten Halaman Home</p>
8 </article>
9
10 <?php
11     include("function.php"); // Fatal error: Cannot redeclare salam()
12     echo koneksi("root");
13
14     include("footer.php");
15 ?>
```

---

Kode program diatas akan menghasilkan error karena saya meng-include file **function.php** sebanyak 2 kali. Jika anda masih ingat, di dalam file **function.php**, terdapat kode program yang mendefenisikan fungsi **salam()** dan fungsi **koneksi()**.

Ketika PHP memproses fungsi **include("function.php")** pertama, fungsi ini akan ‘membawa’ defenisi fungsi **salam()** dan fungsi **koneksi()** ke halaman **index.php**.

Ketika pemanggilan fungsi **include("function.php")** yang kedua, isinya juga akan mendefenisikan kembali fungsi **salam()** dan fungsi **koneksi()**. Di dalam PHP, kita tidak bisa mendefenisikan 2 buah fungsi dengan nama yang sama. Oleh karena inilah PHP akan mengeluarkan **error**.

Tetapi jika saya menggunakan fungsi **include\_once()**, tidak akan terjadi error:

**index.php**

---

```
1 <?php
2     include_once("function.php");
3     include("header.php");
4 ?>
5
6 <article>
7     <p>Konten Halaman Home</p>
8 </article>
9
10 <?php
11     include_once("function.php"); // Hore... tidak error
12     echo koneksi("root");
13
14     include("footer.php");
15 ?>
```

---

Selain meng-*include* sebuah file ke halaman lain, fungsi **include\_once()** akan memeriksa apakah file yang sama sebelumnya telah di-*include*. Jika sudah, tidak perlu di include lagi. Artinya, file yang sama tidak akan di-include lebih dari sekali.

## 16.9 Function required() dan required\_once()

PHP juga memiliki 2 buah fungsi lain yang mirip dengan **include()**, yakni **required()** dan **required\_once()**. Keduanya sama-sama berfungsi untuk menginput file lain ke dalam kode PHP saat ini. Jadi, dimana letak perbedaannya?

Fungsi **required()** akan mengeluarkan *Fatal Error* ketika file yang ingin diinput tidak bisa diakses (atau filenya memang belum ada), sedangkan fungsi **include()** hanya akan meneluarkan *Warning*. Berikut contohnya:

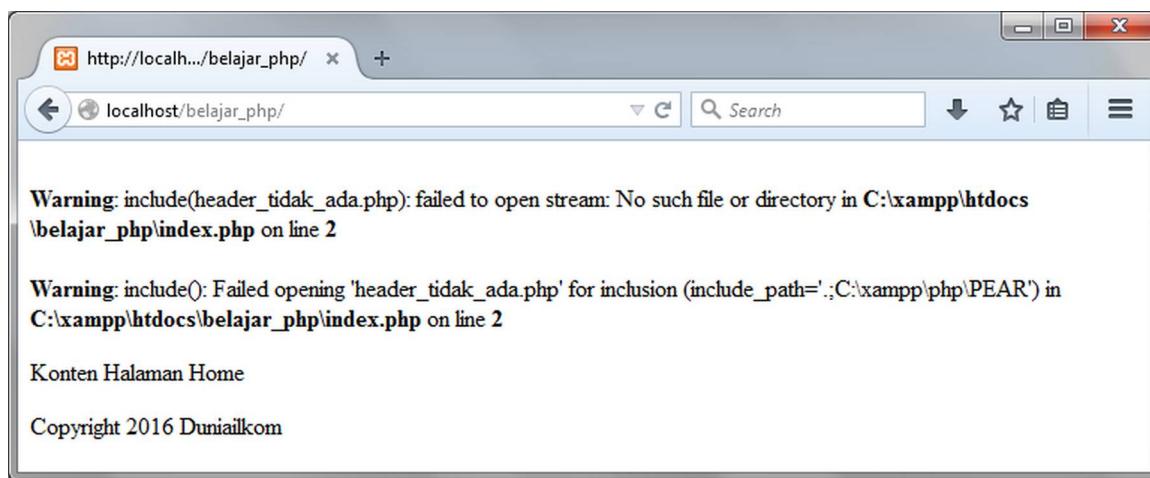
index.php

```

1 <?php
2     include("header_tidak_ada.php"); // error berupa warning
3 ?>
4
5 <article>
6     <p>Konten Halaman Home</p>
7 </article>
8
9 <?php
10    include("footer.php");
11 ?>

```

File yang saya coba include: **header\_tidak\_ada.php** memang tidak ada. Hasil dari kode diatas, PHP akan mengeluarkan 2 error:



Gambar: Error Warning ketika file tidak bisa diinput dengan fungsi include()

Warning adalah tipe error yang rendah. Jika mendapati warning, PHP tetap lanjut untuk menjalankan kode PHP berikutnya. Anda dapat melihat isi konten dan bagian *footer* dari *index.php* tetapi ditampilkan.

Tapi jika saya menggunakan fungsi **require()**, hasilnya sedikit berbeda:

**index.php**

```
1 <?php
2     require("header_tidak_ada.php"); // Fatal error
3 ?>
4
5 <article>
6     <p>Konten Halaman Home</p>
7 </article>
8
9 <?php
10    include("footer.php");
11 ?>
```

Hasilnya adalah:



Gambar: Error Warning ketika file tidak bisa diinput dengan fungsi require()

Selain warning, terdapat “Fatal error”. Fatal error adalah pesan error yang tinggi. Jika menemukan error ini, PHP sepenuhnya akan berhenti memproses halaman.

Fungsi *require()* cocok untuk menginput file yang memang sangat penting, yang jika tidak ditemukan PHP harus berhenti.

Untuk fungsi **require\_once()**, hampir sama dengan **include\_once()**, dimana PHP akan memeriksa terlebih dahulu apakah sebelumnya file sudah diinput atau belum.

---

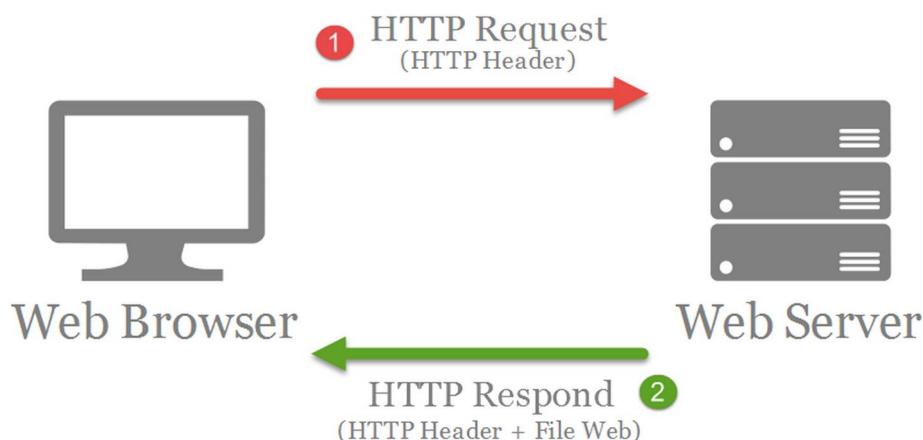
Berikutnya, kita akan membahas tentang **HTTP Header** dan fungsi **header()** di dalam PHP.

# 17. HTTP Header, Output Buffering dan php.ini

Bab kali ini mungkin sedikit teknis, tapi cukup penting untuk dipahami. Kita akan membahas tentang **HTTP Header** dan **Output Buffering**. Selain itu akan dibahas juga tentang cara pengaturan settingan PHP melalui file **php.ini**.

## 17.1 Pengertian HTTP Header

**HTTP Header** adalah sebuah informasi yang dikirim oleh web browser dan web server sebagai sarana komunikasi antara keduanya. Informasi di dalam **HTTP Header** berisi tentang bagaimana cara web browser dan web server mengani data yang dikirim / diminta. Agar pengertian ini lebih mudah dipahami, mari kita bahas sedikit tentang siklus **request-respond** untuk halaman web:



Gambar: Siklus HTTP Request dan HTTP Respond untuk menampilkan halaman web

Ketika kita mengakses sebuah halaman web, web browser akan mengirim **HTTP Request** kepada web server. **HTTP Request** adalah sebuah ‘kegiatan’ yang salah satunya mengirim **HTTP Header**.

Di dalam **HTTP Header** terdapat informasi tentang file apa yang diminta (apakah file HTML, file PHP, file PDF, atau yang lain), serta berbagai informasi tambahan seperti jenis web browser, sistem operasi, alamat IP, dll. Informasi ini dikirim di dalam **HTTP Header** dari web browser ke web server (panah merah dari gambar diatas).

Disisi web server, **HTTP Request** ini akan diproses. Web server mengambil informasi dari **HTTP Header** (yang dikirim web browser) untuk menentukan file apa yang diminta. Setelah ditemukan, Web server kemudian mengirimkannya kembali ke web browser. Proses pengembalian ini dikenal juga dengan sebutan **HTTP Respond** (panah hijau dari gambar sebelumnya).

**HTTP Respond** yang dikirim oleh web server, terdiri dari 2 bagian: **HTTP Header** dan **File Web**. **HTTP Header** kali ini berisi informasi mengenai file web yang dikirim, seperti tipe data, tanggal dikirim, serta berbagai informasi lain seperti nama web server, dan sistem operasi yang digunakan oleh web server.

**File Web** kemudian sampai dan ditampilkan oleh web browser. Jika file yang *di-request* adalah file HTML atau PHP, hasilnya menjadi sebuah halaman web. Tapi jika yang diminta adalah file MP3 atau PDF, web browser akan menampilkan jendela download.

Perhatikan bahwa terdapat 2 buah **HTTP Header** dalam penjelasan saya sebelumnya. Yakni ketika **HTTP Request**, dan pada saat **HTTP Respond**. Dalam bab ini kita hanya fokus kepada **HTTP Header** yang dikirim **dari web server ke web browser**.

Khusus untuk **HTTP Respond**, terdapat satu hal yang sangat penting untuk dipahami: **HTTP Header** harus dikirim sebelum **File Web**.

Sebagai analogi, jika presiden ingin berkunjung ke rumah anda, tentunya akan ada tim *paspampres* yang datang lebih awal. Mereka akan menginformasikan bahwa presiden akan datang pada jam sekian, dengan membawa sekian orang, dan informasi-informasi lain. Tim *paspampres* ini bisa disamakan dengan **HTTP Header** yang datang lebih dulu sebelum file asli dikirim.

Dalam prakteknya, anda mungkin tidak akan menyadari adanya **HTTP Header**, bahkan mungkin tidak pernah mendengarnya. Isi dari **HTTP header** juga ditujukan kepada web browser, bukan pengunjung web. Oleh karena itulah kita tidak pernah melihatnya.

Berikut adalah contoh isi **HTTP Header** dan **File Web** yang dikirim dari web server ke web browser:

```
HTTP/1.1 200 OK
Transfer-Encoding: chunked
Date: Thu, 28 Jan 2016 04:36:25 GMT
Server:"Apache/2.4.17 (Win32) OpenSSL/1.0.2d PHP/5.6.15"
Connection: close
Pragma: public
Expires: Thu, 28 Jan 2016 05:36:25 GMT
Etag: "pub1259380237;gz"
Cache-Control: max-age=3600, public
Content-Type: text/html; charset=UTF-8
Last-Modified: Thu, 28 Jan 2016 03:50:37 GMT
Content-Encoding: gzip
Vary: Accept-Encoding, Cookie, User-Agent
```

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Belajar PHP</title>
</head>
<body>
<!-- ... lanjutan dari kode HTML ... -->
```

Pada bagian paling atas terdapat **status code** HTTP. Angka **200** berarti tidak ada masalah. Jika tertulis **404 page not found**, berarti halaman tidak ditemukan, anda pastinya sudah akrab dengan kode **404** ini. Jika yang tampil adalah Angka **500**, ini berarti ada kesalahan di server (*internal server error*).

Setelah **status code**, terdapat beberapa baris isi dari HTTP header. Format penulisan HTTP header ini adalah “**Label: value**” (label, titik dua, sebuah spasi, dan value).

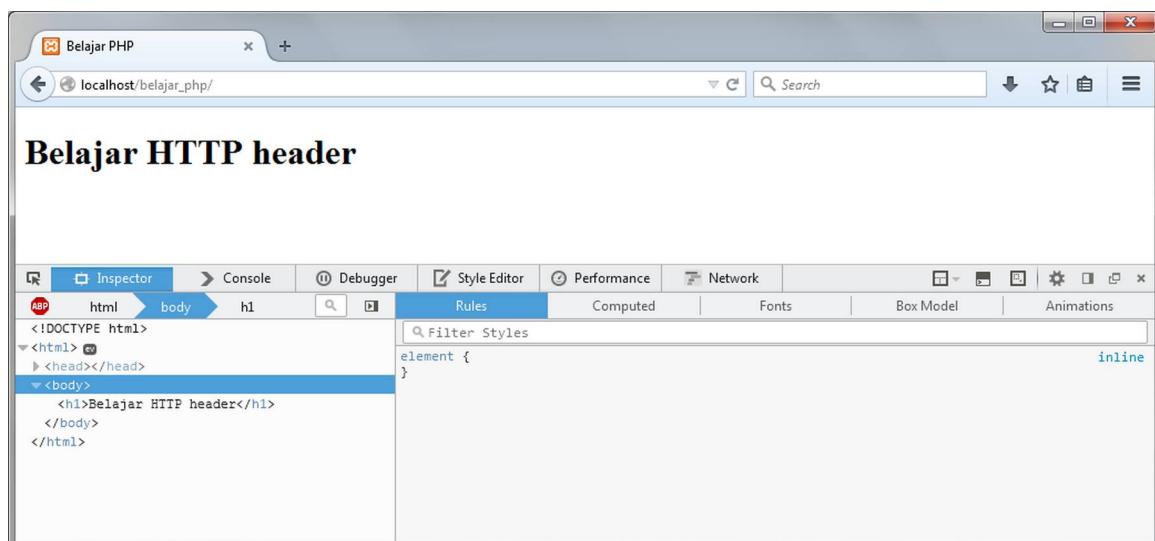
Di baris paling bawah anda bisa melihat kode HTML yang sebenarnya. Inilah kode HTML yang kita lihat sehari-hari. Web browser secara otomatis ‘menyembunyikan’ bagian HTTP header. Sebab itulah kita tidak bisa melihatnya langsung.

## 17.2 Cara Melihat HTTP Header

Sebelum membahas apa fungsi dari mempelajari HTTP header ini, saya ingin mengajak anda untuk melihatnya langsung. Kali ini saya menggunakan web browser **Mozilla Firefox**. Untuk web browser lain seperti *Google Chrome* atau *Opera* caranya hampir sama.

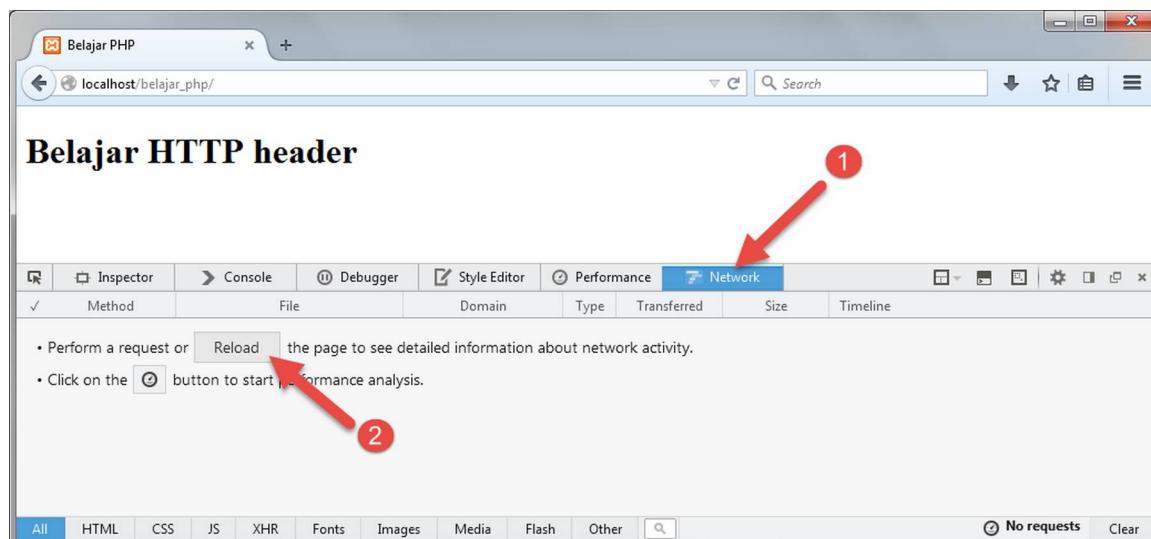
Pertama kali, buka sebuah halaman web. Halaman ini bisa berupa web online atau dari *localhost*. Saya sarankan gunakan halaman dari *localhost* saja agar tampilan HTTP headernya lebih sederhana.

Kemudian buka menu **Web Developer Tools** atau tekan kombinasi tombol **CTRL+SHIFT+I**. Akan tampil jendela **Developer Tools** seperti gambar berikut:



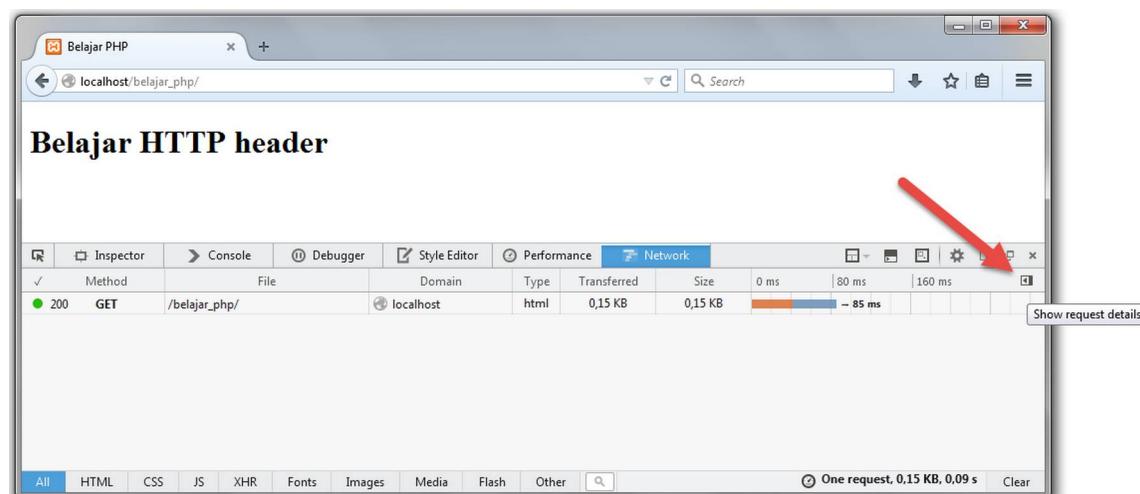
Gambar: Jendela Web Developer Tools di Mozilla Firefox

Pilih tab **Network**, lalu refresh halaman atau tekan tombol “**Reload**”:



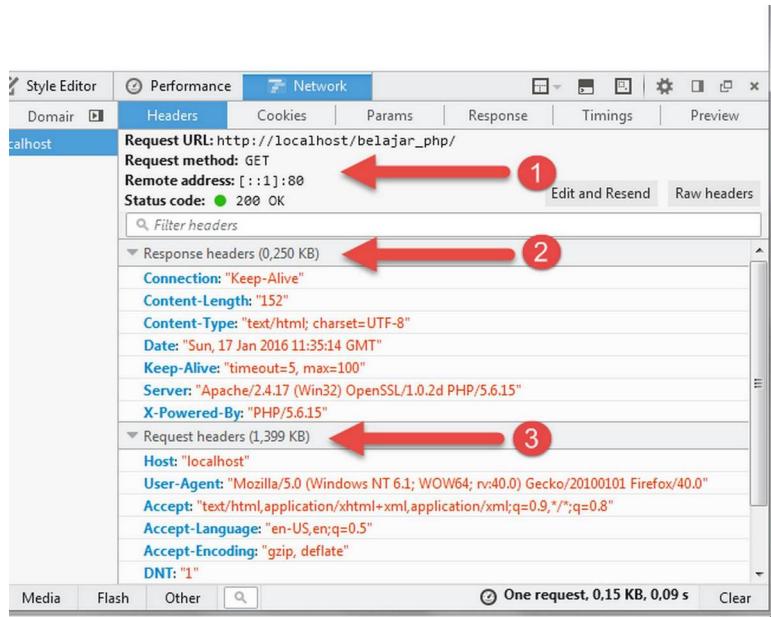
Gambar: Pilih tab Network, lalu klik Reload

Selanjutnya akan tampil sebuah tabel yang diawali dengan angka 200, GET, dst. Nilai ini sebenarnya diambil dari HTTP header. Untuk lebih lengkapnya, klik tombol kecil di sisi paling kanan (show request detail):



Gambar: Klik tombol show request detail

Sekarang, kita bisa melihat isi dari HTTP header:



Gambar: Akhirnya, sebuah HTTP header

Anda mungkin harus memperbesar ukuran jendela **Developer Tools** agar seluruh tampilan bisa terlihat.

Pada bagian paling awal (**nomor 1**) terlihat *status code*, *request method*, serta alamat IP server. Pada bagian tengah (**nomor 2**) adalah isi dari *HTTP header response* (dari web server ke web browser), sedangkan di bagian paling bawah (**nomor 3**) merupakan isi dari *HTTP header request* (dari web browser ke web server).

Saya tidak akan membahas semua isi dari HTTP header. Beberapa cukup jelas seperti tanggal (*Date*), tipe file (*Content-Type*), jenis aplikasi server (*Server*), serta jenis web browser (*User-Agent*). Jika anda tertarik tentang seluruh informasi di HTTP header ini, bisa membacanya di [wikipedia<sup>1</sup>](https://en.wikipedia.org/wiki/List_of_HTTP_header_fields) atau [code.tutsplus.com<sup>2</sup>](http://code.tutsplus.com/tutorials/http-headers-for-dummies%E2%80%93net-8039).

## 17.3 Function header()

Baik, cukup dengan teori seputar **HTTP header**, mari kita masuk ke prakteknya menggunakan PHP. PHP mengizinkan kita untuk mengubah isi HTTP header yang dikirim dari web server melalui fungsi **header()**. Cara penggunaannya, cukup dengan menulis label apa yang ingin diubah.

Sebagai contoh, jika saya ingin mengubah *status code* menjadi **404**, saya bisa menulis sebagai berikut:

<sup>1</sup>[https://en.wikipedia.org/wiki/List\\_of\\_HTTP\\_header\\_fields](https://en.wikipedia.org/wiki/List_of_HTTP_header_fields)

<sup>2</sup><http://code.tutsplus.com/tutorials/http-headers-for-dummies%E2%80%93net-8039>

```

1 <?php
2   header("HTTP/1.1 404 Not Found");
3 ?>

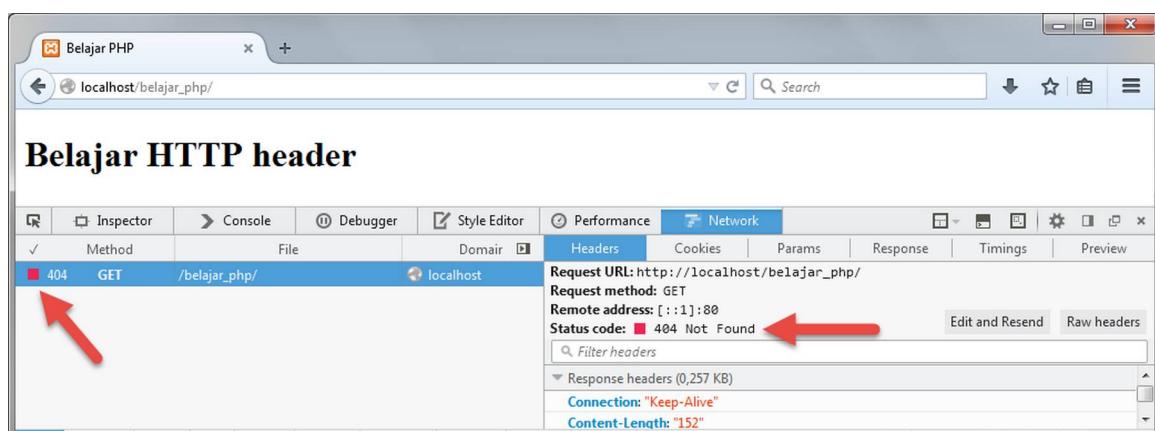
```

Mari kita coba, silahkan jalankan kode berikut dan lihat dari **Web Developer Tools**:

```

1 <?php
2   header("HTTP/1.1 404 Not Found");
3 ?>
4 <!DOCTYPE html>
5 <html>
6 <head>
7   <meta charset="UTF-8">
8   <title>Belajar PHP</title>
9 </head>
10 <body>
11   <h1>Belajar HTTP header</h1>
12 </body>
13 </html>

```



Gambar: Status code halaman berubah menjadi 404

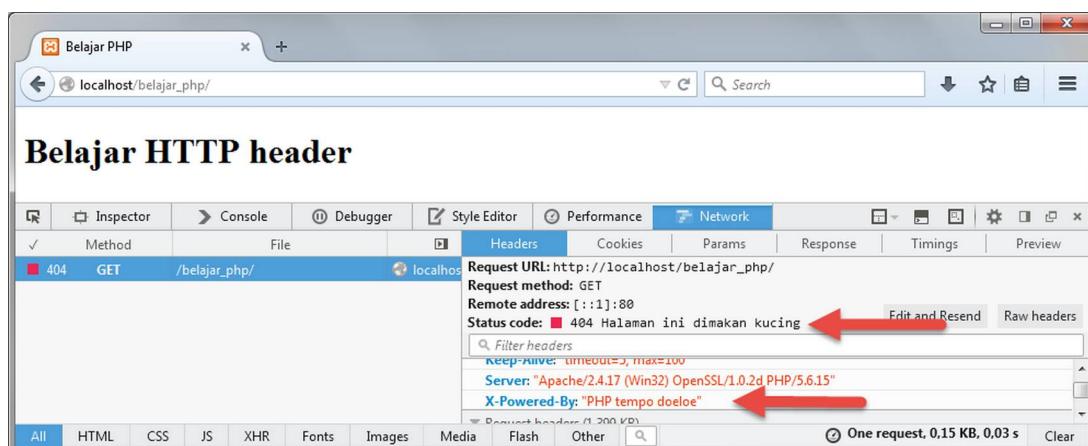
Dapat anda lihat bahwa status codenya berubah menjadi **404**. Walaupun demikian, halaman kita tetap diproses seperti biasa.

Sebagai contoh lain, saya bisa mengubah **HTTP header** seperti berikut ini:

```

1 <?php
2   header("HTTP/1.1 404 Halaman ini dimakan kucing");
3   header("X-Powered-By: PHP tempo doeloe");
4 ?>
5 <!DOCTYPE html>
6 <html>
7 <head>
8   <meta charset="UTF-8">
9   <title>Belajar PHP</title>
10 </head>
11 <body>
12   <h1>Belajar HTTP header</h1>
13 </body>
14 </html>

```



Gambar: Terlihat isi dari HTTP Header juga ikut berubah

Dari sekian banyak label **HTTP header** yang tersedia, terdapat 1 label atau atribut yang paling sering digunakan di dalam PHP, yakni label **“Location”**.

## 17.4 Redirect dengan fungsi header()

Redirect adalah istilah web yang berarti mengalihkan pengunjung ke halaman lain. Fitur seperti ini umum dijumpai pada halaman *login*. Jika masuk dengan username dan password yang benar, kita akan di *redirect* ke halaman *admin.php*, sedangkan jika username dan password tidak cocok, akan di *redirect* ke halaman *error.php*.

Fitur **redirect** ini bisa dibuat melalui fungsi **header()**. Caranya, dengan mengubah label **“Location”** dari HTTP header. Berikut contohnya:

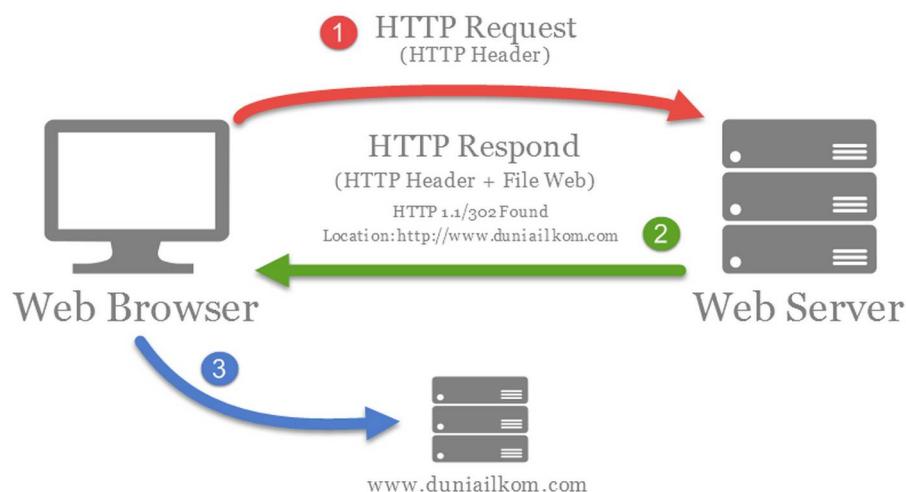
```

1 <?php
2   header("Location: http://www.duniailkom.com");
3 ?>
4 <!DOCTYPE html>
5 <html>
6 <head>
7   <meta charset="UTF-8">
8   <title>Belajar PHP</title>
9 </head>
10 <body>
11   <h1>Belajar HTTP header</h1>
12 </body>
13 </html>

```

Ketika halaman diatas dijalankan, web browser langsung di *redirect* ke situs [www.duniailkom.com](http://www.duniailkom.com). Inilah hasil dari penggunaan fungsi **header("Location: "**).

Secara teknis, yang terjadi adalah: PHP menginstruksikan web server mengirim *status code 302 Redirect* dan **Location: www.duniailkom.com** di dalam **HTTP Header** yang dikirim ke web browser. Ketika mendapati isi HTTP header seperti ini, web browser langsung ‘pindah’ ke alamat yang ditempatkan di bagian **Location**. Berikut ilustrasi dari alur ini:



Gambar: Alur proses redirect menggunakan fungsi **header("Location: http://www.duniailkom.com")**

Yang juga perlu menjadi catatan, penulisan string **Location** harus langsung disambung dengan tanda titik dua. Apabila ditulis seperti ini:

```
header("Location : http://www.duniailkom.com");
```

Fungsi *redirect* tidak akan berjalan. Perhatikan ada spasi tambahan antara “*Location*” dengan tanda titik dua “**:**”.

Selain menggunakan *alamat absolut* seperti contoh diatas, kita juga bisa membuat *alamat relatif* untuk **redirect**, sebagai contoh saya bisa membuat kode berikut:

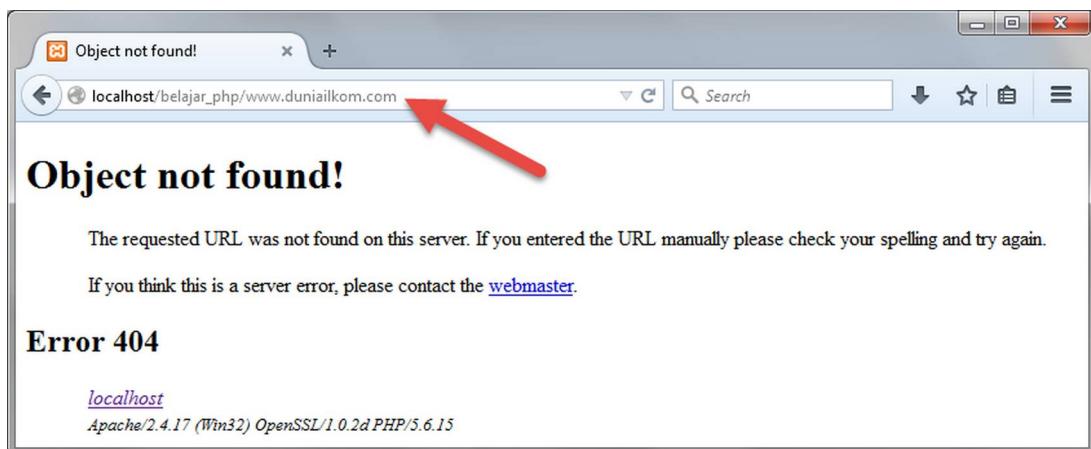
```
1 <?php
2     $password="rahasia";
3
4     if ($password=="rahasia"){
5         header("Location: admin.php");
6     }
7     else{
8         header("Location: error.php");
9     }
10 ?>
11 <!DOCTYPE html>
12 <html>
13 <head>
14     <meta charset="UTF-8">
15     <title>Belajar PHP</title>
16     <!-- lanjutan kode HTML disini -->
```

Jika anda menjalankan kode diatas, akan langsung di *redirect* ke halaman **admin.php**. Ini karena kondisi **if** menghasilkan nilai **true**. Silahkan ubah string pembanding **if** menjadi: **if (\$password=="tidak tau")**, maka halaman akan di redirect ke **error.php**. Kedua halaman ini belum kita buat, jadi memang akan menghasilkan error (halaman yang dituju tidak ditemukan).

Dalam prakteknya, nilai *password* ini biasanya berasal dari form (cara pemrosesan form dengan PHP akan kita bahas di bab terpisah).

Khusus untuk *alamat absolut*, penulisannya harus menyertakan bagian *protocol*, yakni bagian **"http://"**, jika tidak disertakan, ini artinya kita menggunakan alamat relatif. Berikut contoh kasusnya:

```
1 <?php
2     header("Location: www.duniaikom.com");
3 ?>
4 <!DOCTYPE html>
5 <html>
6 <head>
7     <meta charset="UTF-8">
8     <title>Belajar PHP</title>
9     <!-- lanjutan kode HTML disini -->
```



Gambar: Alamat absolute akan disambung ke alamat saat ini

Halaman diatas tidak akan di *redirect* ke situs duniaikom, tetapi disambung ke alamat sekarang. Hasilnya, tentu saja tidak ditemukan alamat [http://localhost/belajar\\_php/www.duniaikom.com](http://localhost/belajar_php/www.duniaikom.com).

Jika anda perhatikan pada beberapa contoh penggunaan fitur *redirect*, halaman web akan langsung berpindah ke tempat lain, padahal di bawahnya tetap terdapat kode HTML. Dalam contoh sebelum ini, saya memiliki teks "<h1>Belajar HTTP header</h1>", yang tidak bisa terlihat, namun sebenarnya teks ini tetap dikirim ke web browser. Jadi, daripada web server tetap mengirim kode yang sebenarnya tidak akan ditampilkan, kita bisa menambahkan fungsi **die()** atau **exit()** dari PHP.

## 17.5 Function die() dan exit()

Fungsi **die()** dan **exit()** sebenarnya tidak berkaitan langsung dengan **HTTP header** maupun fungsi **header()**. Namun fungsi ini memang sering digunakan ketika pembuatan *redirect*.

Fungsi **die()** dan **exit()** digunakan untuk memaksa PHP berhenti memproses kode program. Fungsinya mirip seperti pertintah **break** dalam perulangan, tapi kali ini yang akan berhenti adalah seluruh kode PHP. Jika terdapat kode lain setelah pemanggilan fungsi **die()** atau **exit()**, kode tersebut tidak akan diproses.

Kedua fungsi ini bisa diinput dengan 1 argumen yang akan ditampilkan sebagai 'pesan terakhir' sebelum kode PHP dihentikan.



Fungsi **die()** sebenarnya adalah bentuk lain dari **exit()**. Di dalam PHP Manual, hanya fungsi **exit()** yang dibahas. Bisa dibilang keduanya adalah fungsi identik.

Mari lihat contoh penggunaannya:

```
1 <?php
2     $nama = "joko";
3     if ($nama=="joko") {
4         exit("Maaf, joko tidak boleh masuk");
5         echo "Ah yang bener mas...."; // tidak akan pernah diproses
6     }
7 ?>
```

Dalam kode program diatas, saya membuat sebuah kondisi IF. Jika variabel \$nama berisi string “joko”, kode program akan langsung berhenti, karena terdapat fungsi exit(). Hasil yang akan ditampilkan adalah:

Maaf, joko tidak boleh masuk

String “Ah yang bener mas....” tidak akan pernah diproses karena PHP sudah berhenti terlebih dahulu.

Jika kita tidak ingin menampilkan pesan apa-apa, fungsi exit() bisa ditulis tanpa argumen, atau bahkan tanpa tanda kurung:

```
1 <?php
2     $nama = "andi";
3     if ($nama=="joko") {
4         exit();
5     }
6
7     if ($nama=="andi") {
8         exit;
9     }
10 ?>
```

Fungsi exit() juga bisa ditulis tanpa tanda kurung karena exit() merupakan sebuah *language construct* PHP. **Language construct** adalah sebuah fungsi khusus PHP yang ‘diistimewakan’. Sama seperti perintah echo, break, dan continue yang juga tidak memerlukan tanda kurung, walaupun ketiganya juga merupakan sebuah *function*.

Kembali ke ‘permasalahan’ dengan fungsi header(), kita bisa memaksa PHP untuk segera berhenti jika sudah di *redirect*. Berikut contohnya:

```
1 <?php
2   header("Location: http://www.google.com");
3   die();
4 ?>
5 <!DOCTYPE html>
6 <head>
7   <meta charset="UTF-8">
8   <title>Belajar PHP</title>
9 <!-- lanjutan kode HTML disini -->
```

Ketika kode program diatas dijalankan, web server akan langsung mengirim **HTTP Header** ke web browser untuk segera beralih ke situs www.google.com (*redirect*).

Akan tetapi pada saat itu, sisa halaman web juga ikut dikirim (walaupun tidak akan ditampilkan). Supaya lebih efisien, saya menambahkan fungsi **die()** untuk menghentikan PHP tepat setelah fungsi **header()**, sehingga kode program dibawahnya tidak ikut dikirim ke web browser. Efeknya memang tidak akan terasa, tapi jika kode HTML ini terdiri dari ribuan baris, cukup menyita kapasitas bandwidth web server.

Dalam prakteknya, fungsi **exit()** dan **die()** juga sering digunakan sebagai cara cepat untuk menghentikan kode PHP ketika terjadi suatu masalah. Misalnya jika koneksi ke database gagal, kita bisa memotong kode program pada saat itu juga sehingga PHP tidak akan memproses baris dibawahnya (yang memerlukan database).

## 17.6 Output Buffering

Pada penjelasan di awal bab ini, saya menulis bahwa **HTTP header** harus dikirim lebih dulu sebelum isi web. Di dalam PHP, ini berarti kita **harus** menjalankan fungsi **header()** sebelum menampilkan apapun ke web browser (termasuk sebuah spasi!).

Sebagai contoh, kode program berikut akan menghasilkan error:

```
1 <?php
2   echo "Teks ini sudah dikirim ke web browser";
3   header("Location: http://www.google.com");
4 ?>
5 <!DOCTYPE html>
6 <head>
7   <meta charset="UTF-8">
8   <title>Belajar PHP</title>
9 <!-- lanjutan kode HTML disini -->
```

Hasilnya adalah:



Gambar: Warning: Cannot modify header information – headers already sent by...

Error ini terjadi karena saya menampilkan ‘sesuatu’ dengan perintah **echo** sebelum memanggil fungsi **header("Location: ")**. ‘Sesuatu’ disini bisa berupa karakter apapun, termasuk sebuah spasi:

```

1 <?php
2   header("Location: http://www.google.com");
3 ?>
4 <!DOCTYPE html>
5 <head>
6   <meta charset="UTF-8">
7   <title>Belajar PHP</title>
8 <!-- lanjutan kode HTML disini -->

```

Kode program diatas juga akan menghasilkan error yang sama. Tapi apakah anda melihat ada yang salah? Ini nyatanya tidak terdeteksi, tapi sebelum tag pembukan PHP <?php, terdapat sebuah spasi!

Ketika kita menjalankan fungsi *redirect* PHP, syarat seperti ini harus selalu diingat. Fungsi **header("Location: ")** wajib ditempatkan pada baris paling awal. Akan tetapi, saya bisa menulis kode berikut, dan tidak akan menghasilkan error:

```

1 <?php
2
3   header("Location: http://www.google.com");
4 ?>
5 <!DOCTYPE html>
6 <head>
7   <meta charset="UTF-8">
8   <title>Belajar PHP</title>
9 <!-- lanjutan kode HTML disini -->

```

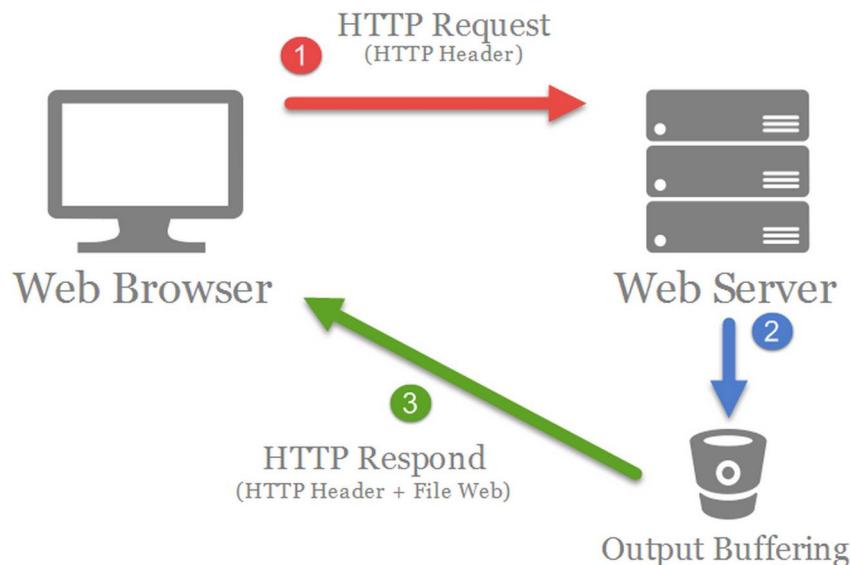
Walaupun saya menambah beberapa spasi, ini tidak menjadi masalah, karena spasi berada di dalam tag PHP. Yang menyebabkan error sebelumnya adalah terdapat spasi **sebelum** tag pembuka php.

*“Tapi mas, sewaktu saya menjalankan kode yang dianggap error sebelumnya, tidak ada masalah tuh, dan langsung redirect ke situs www.google.com). Kenapa ya?”*

Ini terjadi karena settingan **Output Buffering** di versi PHP yang anda gunakan telah aktif.

**Output Buffering** adalah sebuah fitur yang disediakan PHP untuk ‘menampung’ seluruh fungsi yang akan mengubah **HTTP Header**. Salah satu fungsi tersebut adalah *header()* yang kita bahas disini. Jika fitur *output buffering* tidak aktif, seluruh fungsi seperti *header()* wajib hukumnya ditulis di baris paling awal.

**Output Buffering** bisa diibaratkan sebagai “ember” penampung fungsi *header()*. Perhatikan gambar berikut:



Gambar: Ilustrasi penambahan ‘Output Buffering’ dalam alur Request-Respond HTTP

Sekarang, sebelum **HTTP Respond** dikirim ke web browser, web server akan menampungnya ke “ember” *output buffering* terlebih dahulu. Di dalam *output buffering* inilah struktur **HTTP header** disusun dan dirapikan agar kembali berada di posisi paling awal. Dengan demikian, kita bisa mengubah **HTTP header** kapanpun sepanjang kode program PHP.

Pada beberapa versi PHP, fitur *output buffering* telah aktif secara default. Untuk memeriksanya, silahkan jalankan fungsi *phpinfo()*, lalu cari baris “*output\_buffering*”. *Phpinfo* bisa diakses dari menu home XAMPP di alamat <http://localhost><sup>3</sup>, atau bisa juga dengan cara menjalankan kode program berikut:

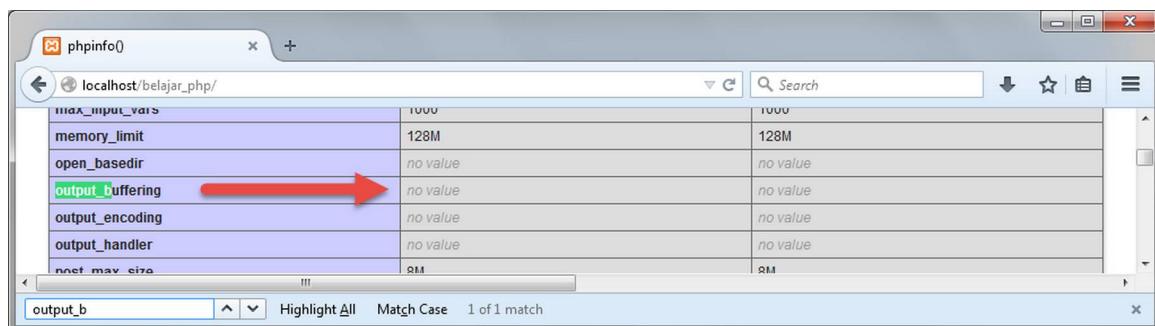
```

1 <?php
2     phpinfo();
3 ?>

```

Kemudian, cari kata “*output\_buffering*”. Gunakan fitur **find** untuk mencarinya (CTRL + F).

<sup>3</sup><http://localhost>



Gambar: Mencari 'output\_buffering' di tampilan phpinfo()

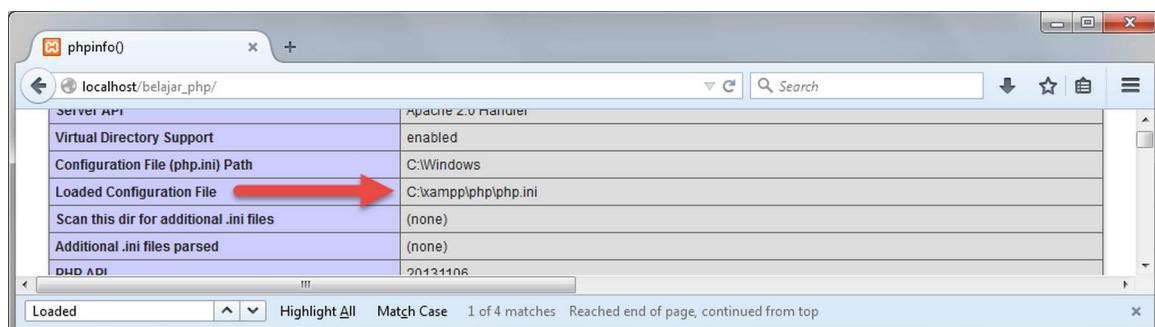
Jika di baris ini tidak ada nilai apa-apa (*no value*), berarti *output buffering* belum aktif. Tapi jika terdapat nilai seperti **4096**, atau **“on”**. Berarti *output buffering* sudah tersedia.

Apabila fitur *output buffering* belum aktif, kita bisa mengaktifkannya dari file **php.ini**.

## 17.7 File Settingan PHP: php.ini

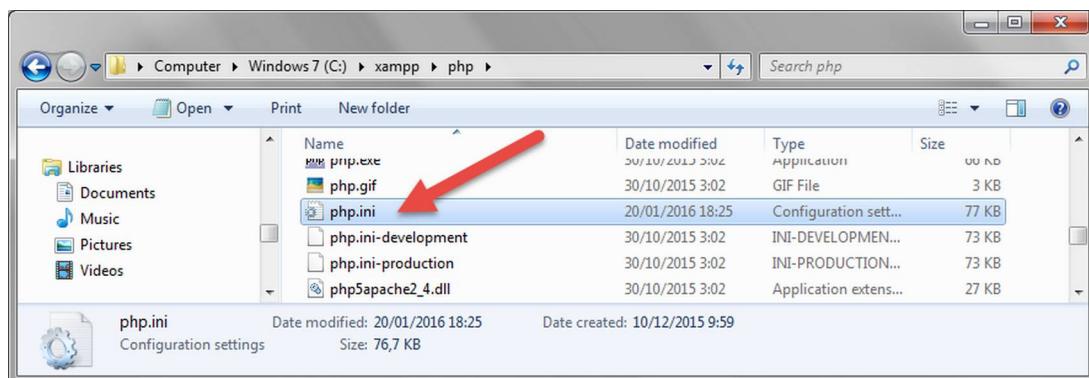
Sampai saat ini, saya belum membahas salah satu file terpenting yang wajib diketahui oleh setiap programmer PHP: file **php.ini**. File **php.ini** berisi seluruh pengaturan PHP. Mulai dari menampilkan pesan error, mengatur lama eksekusi program, hingga mengaktifkan PHP extension.

Kali ini kita akan mencoba mengaktifkan fitur *Output Buffering* dari **php.ini**. Langkah pertama adalah mencari lokasi file tersebut. Masih dari tampilan **phpinfo()**, di bagian atas halaman terdapat nilai **“Loaded Configuration File”**. Anda juga bisa menggunakan fitur **Find** untuk mencarinya.



Gambar: Lokasi dari file php.ini

Isi dari *Loaded Configuration File* merupakan alamat file **php.ini**. Dalam contoh saya, nilainya adalah **C:\xampp\php\php.ini**. Silahkan cari file tersebut menggunakan *Windows Explorer*. Setelah ditemukan, buka menggunakan **Notepad++** atau teks editor yang anda gunakan.



Gambar: Buka file php.ini dengan teks editor

Jika sudah terbuka, anda bisa lihat-lihat sebentar apa saja yang ada di dalam file ini. Seluruh settingan tentang PHP bisa kita ubah dari file ini. Tanda titik koma ( ; ) diawal setiap baris merupakan komentar di dalam **php.ini**. Artinya, apapun yang diawali dengan tanda titik koma ( ; ), berarti tidak aktif.

Untuk mengaktifkan **Output Buffering**, carilah baris “output\_buffering” (gunakan fitur pencarian di teks editor). Anda akan menemukan 3 lokasi dengan teks “output\_buffering”. Carilah yang didepannya tidak memiliki karakter titik koma ( ; ), seperti tampilan berikut:

```

260 ; Default Value: Off
261 ; Development Value: 4096
262 ; Production Value: 4096
263 ; http://php.net/output-buffering
264 output_buffering=Off
265
266 ; You can redirect all of the output of your scripts to a function. For
267 ; example, if you set output handler to "mb output handler", character

```

Gambar: Pengaturan Output Buffering di dalam file php.ini

Untuk mengaktifkan *output buffering*, kita bisa mengisi nilai “on” atau nilai angka seperti “4096”. Angka ini adalah jumlah memory yang disediakan untuk menampung *HTTP header* (dalam satuan byte). 4096 berarti bisa menampung sekitar 4096 karakter. Jika kita memberikan nilai “on”, Berarti tidak ada limit untuk *output buffering*.

Dalam contoh ini saya akan menggunakan nilai **4096**. Silahkan ubah nilai *output\_buffering* menjadi **4096**, seperti gambar berikut:

```

259 ; Note: This directive is hardcoded to Off for the CLI SAPI
260 ; Default Value: Off
261 ; Development Value: 4096
262 ; Production Value: 4096
263 ; http://php.net/output-buffering
264 output_buffering=4096
265

```

Gambar: Mengubah nilai output\_buffering menjadi 4096

Save **php.ini**, kemudian restart Apache web server. Caranya, klik tombol **stop**, lalu klik kembali tombol **start**.

Untuk memeriksa apakah *output buffering* telah aktif, jalankan kembali **phpinfo()**, lalu cari

baris “**output\_buffering**”. Jika sudah berubah menjadi **4096**, berarti fitur output buffering sudah berjalan.

Jika anda mendapati fitur output buffering telah aktif sejak awal, saya menyarankan untuk menghentikannya dengan mengubah nilai “**output\_buffering=Off**” di dalam **php.ini**, lalu jalankan kode program “error” saya sebelumnya. Ini agar anda bisa melihat langsung dampak ketika fitur **output buffering** tidak aktif. Karena untuk website ‘live’, kebanyakan web hosting tidak menyediakan fitur ini.



Mengaktifkan **Output Buffering** sebenarnya akan menambah sedikit beban web server, karena ada proses tambahan yang dilakukan sebelum halaman web dikirim ke web browser. Walaupun begitu, hal ini tidak terlalu berdampak besar jika website yang anda bangun memerlukan banyak konfigurasi HTTP header.

## 17.8 Function **ob\_start()** dan **ob\_end\_flush()**

Di dalam web ‘nyata’, besar kemungkinan kita tidak bisa mengakses file **php.ini**. Biasanya pihak web hosting membatasi hak akses untuk alasan keamanan. Beberapa web hosting secara default juga tidak mengaktifkan fitur **Output Buffering**.

Tapi kita masih bisa menjalankan **output buffering** secara lokal di sebuah halaman. Caranya adalah dengan menggunakan fungsi **ob\_start()** dan **ob\_end\_flush()**. Fungsi **ob\_start()** digunakan untuk memulai proses buffer, sedangkan fungsi **ob\_end\_flush()** untuk mengakhirinya.

Fungsi **ob\_start()** harus ditempatkan pada posisi paling awal kode program, sedangkan fungsi **ob\_end\_flush()** bisa ditempatkan di baris mana saja ketika kode program yang mengakses HTTP Header sudah selesai.

Dengan pengaturan **output\_buffering=“off”** di dalam **php.ini**, saya bisa menjalankan kode berikut:

```
1 <?php
2   ob_start();
3   echo "Teks ini sudah dikirim ke web browser";
4   header("Location: http://www.google.com");
5 ?>
6 <!DOCTYPE html>
7 <head>
8   <meta charset="UTF-8">
9   <title>Belajar PHP</title>
10 <!-- lanjutan kode HTML disini -->
11
12 <?php
13   ob_end_flush();
14 ?>
```

Sekarang, saya bisa membuat perintah **echo** sebelum fungsi **header()**, walaupun setting **output\_buffering** di **php.ini** tidak aktif.

Tapi saya tidak bisa menjalankan kode program berikut:

```
1 <?php
2     echo "Teks ini sudah dikirim ke web browser";
3     ob_start();
4
5     header("Location: http://www.google.com");
6 ?>
7 <!DOCTYPE html>
8 <head>
9     <meta charset="UTF-8">
10    <title>Belajar PHP</title>
11    <!-- lanjutan kode HTML disini -->
12
13 <?php
14     ob_end_flush();
15 ?>
```

Hasilnya adalah “*error Warning: Cannot modify header information*”. Ini terjadi karena fungsi **ob\_start()** harus ditempatkan di baris paling awal kode PHP.

---

Pembahasan dalam bab kali ini mungkin sedikit rumit, tapi pemahaman tentang HTTP Header akan membantu anda untuk memahami cara kerja web server. Selain itu, materi ini juga berguna untuk menjelaskan sistem kerja form, cookie dan session PHP.

# 18. Date and Time Function

Selain tipe data dasar PHP, **date** atau **tanggal** merupakan “tipe data” lain yang tak kalah pentingnya. Hampir dalam setiap pemrosesan data, kita perlu mengolah tanggal. Apakah itu untuk tanggal lahir, tanggal pembuatan artikel, tanggal pembelian barang, tanggal bayar uang kuliah, tanggal jatuh tempo kredit, dll.

Di dalam PHP, tanggal bukanlah sebuah tipe data bawaan. Untuk memproses tanggal, kita menggunakan fungsi **date** dan **time**. Jika diterjemahkan, **date** berarti *tanggal*, sedangkan **time** berarti *waktu*. Walaupun punya arti yang berbeda, **date** dan **time** saling berkaitan.



Di dalam PHP Manual, **Date and Time Function** bisa diakses dari Function Reference  
-> Date and Time Related Extensions -> Date and Time -> Date/Time Functions.

Agar menghemat tempat, contoh kode program di bab ini tidak saya sertakan dengan kode HTMLnya, anda bisa menambahkan sendiri bagian tag HTML seperti `<!DOCTYPE html>`, `<head>`, `<body>`, dst.

## 18.1 Function time()

Format penulisan:

```
int time ( void )
```

Fungsi **time()** digunakan untuk mencari tanggal dan waktu saat ini. Mari langsung kita lihat contoh penggunaanya:

```
1 <?php
2   echo time(); // 1454079881
3 ?>
```

Fungsi **time()** mengembalikan nilai angka integer yang disebut dengan “*Unix timestamp*”. **Unix timestamp** adalah angka dalam satuan detik sejak tanggal 1 Januari 1970, pukul 00:00:00 GMT (*Greenwich Mean Time*). Tanggal ini dikenal juga sebagai **Unix Epoch**. Perhitungan tanggal dan waktu seperti ini umum digunakan dalam sistem **Unix** dan turunannya, seperti **Linux**.

Saat saya menjalankan kode program diatas, berarti sudah 1.454.079.881 detik berlalu sejak tanggal 1 Januari 1970. Setiap kali kode dijalankan, angka ini akan terus bertambah. Silahkan anda coba sendiri dengan me-refresh halaman beberapa kali.

Dengan sistem seperti ini, kita bisa mencari waktu untuk minggu depan atau hari kemaren. Caranya dengan menambah atau mengurangkan nilai *unix timestamp*, seperti contoh berikut:

```

1 <?php
2   $minggu_depan = time() + (7 * 24 * 60 * 60);
3   echo $minggu_depan;    // 1454686985
4   echo "<br>";
5
6   $kemaren = time() - (1 * 24 * 60 * 60);
7   echo $kemaren;         // 1453995785
8 ?>

```

Untuk mencari waktu di minggu depan, saya bisa menambahkan hasil fungsi `time()` dengan angka 604.800. Angka ini didapat dari 7 hari \* 24 jam \* 60 menit \* 60 detik. Begitu juga untuk mencari waktu hari kemaren, saya bisa mengurangi nilai `time()` dengan 24 jam, atau 86.400 detik.

## 18.2 Function date()

Format penulisan:

```
string date ( string $format [, int $timestamp = time()] )
```

Hasil dari fungsi `time()` yang kita bahas sebelumnya belum berarti apa-apa jika tidak berbentuk sebuah “tanggal”. Untuk menampilkan dan men-format tanggal, PHP menyediakan fungsi `date()`.

Fungsi `date()` membutuhkan 1 argumen utama dan 1 argumen opsional. Argumen utama diisi dengan string yang akan menentukan format tampilan tanggal. Sedangkan argumen kedua diisi dengan angka *unix timestamp*. Apabila argumen kedua tidak ditulis, akan menggunakan tanggal sekarang, yakni hasil dari fungsi `time()`.

Berikut contoh penggunaan fungsi `date()`:

```

1 <?php
2   echo date("d m Y", time());    // 30 01 2016
3   echo "<br>";
4   echo date("d m Y");           // 30 01 2016
5 ?>

```

Untuk sementara silahkan abaikan argumen pertama dari fungsi `date()` diatas, saya akan membahasnya sesaat lagi. Perhatikan bahwa kedua fungsi ini akan menampilkan hasil yang sama, yakni tanggal hari ini. Fungsi `time()` pada argumen kedua berarti kita meng-input *unix timestamp* saat ini.

Dengan mengubah nilai argumen kedua, saya bisa menampilkan tanggal minggu depan atau tanggal hari kemaren:

```

1 <?php
2   $minggu_depan = time() + (7 * 24 * 60 * 60);
3   echo date("d m Y", $minggu_depan);           // 06 02 2016
4   echo "<br>";
5
6   $kemaren = time() - (1 * 24 * 60 * 60);
7   echo date("d m Y", $kemaren );              // 29 01 2016
8 ?>

```

Dalam program diatas, saya memadukan pemanggilan fungsi **time()** dan **date()**.

## Format Tanggal dan Waktu Fungsi date()

String “d m Y” yang saya gunakan sebagai argumen pertama fungsi **date()** digunakan untuk mengatur format tampilan. “d” berarti saya ingin menampilkan hari dalam format 2 digit angka, “m” berarti saya ingin menampilkan bulan dalam format 2 digit angka, dan “Y” berarti saya ingin menampilkan tahun dalam format 4 digit angka.

Selain 3 karakter ini, masih banyak karakter lain yang bisa digunakan. Daftar lengkapnya bisa anda lihat di [PHP Manual](http://PHP Manual)<sup>1</sup>. Berikut beberapa diantaranya:

### Format angka/nama hari:

- d: Angka hari dengan format 2 digit dan didahului angka nol: 01 hingga 31.
- j: Angka hari dengan format 2 digit dan tanpa angka nol: 1 hingga 31.
- w: Angka hari untuk satu minggu, 0 hingga 6.
- z: Angka hari untuk satu tahun, 0 hingga 365.
- D: Nama hari dengan singkatan 3 karakter bahasa inggris: Sun, Mon hingga Sat.
- l: Nama hari dalam bahasa inggris: Sunday, Monday hingga Saturday.
- S: Awalan 2 digit karakter dalam bahasa inggris untuk nama hari: st, nd, rd dan th. Biasanya digunakan dengan format “j”.

### Format angka/nama bulan:

- m: Angka bulan dengan format 2 digit dan didahului angka nol: 01 hingga 12.
- n: Angka bulan dengan format 2 digit dan tanpa angka nol: 1 hingga 12.
- M: Nama bulan dengan singkatan 3 karakter bahasa inggris: Jan, Feb hingga Dec.
- F: Nama bulan dalam bahasa inggris: January, February hingga December.
- t: Total jumlah hari dalam 1 bulan : 28 hingga 31.

### Format angka/nama tahun:

- L: 1 jika tahun kabisat, 0 jika bukan tahun kabisat.
- Y: Angka tahun dengan 4 digit, seperti 1998, 2002 dan 2016
- y: Angka tahun dengan 2 digit, seperti 98, 02 dan 16

<sup>1</sup><http://php.net/manual/en/function.date.php>

## Format angka untuk waktu:

- a: am atau pm, dengan huruf kecil.
- A: AM atau PM, dengan huruf besar.
- g: Angka jam dalam format 12 jam, tanpa awalan nol: 1 hingga 12.
- G: Angka jam dalam format 24 jam, tanpa awalan nol: 0 hingga 23.
- h: Angka jam dalam format 12 jam, dengan awalan nol: 01 hingga 12.
- H: Angka jam dalam format 24 jam, dengan awalan nol: 01 hingga 23.
- i: Angka menit dengan awalan nol: 00 hingga 59.
- s: Angka detik dengan awalan nol: 00 hingga 59.

Berikut contoh penggunaan format ini sebagai argumen pertama dari fungsi **date()**:

```

1 <?php
2 echo date("jS F Y");           // 31st January 2016
3 echo "<br>";
4 echo date("1, jS F y ");       // Sunday, 31st January 16
5 echo "<br>";
6 echo date("H:i:s");           // 02:03:00
7 echo "<br>";
8 echo date("d - m - Y, H:i:s"); // 31 - 01 - 2016, 02:03:00
9 echo "<br>";
10 echo date("j F Y, h:i:s A");  // 31 January 2016, 02:03:00 AM
11 ?>

```

Walaupun namanya fungsi **date** (tanggal), tapi juga bisa digunakan untuk memformat waktu (jam, menit dan detik). Selain itu, saya bisa menambahkan karakter lain seperti ‘ – ’ dan ‘ : ’. Kedua karakter ini di gunakan sebagai pembatas format tanggal dan waktu agar lebih rapi.

Bagaimana jika saya ingin menambahkan karakter lain ke dalam argumen pertama, dan tidak ingin dianggap sebagai bagian dari format? Caranya adalah dengan men-escape karakter tersebut menggunakan backslash ‘ \ ’:

```

1 <?php
2 echo date("\T\ a\\n\g\g\ a\l: d - m - Y"); // Tanggal: 31 - 01 - 2016
3 ?>

```

Saya harus men-escape semua huruf ini karena memiliki makna dalam fungsi **date()**. Khusus untuk karakter n, harus di escape 2 kali karena “\n” juga memiliki makna di dalam string (lihat kembali pembahasan tentang tipe data string).

Untuk menambahkan string sebelum atau sesudah tanggal seperti fungsi diatas, akan lebih rapi jika disambung dari luar fungsi **date()**:

```

1 <?php
2 echo "Tanggal: ".date("d - m - Y"); // Tanggal: 31 - 01 - 2016
3 ?>

```

Nama hari dan bulan yang ditampilkan oleh fungsi **date()** menggunakan bahasa inggris. Jika ingin mengubahnya menjadi bahasa indonesia, kita harus membuatnya secara manual.

Caranya bisa dengan menggunakan kondisi if, case, atau bisa dengan array seperti contoh berikut:

```

1 <?php
2 $nama_hari=array("Minggu", "Senin", "Selasa", "Rabu",
3                   "Kamis", "Juma'at", "Sabtu" );
4 $nama_bulan=array("Januari", "Februari", "Maret", "April", "Mei",
5                   "Juni", "Juli", "Agustus", "September", "Oktober",
6                   "November", "Desember");
7
8 $hari = date("w");
9 $tanggal = date("j");
10 $bulan = date("n") - 1;
11 $tahun = date ("Y");
12
13 echo "Tanggal hari ini adalah: $nama_hari[$hari],
14      $tanggal $nama_bulan[$bulan] $tahun";
15
16 // Tanggal hari ini adalah: Minggu, 31 Januari 2016
17 ?>

```

Disini saya menggunakan fungsi **date()** sebagai key untuk array **\$nama\_hari** dan **\$nama\_bulan**.

Fungsi **date("w")** akan menghasilkan angka 0 - 6 sesuai nama hari, dimana 0 untuk hari minggu, 1 untuk hari senin, dst hingga 6 untuk hari sabtu. Nilai inilah yang bisa digunakan untuk key dari array **\$nama\_hari**.

Fungsi **date("n")** akan menghasilkan angka 1 - 12 sesuai dengan nama bulan. Karena array dimulai dari index 0, saya harus mengurangi 1 angka agar pas sebagai key array **\$nama\_bulan**.

Ketika semuanya digabung dengan angka tanggal dan tahun, kita sudah bisa menampilkan tanggal dalam Bahasa Indonesia.

## 18.3 Function **date\_default\_timezone\_set()**

Ketika anda menjalankan fungsi **date()** untuk menampilkan waktu dalam bentuk jam, menit dan detik, besar kemungkinan anda akan mendapatkan hasil yang berbeda dengan waktu yang terdapat di komputer.

Perbedaan waktu ini terjadi karena PHP memiliki settingan khusus terkait zona waktu (*timezone*). Untuk melihat pengaturan ini, silahkan jalankan `phpinfo()`, kemudian cari label “Default timezone”. Gunakan fitur pencarian (CRTL + F) agar mudah ditemukan.

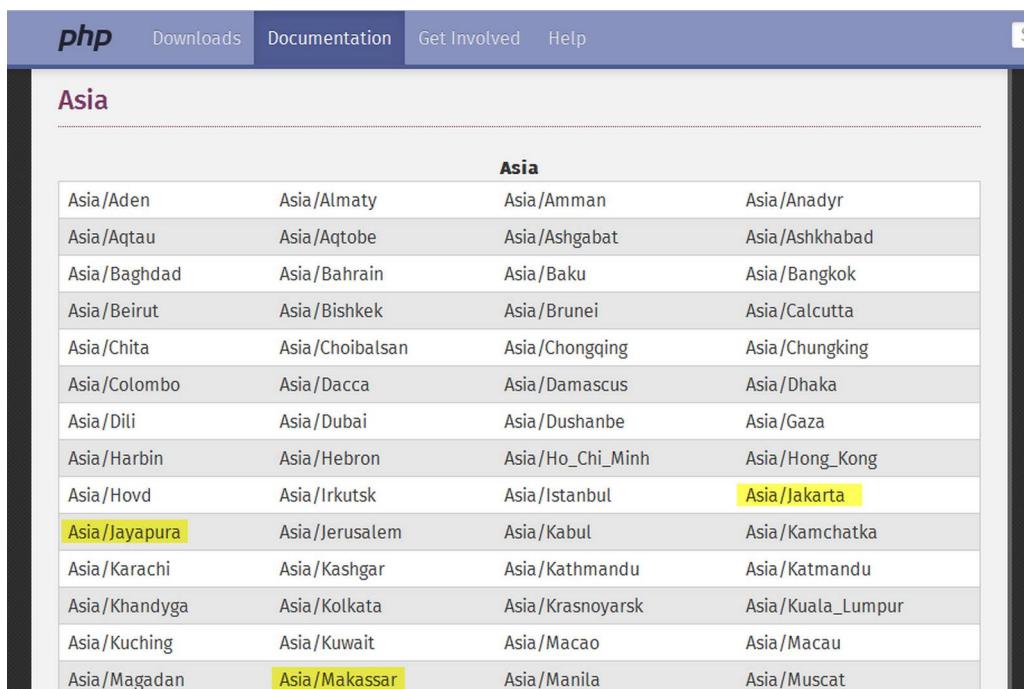
date	
date/time support	enabled
“Olson” Timezone Database Version	2015.7
Timezone Database	internal
Default timezone	Europe/Berlin

Gambar: Hasil ‘default timezone’ di `phpinfo()`

Seperti yang terlihat, ternyata di versi PHP yang saya gunakan *default timezone*-nya berisi nilai **Europe/Berlin**. Ini artinya seluruh fungsi `date()` akan disesuaikan menurut waktu di Berlin.

Untuk mengatur *default timezone* menjadi waktu di Indonesia, terdapat 2 cara: mengubah setting di `php.ini` atau menggunakan fungsi `date_default_timezone_set()`.

Sebelum mengubahnya, kita harus mencari dulu apa nilai yang harus diisi. PHP menyediakan berbagai zona waktu untuk hampir seluruh kota di dunia. Daftar lengkapnya bisa dilihat di [PHP Manual Timezone<sup>2</sup>](http://php.net/manual/en/timezones.php). Atau khusus untuk zona Asia bisa ke [PHP Manual Timezone Asia<sup>3</sup>](http://php.net/manual/en/timezones.asia.php).



Asia			
Asia/Aden	Asia/Almaty	Asia/Amman	Asia/Anadyr
Asia/Aqtau	Asia/Aqtobe	Asia/Ashgabat	Asia/Ashkhabad
Asia/Baghdad	Asia/Bahrain	Asia/Baku	Asia/Bangkok
Asia/Beirut	Asia/Bishkek	Asia/Brunei	Asia/Calcutta
Asia/Chita	Asia/Choibalsan	Asia/Chongqing	Asia/Chungking
Asia/Colombo	Asia/Dacca	Asia/Damascus	Asia/Dhaka
Asia/Dili	Asia/Dubai	Asia/Dushanbe	Asia/Gaza
Asia/Harbin	Asia/Hebron	Asia/Ho_Chi_Minh	Asia/Hong_Kong
Asia/Hovd	Asia/Irkutsk	Asia/Istanbul	Asia/Jakarta
Asia/Jayapura	Asia/Jerusalem	Asia/Kabul	Asia/Kamchatka
Asia/Karachi	Asia/Kashgar	Asia/Kathmandu	Asia/Katmandu
Asia/Khandyga	Asia/Kolkata	Asia/Krasnoyarsk	Asia/Kuala_Lumpur
Asia/Kuching	Asia/Kuwait	Asia/Macao	Asia/Macau
Asia/Magadan	Asia/Makassar	Asia/Mana	Asia/Muscat

Gambar: Timezone Asia

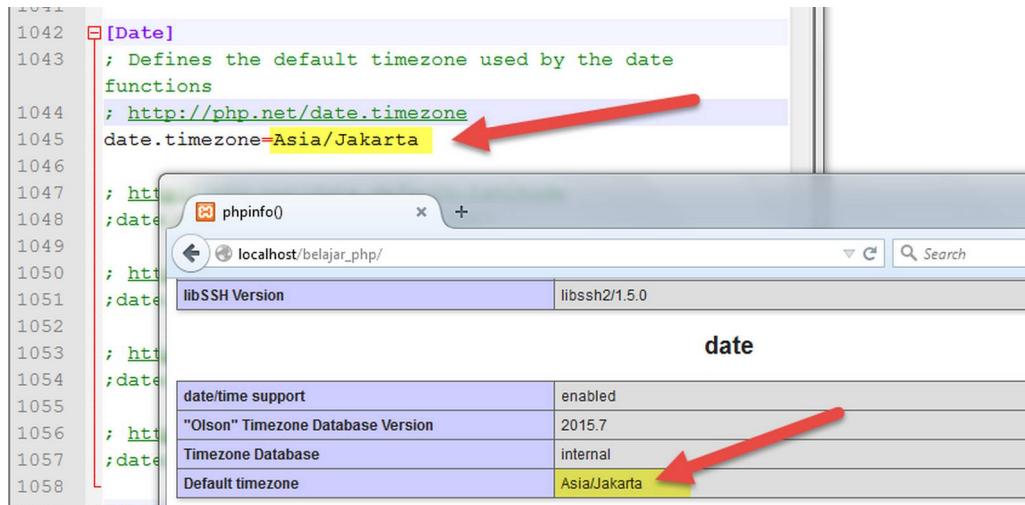
Dari daftar ini, kita bisa menggunakan “**Asia/Jakarta**” untuk waktu WIB, “**Asia/Makassar**” untuk waktu WITA, dan “**Asia/Jayapura**” untuk waktu WIT. Nilai-nilai inilah yang akan diinput kedalam `php.ini` atau fungsi `date_default_timezone_set()`.

Jika ingin melalui `php.ini`, silahkan cari settingan `date.timezone`, ubah nilainya menjadi “**Asia/Jakarta**” (tanpa tanda kutip). Save `php.ini`, lalu restart apache. Lihat kembali di `phpinfo()`

<sup>2</sup><http://php.net/manual/en/timezones.php>

<sup>3</sup><http://php.net/manual/en/timezones.asia.php>

untuk memastikan pengaturan telah diubah.



Gambar: Mengubah pengaturan 'default timezone' di `php.ini`

Jika semuanya sudah sesuai, silahkan jalankan kembali fungsi `date()`. Sekarang tanggal dan waktu yang tampil adalah waktu Jakarta (WIB). Untuk waktu WITA dan WIT, bisa disesuaikan dengan mengubah kembali settingan `date.timezone`.

Mengubah sebuah pengaturan dari `php.ini` akan berdampak ke seluruh server. Jika kita hanya ingin mengubah zona waktu untuk 1 halaman saja (atau karena satu dan lain hal tidak bisa mengakses file `php.ini`), bisa menggunakan fungsi `date_default_timezone_set()`. Zona waktu yang akan digunakan ditulis sebagai argumen fungsi ini. Berikut contoh penggunaannya:

```

1 <?php
2   date_default_timezone_set("Asia/Jakarta");
3   echo date("d - m - Y, H:i:s"); // 31 - 01 - 2016, 13:02:25
4 ?>

```

Fungsi `date_default_timezone_set()` akan mengatur zona waktu per halaman, namun akan tertimpa jika terdapat pemanggilan kedua fungsi ini:

```

1 <?php
2   date_default_timezone_set("Asia/Jakarta");
3   echo "Waktu WIB: ".date("d - m - Y, H:i:s");
4   // Waktu WIB: 31 - 01 - 2016, 13:04:48
5   echo "<br>";
6
7   date_default_timezone_set("Asia/Makassar");
8   echo "Waktu WITA: ".date("d - m - Y, H:i:s");
9   // Waktu WITA: 31 - 01 - 2016, 14:04:48
10  echo "<br>";
11
12  date_default_timezone_set("Asia/Jayapura");

```

```

13 echo "Waktu WIT: ".date("d - m - Y, H:i:s");
14 // Waktu WIT: 31 - 01 - 2016, 15:04:48
15 ?>

```

Dalam kode program diatas saya menampilkan 3 zona waktu di Indonesia (WIT, WITA, dan WIT). Jika saya menulis sebuah fungsi **date()** setelah baris terakhir, zona yang akan digunakan adalah “Asia/Jayapura”, karena inilah settingan terakhir dari fungsi **date\_default\_timezone\_set()**.

## 18.4 Function mktime()

Format penulisan:

```

int mktime ([ int $hour = date("H") [, int $minute = date("i")
[, int $second = date("s") [, int $month = date("n") [, int $day = date("j")
[, int $year = date("Y") [, int $is_dst = -1 ]]]]]] )

```

Tidak perlu pusing melihat format penulisan fungsi **mktime()** ini. Walaupun panjang, penggunaannya sangat sederhana. Fungsi **mktime()** mirip dengan fungsi **time()** yang sama-sama menghasilkan nilai berupa *unix timestamp*. Bedanya dalam fungsi **mktime()** kita bisa mengatur nilai dari *unix timestamp* yang dihasilkan.

Sebagai contoh, untuk mencari nilai *unix timestamp* tanggal 15 Februari 2016 pada pukul 10 lebih 30 menit 50 detik, saya bisa menggunakan kode berikut:

```

1 <?php
2 echo mktime(10, 30, 50, 2, 15, 2016); // 1455507050
3 ?>

```

Angka yang dihasilkan dari fungsi **mktime()** diatas berarti sudah 1.455.507.050 detik berlalu sejak tanggal 1 Januari 1970, pukul 00:00:00 GMT hingga tanggal 15 Februari 2016 pukul 10 lebih 30 menit 50 detik.

Jika menggunakan fungsi **time()**, kita tidak bisa mencari nilai *unix timestamp* seperti ini, karena fungsi **time()** hanya akan menghasilkan angka *unix timestamp* sekarang.

Perhatikan kembali bagaimana format penulisan argumen untuk fungsi **mktime()** ditulis. Susunannya adalah sebagai berikut:

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

Yang akan sering tertukar adalah penulisan bulan dan tanggal. Seperti yang terlihat, penulisannya bulan dahulu, baru diikuti oleh tanggal.

Nilai *unix timestamp* ini selanjutnya bisa diinput ke dalam fungsi **date()**:

```
1 <?php
2   $tanggal = mktime(10, 30, 50, 2, 15, 2016);
3   echo date("d M Y, H:i:s", $tanggal ); // 15 Feb 2016, 10:30:50
4 ?>
```

Atau bisa juga diinput langsung sebagai argumen kedua fungsi **date()**:

```
1 <?php
2   echo date("d M Y, H:i:s", mktime(10, 01, 01, 8, 17, 2016));
3   // 17 Aug 2016, 10:01:01
4   echo "<br>";
5
6   echo date("d M Y, H:i:s", mktime(16, 30, 00, 8, 8, 1945));
7   // 08 Aug 1945, 16:30:00
8   echo "<br>";
9
10  echo date("l, jS F y, H:i:s", mktime(03, 30, 00, 20, 11, 1990));
11  // Sunday, 11th August 91, 03:30:00
12 ?>
```

Penggunaan fungsi **mktime()** seperti ini baru berguna jika kita men-generate hasil **timestamp** secara otomatis, misalnya diambil dari database.

## 18.5 Function **getdate()**

Format penulisan:

```
array getdate ( [ int $timestamp = time() ] )
```

Fungsi **getdate()** akan menghasilkan komponen waktu dalam bentuk array. Fungsi ini bisa diisi dengan 1 argumen berupa nilai *unix timestamp*. Jika tidak ditulis, nilainya akan diambil dari fungsi **time()**, yakni waktu saat ini.

Agar seluruh element array hasil pemanggilan fungsi **getdate()** bisa terlihat, saya akan menggunakan fungsi **print\_r()**. Berikut contohnya:

```

1  <?php
2      date_default_timezone_set("Asia/Jakarta");
3      $saat_ini = getdate();
4
5      echo "<pre>";
6      print_r($saat_ini);
7      echo "</pre>";
8  ?>
9
10 /* Output-----
11 Array
12 (
13     [seconds] => 15
14     [minutes] => 23
15     [hours] => 13
16     [mday] => 31
17     [wday] => 0
18     [mon] => 1
19     [year] => 2016
20     [yday] => 30
21     [weekday] => Sunday
22     [month] => January
23     [0] => 1454221395
24 )
25 -----*/

```

Array yang dihasilkan adalah *associative array* dengan berbagai *key*. Kita bisa menampilkan nilainya menggunakan penulisan array, seperti contoh berikut:

```

1  <?php
2      date_default_timezone_set("Asia/Jakarta");
3      $merdeka = getdate(mktime(10, 00, 00, 8, 17, 1945));
4
5      // Tanggal: 17-8-1945
6      echo "Tanggal: {$merdeka["mday"]}-{$merdeka["mon"]}-{$merdeka["year"]}";
7  ?>

```

Komponen tanggal seperti ini sebenarnya juga bisa diambil menggunakan fungsi **date()**. Namun jika menggunakan fungsi **date()**, kita harus memanggilkan sebanyak 3 kali, yakni sekali untuk menampilkan tanggal, sekali untuk bulan, dan sekali untuk tahun. Dengan menggunakan fungsi **getdate()**, pemanggilan fungsi cukup 1 kali.

Fungsi **getdate()** cocok digunakan jika kita butuh memproses setiap element tanggal. Jika hanya ingin menampilkannya secara keseluruhan, fungsi **date()** akan lebih praktis.

## 18.6 Function strtotime()

Format penulisan:

```
int strtotime ( string $time [, int $now = time() ] )
```

Sesuai dengan namanya, fungsi **strtotime()** digunakan untuk mengkonversi string menjadi waktu. Atau lebih tepatnya mengubah string menjadi nilai *unix timestamp*. Berikut contohnya:

```
1 <?php
2     $tanggal = strtotime("10 September 2020");
3     echo $tanggal;    // 1599670800
4     echo "<br>";
5     echo date("d F Y", $tanggal); // 10 September 2020
6 ?>
```

Yang unik dari fungsi **strtotime()** adalah, string yang bisa diinput amat beragam. Perhatikan contoh-contoh berikut:

```
1 <?php
2     $tanggal = strtotime("last day of next month");
3     echo date("d F Y", $tanggal); // 29 February 2016
4     echo "<br>";
5
6     $tanggal = strtotime("first sunday of Mar 2016");
7     echo date("d F Y", $tanggal); // 06 March 2016
8     echo "<br>";
9
10    $tanggal = strtotime("1 week ago");
11    echo date("d F Y", $tanggal); // 24 January 2016
12    echo "<br>";
13
14    $tanggal = strtotime("3 days ago");
15    echo date("d F Y", $tanggal); // 28 January 2016
16    echo "<br>";
17
18    $tanggal = strtotime("+1 week 2 days 4 hours 2 seconds");
19    echo date("d F Y", $tanggal); // 09 February 2016
20    echo "<br>";
21
22    $tanggal = strtotime("tomorrow");
23    echo date("d F Y", $tanggal); // 01 February 2016
24 ?>
```

Seperti yang terlihat, saya bisa menampilkan tanggal besok dengan string “tomorrow”, atau mencari tanggal terakhir untuk bulan depan dengan string “last day of next month”. Ketika pertama kali mempelajari fungsi `strtotime()`, saya cukup terkesan karena PHP bisa memproses string seperti ini.

Daftar lengkap mengenai string apa saja yang didukung cukup panjang. Anda bisa membacanya di : [PHP Manual Datetime Format<sup>4</sup>](#). Secara umum, semuanya harus ditulis dengan bahasa inggris, seperti: “today”, “ago”, “tomorrow”, “yesterday”, “now”, dll.

Jika kita menginput string yang tidak dimengerti oleh fungsi `strtotime()` hasilnya adalah nilai boolean `false`, seperti contoh berikut:

```

1 <?php
2     $tanggal = strtotime("besok");
3     var_dump($tanggal); // bool(false)
4     echo date("d F Y", $tanggal); // 01 January 1970
5 ?>

```

Ketika nilai `false` ini diproses oleh fungsi `date()`, hasilnya adalah tanggal **Unix Epoch**, yakni 01 January 1970.

Yang perlu menjadi perhatian, fungsi `strtotime()` kadang bersikap ambigu. Perhatikan 2 contoh berikut:

```

1 <?php
2     echo date("d M Y", strtotime("11-10-2016")); // 11 Oct 2016
3     echo "<br>";
4     echo date("d M Y", strtotime("11/10/2016")); // 10 Nov 2016
5 ?>

```

Kedua string diatas hanya dibedakan oleh penulisan tanda pemisah ‘-’ dan ‘/’. Seperti yang terlihat, jika menggunakan tanda pemisah ‘/’, fungsi `strtotime()` akan memproses dengan format: **bulan/tanggal/tahun**. Sedangkan jika menggunakan tanda pemisah ‘-’ yang diproses adalah: **tanggal – bulan – tahun**.

Karena adanya perbedaan ini, harus selalu pastikan bahwa string yang diproses telah sesuai dengan yang diinginkan. Bagi kita di Indonesia, format yang biasa digunakan adalah: **tanggal – bulan – tahun**.

Sampai disini, kita telah mempelajari 2 fungsi yang bisa digunakan untuk mengatur nilai *unix timestamp*, yakni fungsi `mktime()` dan fungsi `strtotime()`. Keduanya bisa digunakan tergantung situasi dan akan menghasilkan nilai yang sama:

---

<sup>4</sup><http://php.net/manual/en/datetime.formats.php>

```

1 <?php
2 // 17 Aug 2016
3 echo date("d M Y", mktime(0, 0, 0, 8, 17, 2016));
4 echo "<br>";
5 echo date("d M Y", strtotime("17-8-2016"));
6 echo "<br>";
7
8 // 21 Apr 2016, 10:30:59
9 echo date("d M Y, H:i:s", mktime(10, 30, 59, 04, 21, 2016));
10 echo "<br>";
11 echo date("d M Y, H:i:s", strtotime("21-4-2016 10:30:59"));
12 echo "<br>";
13 ?>

```

Anda bebas ingin memproses tanggal dengan fungsi `mktime()` maupun `strtotime()`. Kebanyakan akan menggunakan fungsi `strtotime()` karena lebih fleksibel. Tapi hati-hati dengan hasil *ambigu* seperti perbedaan tanda ‘ / ’ dan ‘ – ’ yang sudah kita bahas sebelumnya.

## 18.7 Case Study: Menghitung Selisih Tanggal

Kali ini saya ingin “menantang” anda membuat sebuah kode program PHP untuk menghitung selesih antara dua buah tanggal. Misalkan tanggal pertama adalah: 2-05-2016, dan tanggal kedua: 7-05-2016. Berapa hari selesih antara kedua tanggal ini?

Saya yakin kasus seperti ini akan sering anda hadapi, misalnya ketika menampilkan tanggal jatuh tempo kredit, atau sisa hari sebelum pendaftaran mahasiswa baru.

Lebih detail lagi, saya ingin mendapatkan hasil seperti ini:

```

1 <?php
2 $tgl1 = strtotime("2-05-2016");
3 $tgl2 = strtotime("7-05-2016");
4
5 // kode program menghitung selisih tanggal
6 // kode program menghitung selisih tanggal
7 // kode program menghitung selisih tanggal
8
9 echo $selisih_hari; // 5
10 ?>

```

Variabel `$selisih_hari` akan menampung jumlah selisih hari dari variabel `$tgl1` dan `$tgl2`. Silahkan luangkan waktu sebentar untuk mencoba merancang kode programnya. Tentunya jika variabel `$tgl1` dan `$tgl2` diubah, nilai `$selisih_hari` juga akan berubah. Tidak boleh hanya menulis `$selisih_hari = 5`, hehe...

Selamat! jika anda berhasil membuatnya. Berikut kode program yang saya gunakan:

```

1  <?php
2  $tgl1 = strtotime("2-05-2016");
3  $tgl2 = strtotime("7-05-2016");
4
5  $selisih_tgl = abs($tgl2 - $tgl1);
6  echo $selisih_tgl; // 432000
7  echo "<br>";
8
9  $satu_hari = 24*60*60;
10 $selisih_hari = $selisih_tgl/$satu_hari;
11
12 echo $selisih_hari; //5
13 ?>

```

Sama seperti layaknya kode program lain, kode anda tidak harus sama persis. Yang penting hasilnya benar dan sesuai.

Mari kita bahas sejenak kode program diatas. Jika anda telah memahami pembahasan tentang fungsi `strtotime()`, tentu masih ingat bahwa fungsi ini akan menghasilkan nilai berupa *unix timestamp*.

Nilai *unix timestamp* adalah angka dalam satuan detik. Dengan demikian, untuk mencari selisih hari, saya tinggal mengurangkan nilai `$tgl2` dengan `$tgl1`. Saya menambahkan fungsi `abs()` sebagai antisipasi jika terdapat kemungkinan nilai dari `$tgl2` lebih kecil dari `$tgl1`. Yang penting disini adalah selisih, sehingga kita tidak mempermasalahkan tanggal mana yang lebih besar.

Variabel `$selisih_tgl` akan berisi selisih hari dari `$tgl2` dengan `$tgl1` (atau `$tgl1` dengan `$tgl2`). Tapi satunya masih dalam detik. Untuk mengkonversi detik menjadi hari, saya harus membaginya dengan jumlah detik dalam 1 hari (24 jam \* 60 menit \* 60 detik). Hasilnya, adalah selisih dalam satuan hari.

Agar lebih menantang, saya ingin anda memodifikasi kode diatas, tapi kali ini saya ingin selisih ini bukan lagi dalam satuan hari, tapi dalam 3 satuan: **hari**, **bulan**, dan **tahun**. Berikut hasil yang saya inginkan:

```

1  <?php
2  $tgl1 = strtotime("12-10-2016");
3  $tgl2 = strtotime("9-6-2014");
4
5  // kode program menghitung selisih tanggal
6  // kode program menghitung selisih tanggal
7  // kode program menghitung selisih tanggal
8
9  echo "Selisih tanggal adalah $selisih_tahun tahun,
10      $selisih_bulan bulan, $selisih_hari hari";
11 // Selisih tanggal adalah 2 tahun, 4 bulan, 6 hari
12 ?>

```

Seperti yang terlihat, hasil akhirnya adalah 3 buah variabel: `$selisih_tahun`, `$selisih_bulan`, dan `$selisih_hari`. Setiap variabel ini menampung nilai sesuai namanya.

Perbedaan mendasar dengan kode program kita sebelumnya adalah dalam membagi detik ke dalam tahun, bulan, dan hari. Sebagai contoh, jika diberikan angka 73.958.400 detik, dapatkah anda menghitung berapa nilainya jika diubah menjadi tahun, bulan, dan hari?

Asumsi yang bisa kita pakai adalah, 1 tahun terdiri dari 365 hari. 1 bulan terdiri dari 30 hari, dan 1 hari terdiri dari 24 jam. Silahkan anda buat kode programnya. Kalau perlu gunakan kertas dan pulpen untuk corat-coret.

Jika anda adalah mahasiswa jurusan komputer/IT tentu tidak asing dengan tantangan seperti ini. Namun jika anda tidak memiliki background programming, saya juga sarankan untuk mencoba membuat kode programnya. Untuk menjadi seorang programmer, pemecahan masalah seperti inilah yang harus selalu dilatih.

Sekali lagi selamat jika anda bisa membuatnya. Inilah kode program yang saya gunakan:

```
1 <?php
2   $tgl1 = strtotime("12-10-2016");
3   $tgl2 = strtotime("9-6-2014");
4
5   $selisih_tgl = abs($tgl2 - $tgl1);
6   echo $selisih_tgl; // 73958400
7   echo "<br>";
8
9   $satu_tahun = 365*24*60*60;
10  $satu_bulan = 30*24*60*60;
11  $satu_hari = 24*60*60;
12
13  $selisih_tahun = floor($selisih_tgl/$satu_tahun);
14  $selisih_bulan = floor(($selisih_tgl - ($selisih_tahun * $satu_tahun))/
15                         $satu_bulan);
16  $selisih_hari = floor(($selisih_tgl - ($selisih_tahun * $satu_tahun) -
17                         ($selisih_bulan * $satu_bulan))/
18                         $satu_hari);
19
20  echo "Selisih tanggal adalah $selisih_tahun tahun,
21      $selisih_bulan bulan, $selisih_hari hari";
22  // Selisih tanggal adalah 2 tahun, 4 bulan, 6 hari
23 ?>
```

Silahkan anda pelajari sejenak apa yang dilakukan kode program diatas.

Sudah? Baik, mari kita bahas sedikit. Konsepnya adalah saya akan membagi nilai `$selisih_tgl` dengan total detik dalam 1 tahun, lalu sisanya dibagi lagi dengan total detik dalam 1 bulan, dan jika masih bersisa, akan dibagi dengan total detik dalam 1 hari.

Ketika nilai **73958400** dibagi dengan total detik dalam 1 tahun ( $365*24*60*60$ ), hasilnya adalah 2,34. Angka yang kita perlukan hanyalah bilangan bulatnya saja. Karena itulah saya menggunakan fungsi **floor()**. Fungsi floor akan membulatkan suatu angka ke bawah.

Selanjutnya, untuk menghitung sisanya dalam satuan bulan, didapat dari  $73958400 - 2$  tahun / jumlah detik dalam 1 bulan. Demikian pula dengan menghitung sisa hari.

Nilai setiap satuan ini kemudian disimpan ke dalam variabel **\$selisih\_tahun**, **\$selisih\_bulan**, dan **\$selisih\_hari**.



Di dalam ilmu komputer, apa yang kita lakukan disini adalah membuat sebuah *algoritma* perhitungan selisih tanggal. **Algoritma** adalah sebuah ilmu yang mencari cara memecahkan suatu masalah dengan langkah-langkah logis. Setiap programmer harus bisa menerjemahkan suatu masalah menjadi kode program yang terdiri dari langkah-langkah sederhana.

Studi kasus kita belum selesai. Kali ini saya ingin agar kode program yang sudah kita buat dengan susah payah ini dijadikan sebuah **function**. Dengan demikian, jika kita ingin menghitung selisih tanggal, bisa dengan cara memanggil fungsi tanpa harus membuat ulang seluruh kode programnya.

Saya akan menamakan fungsi ini dengan nama **cari\_selisih\_tanggal()**. Input fungsi membutuhkan 2 argumen, yakni dua buah string yang terdiri dari tanggal yang ingin dicari selisihnya. Hasil pemanggilan fungsi **cari\_selisih\_tanggal()** adalah 3 variabel yang mewakili variabel **\$selisih\_tahun**, **\$selisih\_bulan**, dan **\$selisih\_hari**.

Syarat untuk bisa membuat kode program ini adalah anda sudah memahami cara membuat **function()**. Tapi, mungkin anda akan bertanya, bagaimana cara sebuah function mengembalikan 3 buah nilai? Jawabannya adalah dengan melalui array. Dengan kata lain fungsi ini akan mengembalikan nilai (return) sebuah array yang memiliki 3 element.

Baik, sudah selesai? Atau sudah menyerah? :)

Berikut kode program yang saya rancang untuk fungsi **cari\_selisih\_tanggal()**:

```

1 <?php
2 function cari_selisih_tanggal($tanggal1,$tanggal2)
3 {
4     $tgl1 = strtotime($tanggal1);
5     $tgl2 = strtotime($tanggal2);
6
7     $selisih_tgl = abs($tgl2 - $tgl1);
8     $satu_hari = 24*60*60;
9     $satu_bulan = 30*24*60*60;
10    $satu_tahun = 365*24*60*60;
11
12    $selisih["tahun"] = floor($selisih_tgl/$satu_tahun);
13    $selisih["bulan"] = floor(
14                                ($selisih_tgl - ($selisih["tahun"] * $satu_tahun))/
```

```

15           $satu_bulan);
16   $selisih["hari"] = floor(
17           ($selisih_tgl - ($selisih["tahun"] * $satu_tahun) -
18           ($selisih["bulan"] * $satu_bulan))/
19           $satu_hari);
20   return $selisih;
21 }
22 ?>

```

Perhatikan bagaimana saya membuat 2 buah argumen `$tanggal1` dan `$tanggal2`. Kedua argumen ini kemudian diinput ke dalam fungsi `strtotime()` untuk selanjutnya di proses di dalam fungsi.

Agar dapat mengembalikan 3 nilai, saya harus menyimpan variabel `$selisih_tahun`, `$selisih_bulan`, dan `$selisih_hari` sebagai sebuah array. Agar mudah dibedakan, saya membuatnya menjadi *associative array*.

Bagaimana cara menggunakan fungsi ini? Berikut contoh pemanggilannya:

```

1 <?php
2
3 $hasil = cari_selisih_tanggal("12-10-2016", "27-10-2016");
4
5 echo "Selisih tanggal adalah =
6     {$hasil["tahun"]} tahun,
7     {$hasil["bulan"]} bulan,
8     {$hasil["hari"]} hari";
9 // Selisih tanggal adalah = 0 tahun, 0 bulan, 15 hari
10 echo "<br>";
11
12 $hasil = cari_selisih_tanggal("12-10-1995", "12-10-2016");
13
14 echo "Selisih tanggal adalah =
15     {$hasil["tahun"]} tahun,
16     {$hasil["bulan"]} bulan,
17     {$hasil["hari"]} hari";
18 // Selisih tanggal adalah = 21 tahun, 0 bulan, 5 hari
19
20 echo "<br>";
21
22 $hasil = cari_selisih_tanggal("12-10-1995", "now");
23
24 echo "Selisih tanggal adalah =
25     {$hasil["tahun"]} tahun,
26     {$hasil["bulan"]} bulan,
27     {$hasil["hari"]} hari";
28 // Selisih tanggal adalah = 20 tahun, 3 bulan, 24 hari
29 ?>

```

Dengan mengubah sebuah kode program menjadi fungsi, hasilnya akan jauh lebih efisien. Terutama jika fungsi tersebut digunakan beberapa kali sepanjang kode program.

Satu hal yang patut diperhatikan. Ketika saya menjalankan fungsi `cari_selisih_tanggal("12-10-1995", "12-10-2016")`, hasilnya adalah: 21 tahun, 0 bulan, 5 hari. Bisakah anda menjelaskan kenapa ada lebih 5 hari? Padahal tanggal dan bulannya sama!

Ini terjadi karena di dalam perhitungan hasil fungsi `cari_selisih_tanggal()` saya menetapkan jumlah hari dalam 1 tahun adalah 365 hari. Bagaimana dengan tahun kabisat? Jumlah harinya akan menjadi 366. Inilah asal dari selisih 5 hari tersebut. Diantara tahun 1995 hingga tahun 2016, terdapat 5 kali tahun kabisat (bahasa inggris tahun kabisat: *leap year*).

Untuk mengatasi masalah ini kita terpaksa harus merubah kembali kode program. Ini lumayan rumit untuk programmer pemula, karena membuat kode untuk memeriksa apakah saat ini tahun kabisat atau tidak. Untungnya, di dalam versi 5.3 keatas, PHP menyertakan **DateTime object**.

## 18.8 DateTime Object PHP

Selain menggunakan fungsi-fungsi untuk pemrosesan tanggal dan waktu, PHP menyediakan fitur lanjutan yang dikemas ke dalam **DateTime Object**. Pembahasan tentang object memang tidak saya bahas dalam buku ini karena termasuk materi PHP lanjutan. Namun untuk **DateTime object**, juga tersedia dengan “rasa” procedural (dipanggil dengan fungsi).

Untuk membuat sebuah *datetime object*, bisa menggunakan fungsi `date_create()`. Berikut contohnya:

```
1 <?php
2   $tanggal = date_create("17-08-2016");
3   echo $tanggal;
4   // Catchable fatal error: Object of class DateTime could
5   // not be converted to string
6 ?>
```

Error diatas terjadi karena saya mencoba menampilkan isi dari variabel `$tanggal` menggunakan perintah `echo`. Di dalam PHP, sebuah *object* tidak bisa langsung ditampilkan seperti ini.

Argumen untuk fungsi `date_create()` sama seperti fungsi `strtotime()`, jadi kita bisa mengisinya dengan string lain seperti: “tomorrow” atau “next week”.

Untuk menampilkan nilai *datetime object*, bisa menggunakan fungsi `date_format()`. Berikut contohnya:

```

1 <?php
2   $tanggal = date_create("17-08-2016");
3   echo date_format($tanggal, "d M Y");           // 17 Aug 2016
4   echo "<br>";
5
6   $tanggal = date_create("21-4-2016 10:30:59");
7   echo date_format($tanggal, "d M Y, H:i:s");    // 21 Apr 2016, 10:30:59
8   echo "<br>";
9
10  $tanggal = date_create("next week");
11  echo date_format($tanggal, "d M Y");           // 08 Feb 2016
12 ?>

```

Fungsi **date\_format()** membutuhkan 2 buah argumen. Argumen pertama adalah variabel yang berisi *datetime object* hasil dari fungsi **date\_create()**. Sedangkan argumen kedua adalah format tampilan yang diinginkan. Cara penulisan format tampilan ini sama seperti fungsi **date()**.

Baik, karena kita sudah membahas cara membuat dan menampilkan *datetime object*, saatnya kita pelajari fungsi **date\_diff()** untuk mencari selisih tanggal.

Sama seperti fungsi **cari\_selisih\_tanggal()** yang sudah kita rancang, fungsi **date\_diff()** juga membutuhkan 2 buah argumen yang harus diisi dengan tanggal. Bedanya, tanggal ini harus berupa *datetime object*. Selain itu, fungsi **date\_diff()** akan mengembalikan hasil dalam bentuk *object*, bukan array. Berikut contoh penggunaannya:

```

1 <?php
2   $tgl1 = date_create("17-8-2016");
3   $tgl2 = date_create("19-08-2016");
4
5   $selisih = date_diff($tgl2, $tgl1);
6
7   echo "<pre>";
8   print_r($selisih);
9   echo "</pre>";
10 ?>
11
12 /* Output-----
13 DateInterval Object
14 (
15   [y] => 0
16   [m] => 0
17   [d] => 2
18   [h] => 0
19   [i] => 0
20   [s] => 0
21   [weekday] => 0
22   [weekday_behavior] => 0

```

```

23     [first_last_day_of] => 0
24     [invert] => 1
25     [days] => 2
26     [special_type] => 0
27     [special_amount] => 0
28     [have_weekday_relative] => 0
29     [have_special_relative] => 0
30 )
31 -----*/

```

Untuk sementara, anda bisa menganggap bahwa object ini tidak lain sebuah *associative array*. Sebagaimana layaknya array, saya menggunakan fungsi `print_r()` untuk menampilkan isi dari object ini.

Hasil dari fungsi `date_diff()` sangat lengkap dan detail. Untuk menampilkan selisih tanggal kita bisa mengakses nilai “`d`”, “`m`” dan “`y`” yang akan berisi selisih dalam satuan hari, bulan dan tahun. Untuk mencari perbedaan jam, menit dan detik bisa mengakses nilai “`h`”, “`i`” dan “`s`”.

Berikut cara mengakses nilai-nilai ini:

```

1 <?php
2   $tgl1 = date_create("17-8-2016");
3   $tgl2 = date_create("19-9-2019");
4
5   $selisih = date_diff($tgl2, $tgl1);
6
7   echo $selisih->d;    // 2
8   echo "<br>";
9   echo $selisih->m;    // 1
10  echo "<br>";
11  echo $selisih->y;    // 3
12 ?>

```

Karena hasil dari fungsi `date_diff()` adalah sebuah *objek*, saya menggunakan operator “`->`” untuk mengakses setiap komponen nilainya. Nilai-nilai di dalam object ini mirip dengan sebuah variabel. Dalam konsep OOP (*Object Oriented Programming*), nilai ini dikenal dengan istilah **property**.

Kita juga bisa mengakses property ini langsung dari dalam string:

```

1 <?php
2   $tgl1 = date_create("17-8-2016");
3   $tgl2 = date_create("19-9-2019");
4
5   $selisih = date_diff($tgl2, $tgl1);
6
7   echo "Selisih tanggal adalah = $selisih->y tahun,
8     $selisih->m bulan, $selisih->d hari";
9   // Selisih tanggal adalah = 3 tahun, 1 bulan, 2 hari
10 ?>

```

Terlihat, fungsi `date_diff()` sudah melakukan perhitungan selisih tanggal untuk kita.

## 18.9 Case Study: Refactoring Fungsi `cari_selisih_tanggal()`

Sebelum saya menutup bab tentang fungsi `date` dan `time` ini, saya ingin mengajak anda untuk “memperbaiki” fungsi `cari_selisih_tanggal()` yang telah kita buat sebelumnya.

Kali ini saya ingin mengganti isi dari fungsi `cari_selisih_tanggal()` dengan fungsi `date_diff()`. Syaratnya, nilai *input* dan *output* fungsi tetap sama, yakni tetap diinput dengan 2 argumen, dan mengembalikan sebuah *associative array* dengan 3 nilai.

Di dalam teori programming, memperbaiki sebuah fungsi atau bagian kode program tanpa mengubah struktur luarnya (input dan output) dikenal dengan istilah **Code Refactoring**. Dalam contoh ini, saya ingin memperbaiki fungsi `cari_selisih_tanggal()`, tanpa mengubah input (*argumen*) dan outputnya (*return value*).

Berikut *code refactoring* dari fungsi `cari_selisih_tanggal()`:

```

1 <?php
2 function cari_selisih_tanggal($tanggal1,$tanggal2)
3 {
4   $tgl1 = date_create($tanggal1);
5   $tgl2 = date_create($tanggal2);
6
7   $selisih_tgl = date_diff($tgl2, $tgl1);
8
9   $selisih[ "tahun" ] = $selisih_tgl->y;
10  $selisih[ "bulan" ] = $selisih_tgl->m;
11  $selisih[ "hari" ] = $selisih_tgl->d;
12
13  return $selisih;
14 }
15 ?>

```

Dapat anda lihat bahwa sekarang fungsi `cari_selisih_tanggal()` menggunakan fungsi `date_diff()` di dalam internalnya. Mari kita uji seperti *case study* pertama:

```
1 <?php
2 $hasil = cari_selisih_tanggal("12-10-2016", "27-10-2016");
3
4 echo "Selisih tanggal adalah =
5     {$hasil["tahun"]} tahun,
6     {$hasil["bulan"]} bulan,
7     {$hasil["hari"]} hari";
8 // Selisih tanggal adalah = 0 tahun, 0 bulan, 15 hari
9
10 echo "<br>";
11
12 $hasil = cari_selisih_tanggal("12-10-1995", "12-10-2016");
13
14 echo "Selisih tanggal adalah =
15     {$hasil["tahun"]} tahun,
16     {$hasil["bulan"]} bulan,
17     {$hasil["hari"]} hari";
18 // Selisih tanggal adalah = 21 tahun, 0 bulan, 0 hari
19
20 echo "<br>";
21
22 $hasil = cari_selisih_tanggal("12-10-1995", "now");
23
24 echo "Selisih tanggal adalah =
25     {$hasil["tahun"]} tahun,
26     {$hasil["bulan"]} bulan,
27     {$hasil["hari"]} hari";
28 // Selisih tanggal adalah = 20 tahun, 3 bulan, 17 hari
29 ?>
```

Kode ini sama persis seperti yang digunakan pada pembahasan fungsi *cari\_selisih\_tanggal()* sebelumnya. Perhatikan error yang dihasilkan karena masalah tahun kabisat sekarang sudah terselesaikan. Selain itu kode programnya jauh lebih singkat dan tidak membuat pusing. PHP memprosesnya secara internal dari fungsi **date\_diff()**.

Sampai disini mungkin anda bertanya, kenapa saya tidak membahas fungsi **date\_diff()** sejak awal? Kenapa harus repot-repot menghitung perbedaan tanggal secara manual?

Menggunakan fungsi **date\_diff()** memang lebih praktis, tapi belajar membuat algoritma untuk mencari selisih tanggal juga tidak kalah penting. Seandainya nanti anda lanjut belajar bahasa pemrograman lain seperti **JavaScript**, fungsi **date\_diff()** tidak akan tersedia. Oleh karena itulah dasar-dasar seperti ini saya rasa cukup penting untuk dipelajari.

Terlepas dari itu semua, mencari fungsi yang lebih praktis juga sebuah “ilmu” yang harus dikuasai. Di dalam programming, terdapat istilah “*lazy programming*”. Artinya bukan programmer yang malas, tetapi programmer kreatif yang bisa mencari jalan pintas. Daripada membuat manual cara menghitung selisih tanggal, kita bisa mencari fungsi yang siap pakai seperti

`date_diff()` ini. Namun jika fungsi tersebut tidak tersedia, anda juga harus bisa membuatnya sendiri :)

---

Dalam bab ini kita telah membahas tentang fungsi-fungsi date and time untuk pemrosesan tanggal. Selain itu studi kasus untuk membuat fungsi pencari selisih tanggal. Berikutnya saya akan masuk ke **Globals Variabel PHP**.

# 19. Magic Constants dan Superglobals Variable

Di dalam PHP, terdapat beberapa konstanta dan variabel khusus yang memiliki status sebagai ‘Magic’ dan ‘SuperGlobal’. Di sebut seperti itu, karena konstanta dan variabel ini di-generate otomatis oleh PHP. Keduanya juga bisa diakses dari mana saja, baik di *global scope* maupun *local scope* seperti di dalam *function*.

## 19.1 Magic Constant PHP

Terdapat 8 konstanta yang menyadang nama sebagai **Magic Constants** di dalam PHP. Konstanta ini bisa diakses darimana saja. Berikut ke-8 konstanta tersebut:

- `__LINE__`
- `__FILE__`
- `__DIR__`
- `__FUNCTION__`
- `__CLASS__`
- `__TRAIT__`
- `__METHOD__`
- `__NAMESPACE__`

Satu hal yang bisa langsung diperhatikan, seluruh konstanta ini diawali dan diakhiri dengan *dua buah tanda spasi “\_ \_”*. Ini di desain agar menghindari bentrok dengan konstanta yang umum digunakan (yang dibuat oleh programmer sendiri). Oleh karena alasan ini pula sebaiknya kita tidak membuat konstanta yang juga diawali dua tanda spasi.

Secara umum, seluruh *konstanta ajaib* ini (**magic constants**), berfungsi memberikan informasi terkait posisi ketika perintah tersebut dipanggil.

Konstanta `__LINE__` berisi nomor baris. Konstanta `__FILE__` dan `__DIR__` berisi nama file dan folder (*directory*) dari kode program saat ini. Begitu juga dengan `__FUNCTION__` yang akan berisi nama function pada saat konstanta ini dipanggil.

Konstanta `__CLASS__`, `__TRAIT__`, `__METHOD__`, dan `__NAMESPACE__` berkaitan dengan **OOP** (*Object Oriented Programming*) oleh karena itu tidak saya bahas.

Berikut contoh penggunaan **magic constants** PHP :

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6  </head>
7  <body>
8  <h1>PHP Magic Constants</h1>
9  <?php
10     function ada_ada_saja() {
11         return "Teks ini berasal dari fungsi: ".__FUNCTION__;
12     }
13
14     echo "Teks ini berada di baris : ".__LINE__."
15     echo "Alamat lengkap file ini adalah : ".__FILE__."
16     echo "File ini berada di folder: ".__DIR__."
17     echo ada_ada_saja().""
18 ?>
19 </body>
20 </html>
21
22 /* Output-----
23 Teks ini berada di baris : 14
24 Alamat lengkap file ini adalah : C:\xampp\htdocs\belajar_php\index.php
25 File ini berada di folder: C:\xampp\htdocs\belajar_php
26 Teks ini berasal dari fungsi: ada_ada_saja
27 -----*/

```

Dari hasil yang diperoleh, terlihat bahwa setiap konstanta akan memberikan hasil tergantung posisinya. **Magic constants** ini biasanya digunakan untuk proses debugging atau pencarian kesalahan.

## 19.2 Superglobals Variable

**Superglobals Variable** adalah variabel khusus yang juga bisa diakses kapan saja dan dimana saja sepanjang kode program. Isi dari variabel ini di generate secara otomatis oleh PHP. Terdapat 9 *superglobals variable*:

- **\$GLOBALS**
- **\$\_SERVER**
- **\$\_GET**
- **\$\_POST**
- **\$\_FILES**
- **\$\_COOKIE**

- `$_SESSION`
- `$_REQUEST`
- `$_ENV`

Juga bisa diperhatikan cara penulisan variabel ini. Hampir semua *superglobals variable* PHP diawali dengan karakter `"$_"`. Oleh karena itu sebaiknya kita juga tidak menulis variabel dengan karakter awal seperti ini.

Seluruh *superglobal variabel* (kecuali `$GLOBALS`) digunakan sebagai sarana input ke dalam PHP. Sebagai contoh, untuk memproses form kita akan menggunakan variabel `$_GET` dan `$_POST`.

Dalam bab ini saya hanya akan membahas 4 variabel superglobal: `$GLOBALS`, `$_SERVER`, `$_GET` dan `$_ENV`. Untuk variabel `$_POST`, `$_FILES`, `$_COOKIE`, `$_SESSION` dan `$_REQUEST` akan dibahas dalam bab-bab berikutnya.

## 19.3 Superglobals Variable `$GLOBALS`

Jika variabel *superglobal* lain digunakan untuk mengambil data dari luar ke dalam PHP, variabel `$GLOBALS` berfungsi untuk mengakses variabel global dari dalam *local scope*. Contoh *local scope* adalah dari dalam sebuah *function*.

Fungsi variabel `$GLOBALS` ini hampir sama dengan fungsi *keyword global* yang kita pelajari ketika membahas *user defined function*. Sebagai contoh, perhatikan kode program berikut:

```
1 <?php
2     $hari="Senin";
3
4     function nama_hari(){
5         return "Sekarang hari $hari";
6     }
7
8     echo nama_hari() // Notice: Undefined variable
9 ?>
```

Kode program diatas akan menghasilkan **error** karena variabel `$hari` tidak dikenal di dalam *function* (*local scope*). Variabel `$hari` saya definisikan di luar *function* (*global scope*). Agar dapat diakses, bisa dengan menambahkan *keyword global*, seperti contoh berikut:

```

1  <?php
2      $hari="Senin";
3
4      function nama_hari(){
5          global $hari;
6          return "Sekarang hari $hari";
7      }
8
9      echo nama_hari() // Sekarang hari Senin
10 ?>

```

Kali ini variabel `$hari` sudah bisa diakses dari dalam fungsi.

Menggunakan *variable superglobal* `$GLOBALS`, saya bisa melakukan hal yang sama:

```

1  <?php
2      $hari="Senin";
3
4      function nama_hari(){
5          return "Sekarang hari {$GLOBALS["hari"]}";
6      }
7
8      echo nama_hari() // Sekarang hari Senin
9 ?>

```

Perhatikan bagaimana cara penggunaannya. Variabel `$GLOBALS` merupakan *associative array* yang berisi seluruh variabel global. Untuk mengakses variabel `$hari`, saya bisa menggunakan `$GLOBALS["hari"]`.

Saya juga bisa menggunakan variabel `$GLOBALS` untuk menukar nilai variabel `$hari` dari dalam function. Seperti contoh berikut:

```

1  <?php
2      $hari="Senin";
3
4      function nama_hari(){
5          $GLOBALS["hari"] = "Jum'at";
6      }
7
8      // jalankan fungsi nama_hari()
9      nama_hari();
10     echo "Sekarang hari $hari"; // Sekarang hari Jum'at
11 ?>

```

Hasilnya, variabel `$hari` juga akan berubah ketika diakses dari luar function.

## 19.4 Superglobals Variable `$_ENV`

Variabel `$_ENV` berisi informasi tentang *environment* atau lingkungan pemrograman. Di sistem operasi Windows, variabel ini tidak berisi nilai apa-apa. Jika diakses di sistem Unix seperti Linux, `$_ENV` bisa berisi informasi seperti nama user yang menjalankan kode program. Karena saya tidak bisa mencontohkannya dan penggunaannya juga tidak banyak, mari kita lanjut ke variabel *superglobal* berikutnya.

## 19.5 Superglobals Variable `$_SERVER`

Variabel *superglobals* `$_SERVER` berisi informasi server dan informasi web browser yang digunakan pengunjung. Kebanyakan informasi ini berasal dari **HTTP Header** yang dikirim dari dan ke web server. Sama seperti *variabel superglobals* lainnya, informasi ini disusun ke dalam sebuah *associative array*.

Untuk melihat seluruh informasi yang ada di dalam variable `$_SERVER`, kita bisa menggunakan fungsi `print_r()`. Berikut contoh kode programnya:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6  </head>
7  <body>
8  <h1>PHP Super Global Variable </h1>
9  <pre>
10 <?php
11     print_r($_SERVER);
12 ?>
13 </pre>
14 </body>
15 </html>
```

Saya sengaja meletakkan fungsi `print_r()` diantara tag `<pre>` agar tampilannya lebih rapi. Berikut hasil yang dapat:



Gambar: Tampilan dari variabel \$\_SERVER

Berikut rincian dari tampilan diatas:

```

Array
(
    [MIBDIRS] => C:/xampp/php/extras/mibs
    [MYSQL_HOME] => \xampp\mysql\bin
    [OPENSSL_CONF] => C:/xampp/apache/bin/openssl.cnf
    [PHP_PEAR_SYSCONF_DIR] => \xampp\php
    [PHPRC] => \xampp\php
    [TMP] => \xampp\tmp
    [HTTP_HOST] => localhost
    [HTTP_USER_AGENT] => Mozilla/5.0 (Windows NT 6.1; WOW64; rv:40.0)
        Gecko/20100101 Firefox/40.0
    [HTTP_ACCEPT] => text/html,application/xhtml+xml,
                    application/xml;q=0.9,*/*;q=0.8
    [HTTP_ACCEPT_LANGUAGE] => en-US,en;q=0.5
    [HTTP_ACCEPT_ENCODING] => gzip, deflate
    [HTTP_DNT] => 1
    [HTTP_COOKIE] => __utma=111872281.677479608.1337501228;
    [HTTP_CONNECTION] => keep-alive
    [PATH] => C:\Program Files (x86)\ActiveState Komodo Edit 9\
    [SystemRoot] => C:\Windows
    [COMSPEC] => C:\Windows\system32\cmd.exe
    [PATHEXT] => .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.RB;
    [WINDIR] => C:\Windows
    [SERVER_SIGNATURE] => Apache/2.4.17 (Win32) OpenSSL/1.0.2d PHP/5.6.15
        Server at localhost Port 80
    [SERVER_SOFTWARE] => Apache/2.4.17 (Win32) OpenSSL/1.0.2d PHP/5.6.15
    [SERVER_NAME] => localhost
    [SERVER_ADDR] => ::1
    [SERVER_PORT] => 80
    [REMOTE_ADDR] => ::1
    [DOCUMENT_ROOT] => C:/xampp/htdocs
)

```

```
[REQUEST_SCHEME] => http
[CONTEXT_PREFIX] =>
[CONTEXT_DOCUMENT_ROOT] => C:/xampp/htdocs
[SERVER_ADMIN] => postmaster@localhost
[SCRIPT_FILENAME] => C:/xampp/htdocs/belajar_php/index.php
[REMOTE_PORT] => 2191
[GATEWAY_INTERFACE] => CGI/1.1
[SERVER_PROTOCOL] => HTTP/1.1
[REQUEST_METHOD] => GET
[QUERY_STRING] =>
[REQUEST_URI] => /belajar_php/
[SCRIPT_NAME] => /belajar_php/index.php
[PHP_SELF] => /belajar_php/index.php
[REQUEST_TIME_FLOAT] => 1453600127.691
[REQUEST_TIME] => 1453600127
)
```

Seperti yang terlihat, cukup banyak informasi yang ada dalam variabel `$_SERVER`. Sebagian besar memang tidak terlalu jelas fungsinya karena butuh pembahasan lebih lanjut, namun terdapat beberapa variabel yang bisa bermanfaat:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6  </head>
7  <body>
8  <h1>Superglobals Variable $_SERVER</h1>
9  <?php
10     echo "HTTP_HOST = {" . $_SERVER["HTTP_HOST"] . "}" . "<br>";
11     echo "HTTP_USER_AGENT = {" . $_SERVER["HTTP_USER_AGENT"] . "}" . "<br>";
12     echo "SERVER_SOFTWARE = {" . $_SERVER["SERVER_SOFTWARE"] . "}" . "<br>";
13     echo "SERVER_ADDR = {" . $_SERVER["SERVER_ADDR"] . "}" . "<br>";
14     echo "DOCUMENT_ROOT = {" . $_SERVER["DOCUMENT_ROOT"] . "}" . "<br>";
15     echo "SCRIPT_FILENAME = {" . $_SERVER["SCRIPT_FILENAME"] . "}" . "<br>";
16     echo "REQUEST_URI = {" . $_SERVER["REQUEST_URI"] . "}" . "<br>";
17     echo "PHP_SELF = {" . $_SERVER["PHP_SELF"] . "}" . "<br>";
18     echo "SERVER_SIGNATURE = {" . $_SERVER["SERVER_SIGNATURE"] . "}";
19 ?>
20 </body>
21 </html>
```



Gambar: Hasil dari pemanggilan variabel global \$\_SERVER

Kali ini saya mengakses beberapa isi dari variabel \$\_SERVER. Perhatikan bagaimana cara menampilkan array ini. Karena berupa *associative array*, saya menulis **key** array untuk setiap informasi. Berikut penjelasan dari nilai-nilai ini:

**\$\_SERVER["HTTP\_HOST"]**: Berisi informasi mengenai nama server. Karena kita mengaksesnya secara local, isi dari variabel ini adalah: *localhost*.

**\$\_SERVER["HTTP\_USER\_AGENT"]**: Berisi informasi tentang web browser yang digunakan pengunjung. Hasil yang didapat lengkap berserta nomor versi serta informasi lain. Karena saya menjalankan perintah diatas dari web browser Mozilla Firefox, hasilnya adalah: *Mozilla/5.0 (Windows NT 6.1; WOW64; rv:40.0) Gecko/20100101 Firefox/40.0*.

Informasi ini bisa digunakan untuk membedakan tampilan web untuk setiap web browser. Jika pengujung menggunakan web browser *Mozilla Firefox* tampilkan halaman A. Jika menggunakan *Google Chrome*, tampilkan halaman B, dst.

**\$\_SERVER["SERVER\_SOFTWARE"]**: Berisi informasi mengenai software yang digunakan sebagai server. Informasi ini termasuk versi web server serta aplikasi yang memproses kode program. Karena menggunakan **XAMPP 5.6.15**, hasil yang saya dapat adalah: *Apache/2.4.17 (Win32) OpenSSL/1.0.2d PHP/5.6.15*

**\$\_SERVER["SERVER\_ADDR"]**: Berisi informasi mengenai alamat IP server. Karena halaman ini saya jalankan dari localhost, hasilnya adalah **::1**. Ini adalah alamat IP address versi 5 untuk *localhost*. Jika di jalankan dari web hosting, hasilnya bisa berupa alamat IP seperti **199.32.11.01**.

**\$\_SERVER["DOCUMENT\_ROOT"]**: Berisi informasi nama folder tempat file yang sedang dijalankan. Terlihat bahwa saya menjalankan file ini dari *C:/xampp/htdocs*. Hasilnya variabel ini sama dengan magic constant **\_\_DIR\_\_**.

**\$\_SERVER["SCRIPT\_FILENAME"]**: Berisi informasi alamat lengkap file. Hasil yang saya dapat adalah: *C:/xampp/htdocs/belajar\_php/index.php*. Ini sama dengan magic constant **\_\_FILE\_\_**.

**\$\_SERVER["REQUEST\_URI"]**: Berisi informasi tentang alamat yang diminta oleh pengunjung. Dalam contoh ini, saya mengakses halaman ini dari alamat *http://localhost/belajar\_php/*, oleh karena itu hasilnya juga sama. Perhatikan bahwa bagian **index.php**-nya tidak saya tulis.

**\$\_SERVER["PHP\_SELF"]**: Berisi informasi tentang alamat file, relatif kepada *root folder*. **Root folder** adalah istilah yang digunakan untuk menyebut folder 'paling awal' di dalam sebuah

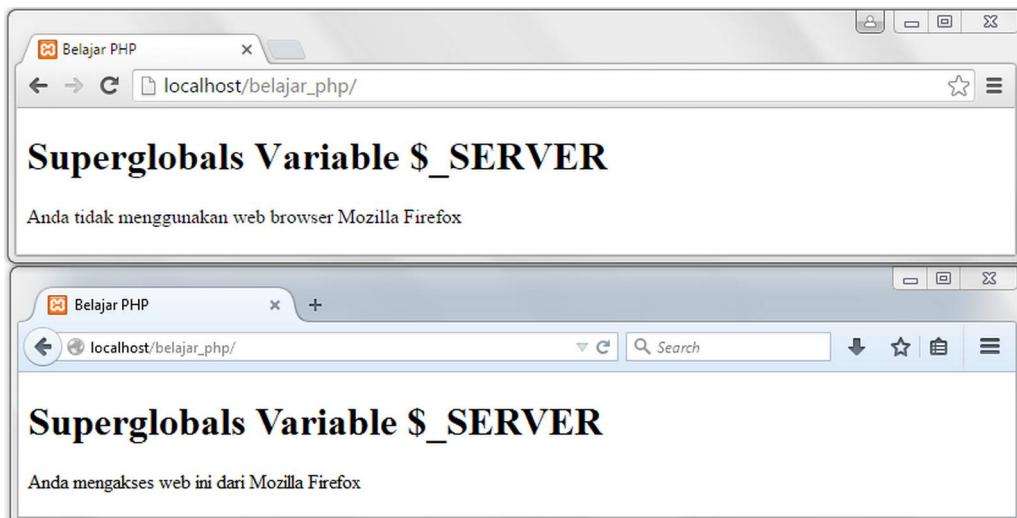
web server. Di dalam XAMPP, folder tersebut adalah folder **htdocs**. Seluruh dokumen yang bisa diakses dari web browser harus berada di dalam folder ini.

Dalam contoh ini hasil yang saya dapat adalah */belajar\_php/index.php*. Artinya, jika jika diakses dari folder **htdocs**, file yang ini berada di */belajar\_php/index.php*.

**`$_SERVER["SERVER_SIGNATURE"]`**: Berisi *signature* atau informasi mengenai web server. Tampilan teks ini sedikit berbeda karena berada di dalam tag `<address>`. Secara default, tag ini akan ditampilkan dengan huruf miring. *Server Signature* terdiri dari nama web server, versi web server, versi PHP, alamat server, serta port yang digunakan.

Sebagai contoh dari penggunaan *global variable* `$_SERVER`, saya bisa membuat kode program yang tampil beda jika diakses dari web browser yang berbeda:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6  </head>
7  <body>
8  <h1>Superglobals Variable $_SERVER</h1>
9  <?php
10     $cari = strpos($_SERVER["HTTP_USER_AGENT"], "Firefox");
11     if ($cari !== false) {
12         echo "Anda mengakses web ini dari Mozilla Firefox";
13     }
14     else {
15         echo "Anda tidak menggunakan web browser Mozilla Firefox";
16     }
17 ?>
18 </body>
19 </html>
```



Gambar: Tampilan teks akan berbeda jika di akses dari web browser selain Firefox

Disini, teks “*Anda mengakses web ini dari Mozilla Firefox*” hanya tampil ketika halaman diakses dari web browser *Mozilla Firefox*. Jika diakses dari web browser lain, teks yang tampil adalah: “*Anda tidak menggunakan web browser Mozilla Firefox*”. Fungsi `strpos()` saya gunakan untuk mencari teks “*Firefox*” dari variabel `$_SERVER[“HTTP_USER_AGENT”]`.

Kode pendekripsi web browser seperti ini dikenal dengan istilah **browser sniffing**. Terdapat berbagai teknik untuk membuat *browser sniffing*. Biasanya ini dilakukan untuk mengatasi perbedaan fitur pada tiap-tiap web browser, terutama pada dukungan kode *CSS* dan *JavaScript*.

## 19.6 Superglobals Variable `$_GET`

Superglobals Variable `$_GET` biasanya dikenal sebagai variabel global penampung nilai form (akan kita bahas dalam bab berikutnya). Tapi sebenarnya variabel ini juga bisa di set tanpa melalui form.

Secara sederhana, variabel `$_GET` berisi nilai yang ditulis ke dalam *query string*. **Query string** adalah sebutan untuk karakter tambahan di belakang URL atau di belakang penulisan alamat dari sebuah situs.

Sebagai contoh, perhatikan penulisan URL atau link berikut:

`http://www.duniaikom.com?s=php`

Setelah alamat situs `www.duniaikom.com`, terdapat tambahan teks “`?s=php`”. Tambahan teks inilah yang disebut sebagai **query string**. Karakter tanda tanya (?) digunakan sebagai pembatas dari alamat situs dengan **query string**.

Dalam contoh diatas, saya mengirim data “`php`” melalui variabel “`s`” ke dalam halaman `www.duniaikom.com`. Nilai “`php`” ini, bisa diakses dari variabel `$_GET[“s”]`.

Agar lebih jelas, mari kita masuk ke dalam contoh prakteknya. Sama seperti contoh variabel `$_SERVER`, saya menampilkan seluruh isi variabel `$_GET` dengan fungsi `print_r()`:

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6  </head>
7  <body>
8  <h1>Superglobals Variable $_GET</h1>
9  <pre>
10 <?php
11     print_r($_GET);
12 ?>
13 </pre>
14 </body>
15 </html>

```

Jika anda menjalankan kode diatas, tidak akan terlihat tampilan apa-apa. Ini karena saya belum menambahkan *query string* di dalam alamat URL.

File diatas saya jalankan dari alamat *http://localhost/belajar\_php/*. Seperti yang sudah kita pelajari, alamat ini sebenarnya akan mengakses file *http://localhost/belajar\_php/index.php*. Untuk menambahkan *query string*, saya bisa menulis salah satu dari:

```

http://localhost/belajar_php/?nama=Siska
http://localhost/belajar_php/index.php?nama=Siska

```

Kedua URL diatas akan mengirim varibel GET “**nama**” dengan nilai “**Siska**” ke dalam halaman **index.php**. Berikut hasil yang saya dapat:



Gambar: Menampilkan nilai query string dengan \$\_GET

Terlihat, varibel **\$\_GET** berisi hasil dari *query string*. Silahkan anda coba mengubah *query string* ini, misalnya dengan *index.php?nama=Alex*, atau dengan *index.php?nama=DuniaIlkom*.

Untuk menambah nilai baru, tambahkan tanda “*ampersand*” ( & ) sebagai karakter pemisah di dalam *query string*. Sebagai contoh, untuk mengirim nilai **nama** dan **alamat**, saya bisa menggunakan URL berikut:

http://localhost/belajar\_php/index.php?nama=Siska&alamat=Medan

Seluruh *query string* ini ditulis serangkai tanpa ada tanda spasi. Berikut hasilnya:



Gambar: Mengakses 2 variabel dari query string

Anda bisa menambahkan berapa pun nilai di dalam *query string* ini. Untuk beberapa web browser tua seperti *Internet Explorer*, karakter untuk *query string* dibatasi hingga 2.048 karakter.

Dalam kode program sebelumnya, saya menampilkan seluruh variabel `$_GET` dengan perintah `print_r()`. Bagaimana cara mengakses satu variabel saja? cukup dengan menulis *associative array*-nya:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6  </head>
7  <body>
8  <h1>Superglobals Variable $_GET</h1>
9  <?php
10     $nama=$_GET["nama"];
11     $alamat=$_GET["alamat"];
12
13     echo "Nama siswa = $nama <br>";
14     echo "Alamat siswa = $alamat";
15 ?>
16 </body>
17 </html>
```



Gambar: Mengambil variabel dari query string secara individu

Sekarang, kita bisa memproses terlebih dahulu nilai-nilai yang ada di dalam variabel `$_GET` ini.

## 19.7 Mengirim Pesan Antar Halaman Dengan Query String

Dalam prakteknya, *query string* umum digunakan sebagai sarana “pengirim pesan” antar halaman. Sebagai contoh, buatlah sebuah halaman baru dengan kode program berikut:

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6  </head>
7  <body>
8  <h1>Halaman Proses</h1>
9  <?php
10     echo "Nama : " . $_GET["nama"] . "<br>";
11     echo "Alamat : " . $_GET["alamat"] . "<br>";
12     echo "Umur : " . $_GET["umur"] . "<br>";
13 ?>
14 </body>
15 </html>

```

Save kode diatas sebagai **proses.php** di folder *htdocs*. Saya yakin anda sudah paham apa maksud kode diatas. Disini saya ingin menampilkan 3 variabel yang berasal dari *query string*, yakni **nama**, **alamat**, dan **umur**.

Sekarang, buka file **index.php**, lalu ubah kodennya menjadi sebagai berikut:

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6  </head>
7  <body>
8  <h1>Query String</h1>
9  <?php
10     $nama="Dewi";
11     $alamat="Bandung";
12     $umur="23";
13
14     $query_string="?nama={$nama}&alamat={$alamat}&umur={$umur}";
15 ?>
16 <a href="proses.php<?php echo $query_string; ?>">Link Menuju proses.php</a>
17 </body>
18 </html>

```

Dalam halaman ini saya membuat beberapa variabel lalu menyatukannya menjadi *query string*. *Query string* ini kemudian ‘disambung’ dan ditampilkan menjadi sebuah link ke halaman `proses.php`. Berikut hasilnya:



Gambar: Halaman index.php

Perhatikan tampilan di sudut kiri bawah, inilah *link preview* yang ditampilkan web browser. Kita bisa melihat kemana alamat yang akan dituju. Ketika link ini di-klik, hasilnya adalah:



Gambar: Query string ‘ditangkap’ oleh halaman proses.php

Seluruh variabel di dalam *query string* langsung ditampilkan sesuai dengan nilai yang dikirim. Cara seperti inilah yang banyak digunakan sebagai sarana mengirim pesan antar halaman.

Jika anda mengetik alamat: [www.duniaikom.com?s=php](http://www.duniaikom.com?s=php)<sup>1</sup> di web browser yang terhubung ke internet, *query string* “s=php” akan di proses sebagai fitur pencarian dengan kata kunci “php”. Ini karena di situs duniaikom (dan situs berbasis wordpress lainnya), variabel “s” di dalam *query string* akan diproses sebagai kata kunci untuk pencarian.

Contohnya, jika ingin mencari seluruh artikel yang mengandung kata “css” di duniaikom, bisa menggunakan alamat [www.duniaikom.com?s=css](http://www.duniaikom.com?s=css)<sup>2</sup>.



Gambar: Pencarian di duniaikom dengan url [www.duniaikom.com?s=css](http://www.duniaikom.com?s=css)

Situs search engine seperti google juga menggunakan *query string* untuk proses pencarian. Sebagai contoh, untuk mencari kata “php” di google, saya bisa menulis: [www.google.com/search?q=php](http://www.google.com/search?q=php)<sup>3</sup>.

Untuk mencari dengan keyword “css”, bisa mengakses alamat: [www.google.com/search?q=css](http://www.google.com/search?q=css)<sup>4</sup>. Saya tinggal menukar nilai variabel “q” yang menjadi *query string* di google.

Kembali ke pembahasan kita tentang variabel `$_GET`, dapatkah anda melihat masalah dari kode berikut ini?

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6  </head>
7  <body>
8  <h1>Query String</h1>
9
10 <?php
11     $nama="Dewi & Thia";
12     $alamat="Bandung & Medan";
13     $umur="23";
14

```

<sup>1</sup><http://www.duniaikom.com?s=php>

<sup>2</sup><http://www.duniaikom.com?s=css>

<sup>3</sup><https://www.google.com/search?q=php>

<sup>4</sup><https://www.google.com/search?q=css>

```

15  $query_string="?nama={$nama}&alamat={$alamat}&umur={$umur}";
16  ?>
17 <a href="proses.php<?php echo $query_string; ?>">Link Menuju proses.php</a>
18 </body>
19 </html>

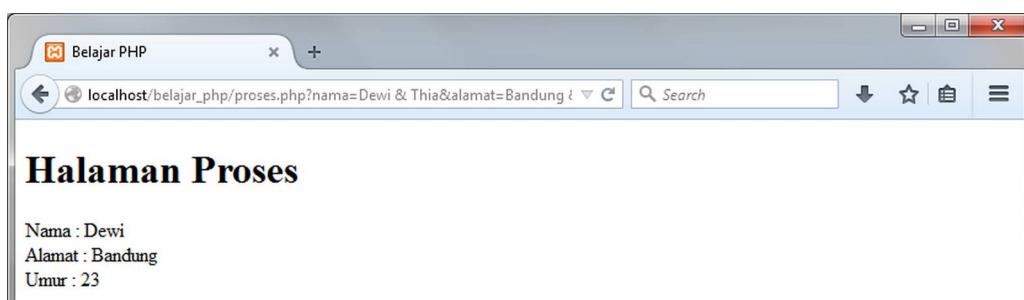
```

Jika anda belum bisa menemukan sesuatu yang salah, mari kita jalankan kode diatas:



Gambar: Tampilan halaman index.php

Berikut hasilnya dihalaman proses.php:



Gambar: Tampilan halaman proses.php

Terlihat bahwa data yang ditampilkan untuk variabel **nama** hanya 1, yakni: “**Dewi**”. Padahal saya mengirim 2 buah nama di dalam variabel ini: “**Dewi & Thia**”. Apa yang terjadi?

Ini disebabkan karena karakter *ampersand* ( & ) yang di kirim dalam string “**Dewi & Thia**”, dianggap sebagai karakter pembatas *query string*, dan bukan sebagai nilai dari variabel **nama**. Hal yang sama juga terjadi untuk nilai alamat “**Bandung & Medan**”.

Untuk mengatasi masalah ini, saya harus mengubah karakter “&” menjadi ‘karakter lain’ agar tidak dianggap sebagai pembatas *query string*. PHP menyediakan fungsi **urlencode()** dan **rawurlencode()** untuk mengatasi masalah ini.

## 19.8 Function urlencode() dan rawurlencode()

Fungsi **urlencode()** dan **rawurlencode()** digunakan untuk mengubah karakter khusus di dalam *query string* menjadi ‘karakter lain’ agar bisa diproses sebagai nilai *variabel*. Karakter khusus ini disebut sebagai **URL Reserved Character**.

Selain tanda “&” terdapat beberapa karakter lain yang bisa menjadi ‘masalah’ ketika di tulis di dalam *query string*. Berikut daftar karakter **URL Reserved Character**:

!	*	'	(	)	;	:	@	&	=	+	\$	,	/	?	#	[	]
---	---	---	---	---	---	---	---	---	---	---	----	---	---	---	---	---	---

Gambar: URL Reserved Character

Agar karakter-karakter ini dapat dianggap sebagai bagian dari variabel, kita harus mengkonversinya menjadi *percent-encoded*. **Percent-encoded** dibuat dari gabungan karakter ‘%’, kemudian diikuti dengan nomor heksadesimal ASCII. Sebagai contoh, karakter ‘&’ berada di urutan 36 desimal daftar karakter ASCII atau 26 heksadesimal. Dengan demikian, *percent-encoded* untuk karakter ‘&’ adalah %26.

Berikut tabel konversi **URL Reserved Character** menjadi *percent-encoded*:

!	*	'	(	)	;	:	@	&	=	+	\$	,	/	?	#	[	]
21%	23%	24%	26%	27%	28%	29%	%2A	%2B	%2C	%2F	%3A	%3B	%3D	%3F	40%	%5B	%5D

Gambar: Konversi URL Reserved Character menjadi Percent-encoded

Menggunakan PHP, kita tidak perlu menulis karakter ini secara manual. Fungsi **urlencode()** dan **rawurlencode()** akan melakukannya secara otomatis.

Perbedaan mendasar antara kedua fungsi ini adalah ketika menerjemahkan karakter ‘spasi’. Fungsi **urlencode()** akan mengkonversi karakter spasi menjadi tanda tambah ( + ). Sedangkan fungsi **rawurlencode()** akan mengkonversi karakter spasi menjadi *percent-encoded* “%21”.

Berikut penambahan fungsi **urlencode()** dalam kode program kita sebelumnya:

```

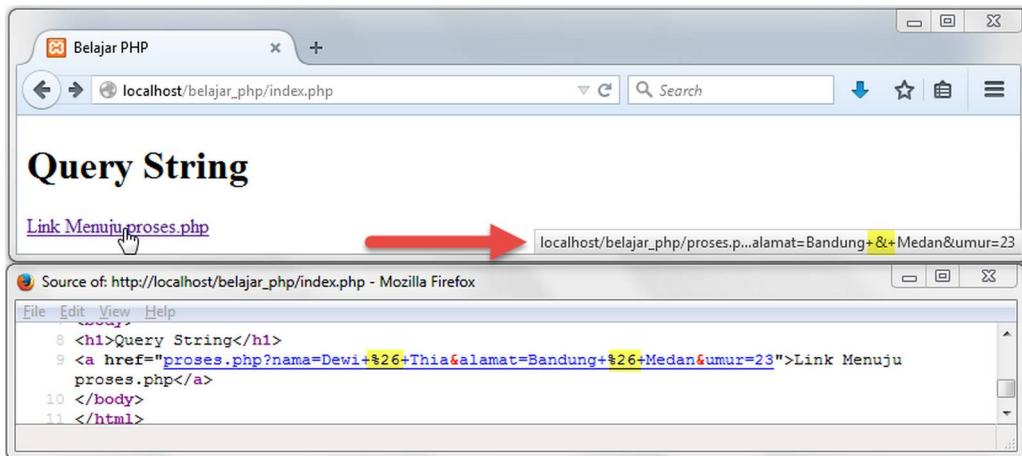
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6  </head>
7  <body>
8  <h1>Query String</h1>
9  <?php
10     $nama=urlencode("Dewi & Thia");
11     $alamat=urlencode("Bandung & Medan");
12     $umur="23";
13
14     $query_string="?nama={$nama}&alamat={$alamat}&umur={$umur}";
15 ?>
16 <a href="proses.php<?php echo $query_string; ?>">Link Menuju proses.php</a>
17 </body>
18 </html>

```

Dalam kode diatas saya menggunakan fungsi **urlencode()** sebelum menyambung nilai variabel

ke dalam *query string*. Perlu dicatat bahwa fungsi `urlencode()` dilakukan bukan pada seluruh *query string*, tetapi hanya pada setiap variabel. Jika saya menulis `urlencode($query_string)`, maka karakter “&” yang berfungsi sebagai pembatas variabel juga ikut dikonversi.

Berikut tampilan kode diatas:

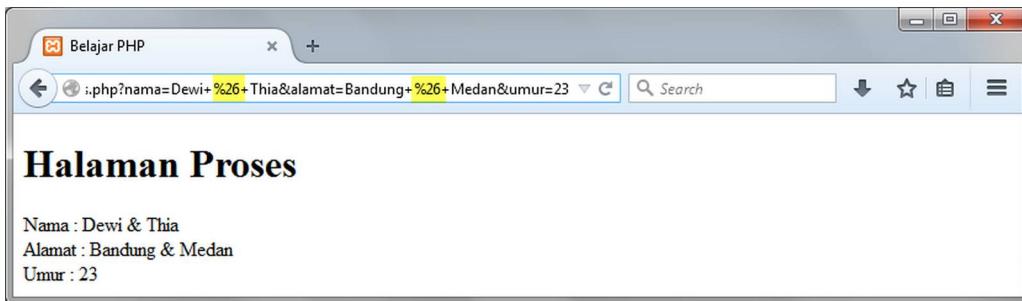


Gambar: Halaman index.php dengan fungsi urlencode()

Ada 2 hal yang bisa kita lihat. Pertama, karakter spasi telah diubah menjadi tanda tambah. Ini juga terlihat ketika saya meletakkan cursor mouse diatas link. Kedua, karakter “&” telah diubah menjadi %26.

Perhatikan karakter “&” di dalam web browser, tetapi tampil sebagai karakter “&” dan bukan %26. Namun ketika dilihat dari *source code* kode HTMLnya, sebenarnya yang diproses adalah %26.

Berikut hasil akhir ketika link di-klik:



Gambar: Halaman proses.php yang menampilkan query string

Sekarang, seluruh variabel telah sampai ke halaman `proses.php`. Dari tampilan URL juga terlihat bahwa tanda ‘+’ digunakan sebagai pengganti spasi, dan tanda %26 sebagai pengganti karakter “&”.

Sebenarnya, di halaman `proses.php` kita harus mengkonversi kembali karakter yang sudah di *encode* ini ke karakter awal (dikenal dengan proses *decode*). PHP menyediakan fungsi `urldecode()` untuk keperluan ini.

Tetapi saat kita mengakses nilai *query string* dengan variabel `$_GET`, PHP sudah melakukan proses *decoding* secara otomatis, dimana tanda “+” akan dikembalikan sebagai spasi, dan tanda

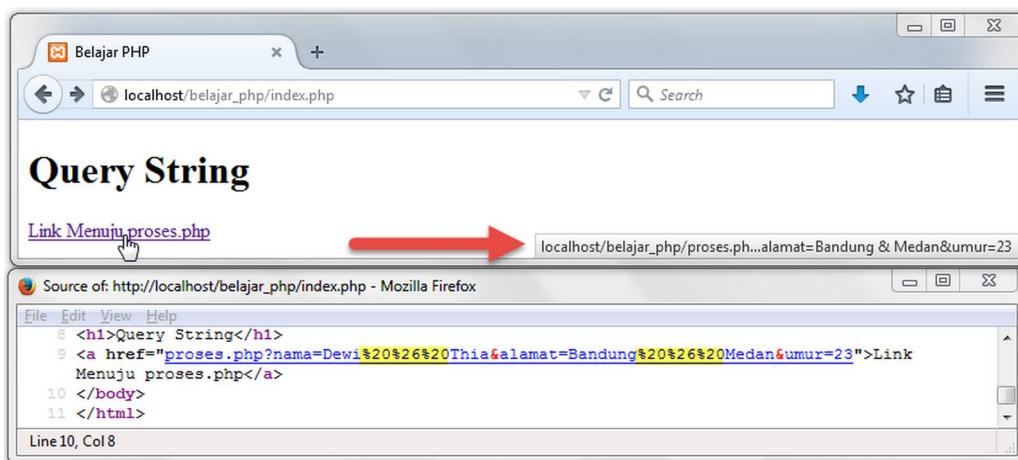
%26 dikembalikan ke dalam bentuk “ & ”.

Bagaimana dengan fungsi **rawurlencode()**? Berikut contoh penggunaannya:

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6  </head>
7  <body>
8  <h1>Query String</h1>
9
10 <?php
11     $nama=rawurlencode( "Dewi & Thia");
12     $alamat=rawurlencode( "Bandung & Medan");
13     $umur="23";
14
15     $query_string="?nama={$nama}&alamat={$alamat}&umur={$umur}";
16 ?>
17 <a href="proses.php<?php echo $query_string; ?>">Link Menuju proses.php</a>
18 </body>
19 </html>

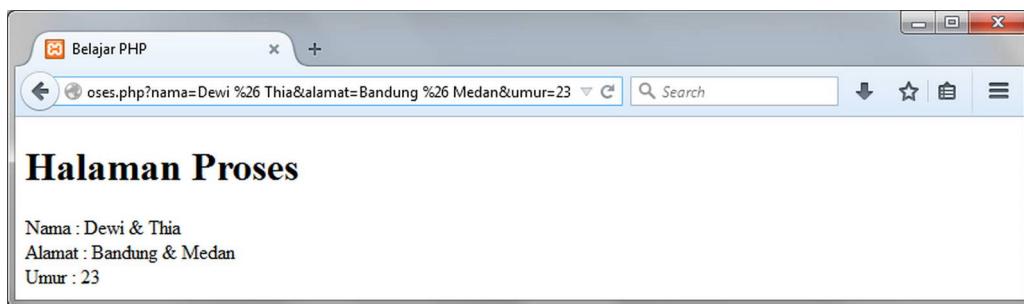
```



Gambar: Halaman index.php dengan fungsi rawurlencode()

Terdapat beberapa hal yang berbeda. Sewaktu saya meletakkan cursor mouse di atas link, terlihat bahwa link ini seperti tidak diproses apa-apa. Tanda spasi tetap ditampilkan sebagai spasi, dan tanda *ampersand* juga tetap ditampilkan sebagai “&”.

Namun ketika dilihat dari *source code* HTMLnya, karakter spasi dan *ampersand* ini sudah dikonversi ke dalam bentuk *percent-encoded*. Baik, mari kita klik link tersebut:



Gambar: Halaman proses.php yang menampilkan query string

Kembali perhatikan penulisan alamat URL. Tanda spasi tetap tampil sebagai spasi, dan bukan %20. Ini semua merupakan “ulah” web browser yang menampilkan %20 sebagai spasi. Sedangkan karakter *ampersand* tetap ditampilkan sebagai %26.

Sama seperti fungsi `urldecode()`, juga terdapat fungsi `rawurldecode()` yang bisa digunakan untuk mengembalikan *percent-encoded* kedalam bentuk asalnya (proses *decoding*), tapi hal ini juga tidak perlu kita lakukan.

Sekarang pertanyaannya, fungsi manakah sebaiknya gunakan untuk memproses *query string*? Fungsi `urlencode()` atau fungsi `rawurlencode()`?

Seperti yang sudah kita lihat, perbedaan dari kedua fungsi ini hanya pada saat memproses karakter spasi. Keduanya juga diproses dengan sempurna oleh web browser.

Dari referensi<sup>5</sup> yang saya dapat, fungsi `rawurlencode()` sebaiknya digunakan untuk memproses alamat URL sebelum tanda “?”, yakni untuk bagian **sebelum** *query string*. Untuk bagian setelah tanda “?”, yakni bagian *query string*, sebaiknya menggunakan fungsi `urlencode()`.

Contoh kode program berikut menggabungkan kedua hal ini:

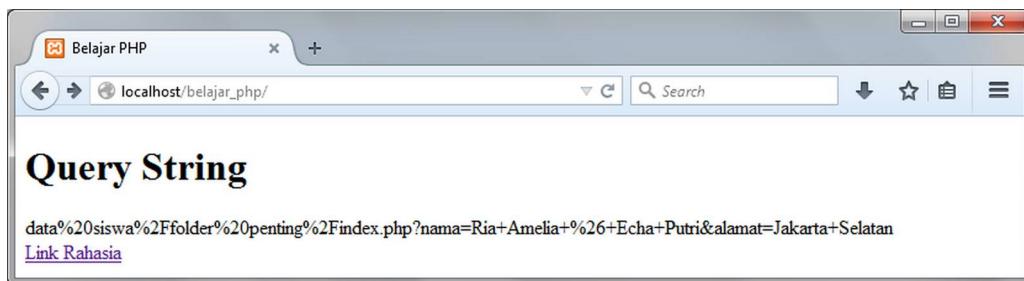
```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6  </head>
7  <body>
8  <h1>Query String</h1>
9  <?php
10     $alamat_folder=rawurlencode("data siswa/folder penting/index.php");
11     $nama=urlencode("Ria Amelia & Echa Putri");
12     $alamat=urlencode("Jakarta Selatan");
13     $url="${alamat_folder}?nama={$nama}&alamat=$alamat";
14
15     echo $url."<br>";
16 ?>
17 <a href="<?php echo $url; ?>">Link Rahasia</a>

```

<sup>5</sup><http://stackoverflow.com/questions/996139/urlencode-vs-rawurlencode>

```
18  </body>
19  </html>
```



Gambar: Gabungan dari fungsi urlencode() dan fungsi rawurlencode()

Dalam kode diatas, saya membuat sebuah link yang jika tanpa proses encoding akan tampil sebagai:

```
data siswa/folder penting/index.php?nama=Ria Amelia &
Echa Putri&alamat=Jakarta Selatan
```

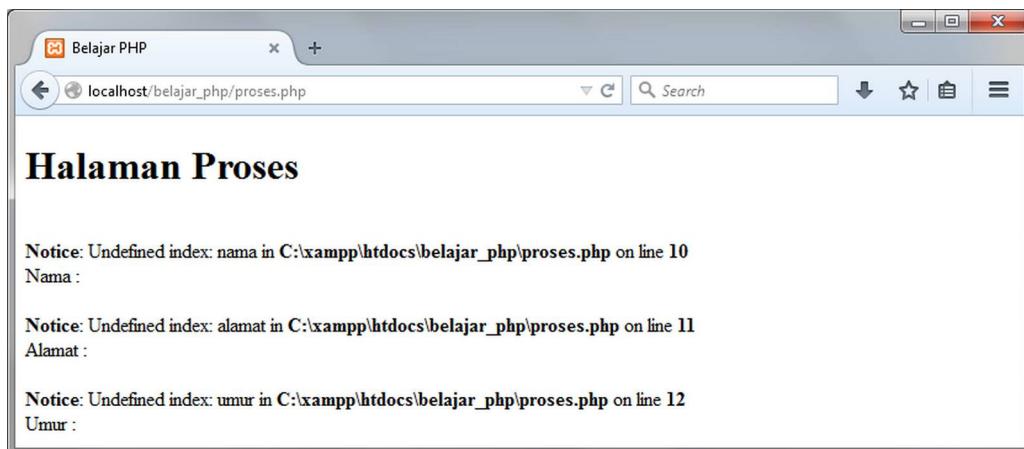
Terlihat bahwa link ini memiliki banyak karakter khusus. Saya harus mengkonversinya terlebih dahulu.

Untuk bagian sebelum tanda “?”, diproses menggunakan fungsi **rawurlencode()**, hasilnya menjadi: `data%20siswa%2Ffolder%20penting%2Findex.php`.

Sedangkan bagian setelah tanda “?” diproses dengan fungsi **urlencode()**, yang akan menjadi: `nama=Ria+Amelia+%26+Echa+Putri&alamat=Jakarta+Selatan`.

Alasan di balik pemilihan ini adalah, hasil dari fungsi **rawurlencode()** dianggap lebih cocok ketika diakses oleh sistem, seperti struktur folder atau nama file, sedangkan fungsi **urlencode()** dianggap lebih cocok untuk bagian *query string*.

Sebelum menutup bab ini, saya ingin memperlihatkan apa yang terjadi ketika halaman **proses.php** di akses secara langsung tanpa *query string*:



Gambar: Error karena URL tanpa query string

Pesan error “*Notice: Undefined index*” terjadi karena di dalam kode program **proses.php** saya mencoba mengakses variabel `$_GET[“nama”]` yang tidak terdefinisi (karena memang tidak ada *query string* ). Untuk mengatasi masalah ini, kita harus memastikan terlebih dahulu apakah variabel `$_GET[“nama”]` ada atau tidak. Jika berisi nilai, baru dipindahkan ke dalam variabel `$nama`, jika tidak ada, lanjut ke kode program berikutnya.

Solusi untuk hal ini akan saya bahas dalam bab berikutnya: Memproses form dengan PHP

# 20. Form Processing

Pemrosesan form mungkin menjadi salah satu materi yang paling menarik di dalam buku ini. Form-lah yang membuat sebuah website menjadi dinamis. Melalui form juga pengguna bisa melakukan interaksi, apakah itu mengisi form pendaftaran, login, form pencarian, komentar, dll.

Di balik itu semua, pemrosesan form juga menjadi materi yang sangat kompleks. Setiap data yang diinput melalui form harus kita periksa terlebih dahulu, apakah sudah sesuai atau tidak. Proses ini dikenal juga dengan **validasi form**.

Dalam bab ini, saya akan membahas cara penanganan data form dalam PHP, mulai dari cara menampilkan hasil form, proses validasi, menampilkan pesan kesalahan, re-populate isian form, hingga cara penggunaan *regular expression*. Di akhir bab akan ada sebuah *case study* membuat form utuh dengan menggabungkan seluruh materi yang telah dibahas.



Contoh kode program yang akan saya gunakan mungkin terasa rumit dan lumayan panjang. Jika anda kurang paham, silahkan pelajari secara perlahan :)

## 20.1 Menampilkan Hasil Form

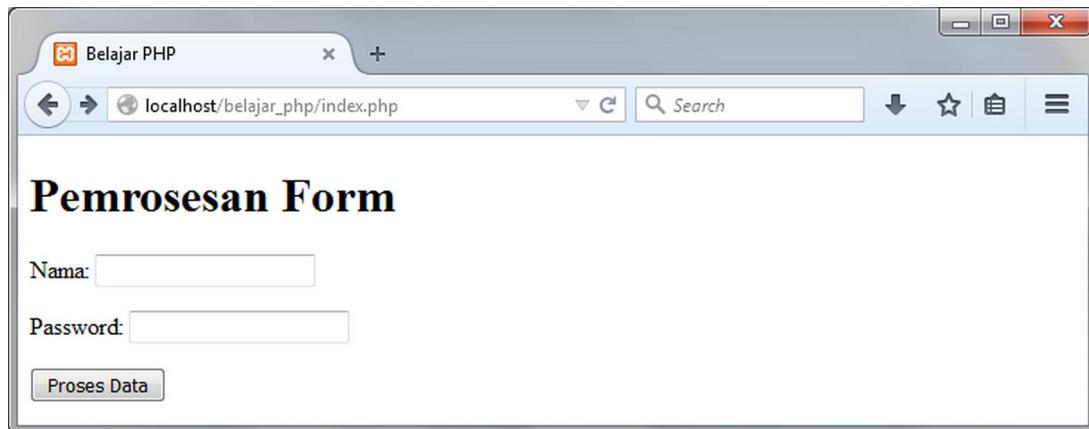
Menampilkan hasil sebuah form sebenarnya cukup mudah, kita tinggal mengakses variabel *superglobals* `$_GET` atau `$_POST`. Sebelum membahas perbedaan keduanya, mari siapkan sebuah halaman yang berisi form. Berikut kode HTML yang saya gunakan:

index.php

---

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar PHP</title>
6 </head>
7 <body>
8 <h1>Pemrosesan Form</h1>
9   <form action="proses.php" method="get">
10    <p>Nama: <input type="text" name="nama"></p>
11    <p>Password: <input type="password" name="password"></p>
12    <input type="submit" value="Proses Data" >
13  </form>
14 </body>
15 </html>
```

---



Gambar: Sebuah form HTML (index.php)

Kode diatas murni HTML dan tidak ada kode PHP sama sekali. Tag HTML untuk membuat form tidak akan saya bahas lagi, karena saya berasumsi anda sudah paham cara membuat form dengan HTML. Silahkan simpan halaman tersebut sebagai *index.php*.

Bagian yang perlu kita diperhatikan adalah atribut **action** dan **method** di dalam tag pembuka `<form>`:

```
<form action="proses.php" method="get">
```

Atribut **action** diisi dengan alamat file dimana form akan diproses. Dalam contoh ini saya akan mengirim hasil form ke halaman *proses.php*. Halaman inilah yang akan memproses dan menampilkan isi form. Kita akan membuatnya sesaat lagi.

Atribut **method** berfungsi untuk mengatur bagaimana hasil form ini dikirimkan. Terdapat 2 jenis **method**, yakni "get" dan "post". Dalam contoh diatas saya menggunakan **method="get"**.

Selanjutnya, saya akan menyiapkan file kedua untuk menampilkan hasil form, yakni halaman *proses.php*, berikut kode untuk halaman tersebut:

#### proses.php

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar PHP</title>
6 </head>
7 <body>
8 <h1>Halaman Proses</h1>
9 <pre>
10 <?php
11   print_r($_GET);
12 ?>
13 </pre>
14 </body>
15 </html>
```

Sama seperti contoh bab *Superglobals Variable* sebelumnya, saya hanya menggunakan 1 baris kode PHP, yakni `print_r($_GET)`. Fungsi `print_r()` bertujuan untuk menampilkan seluruh isi variabel superglobals `$_GET`. Mari kita jalankan.



Gambar: Hasil form ditampilkan dalam halaman proses.php

Jika semuanya berjalan lancar, hasil form dari halaman `index.php` akan tampil di halaman `proses.php`. Syarat agar ini bisa bekerja, halaman `index.php` dan `proses.php` harus berada dalam folder yang sama. Jika berbeda, dapat disesuaikan dengan mengatur penulisan atribut `action`.

Mari kita bahas kembali proses kerja untuk menampilkan form ini.

Pertama, perhatikan tampilan di halaman `proses.php`, array *superglobals* `$_GET` memiliki *key* "nama" dan "password". Kedua *key* ini sebenarnya berpasangan dengan atribut `name` di dalam struktur form.

Di dalam form, saya membuat 2 buah kotak input:

```
<input type="text" name="nama">
<input type="password" name="password">
```

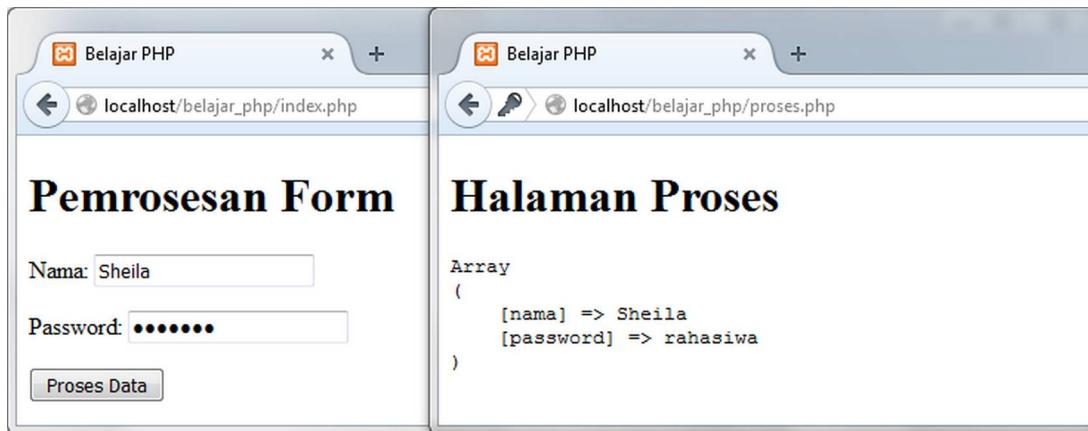
Atribut `name` akan 'ditangkap' oleh variabel `$_GET["name"]` sedangkan atribut `password` akan ditangkap oleh variabel `$_GET["password"]`. Silahkan anda coba mengganti atribut ini menjadi nilai lain, dan *key*-nya juga akan berubah.

Yang kedua, perhatikan alamat URL saat form ditampilkan di halaman `proses.php`. Anda bisa melihat 'sesuatu' ditambahkan di bagian *query string*:

`http://localhost/belajar_php/proses.php?nama=Alesandro&password=123456`

Inilah efek dari penggunaan `method="get"` dari tag pembuka form. Selain `method="get"`, kita juga bisa menggunakan `method="post"`. Mari kita coba.

Buka kembali halaman `index.php`, lalu ubah di bagian atribut `method="get"` menjadi `method="post"`. Kemudian di halaman `proses.php`, ubah kode `print_r($_GET)` menjadi `print_r($_POST)`. Berikut hasilnya:



Gambar: Hasil pemrosesan form menggunakan method POST

Sekilas tidak tampak perbedaan, baik `$_GET` maupun `$_POST` sama-sama digunakan untuk menampilkan hasil form. Namun perhatikan di alamat URL, sekarang tidak terlihat lagi *query string*. Inilah perbedaan utama antara method **get** dengan method **post**.

## 20.2 Perbedaan Method GET dan POST

Dalam praktek sebelumnya, kita telah mempelajari cara menampilkan nilai form menggunakan method **get** dan **post**. Yang mana sebaiknya digunakan?

Perbedaan paling jelas dari kedua method ini adalah: “**get**” akan mengirimkan hasil form di dalam *query string*, sedangkan “**post**” tidak. Secara teknis, method **post** mengirim hasil form di dalam **HTTP header**, oleh karena itulah tidak terlihat di dalam web browser.

Method **post** sebaiknya digunakan untuk form yang bersifat rahasia dan bisa mempengaruhi suatu kondisi di server. Contohnya, apabila hasil form akan disimpan ke dalam database, seperti form register, form login, form komentar, dll.

Method **get** sebaiknya digunakan untuk form yang bersifat terbuka dan tidak mengubah apa-apa di sisi server. Contohnya seperti form pencarian. Dengan membuat hasil input form terlihat jelas dalam URL, pengunjung web bisa dengan mudah menebak ‘alur kerja’ dari situs kita. Seperti praktek dalam bab sebelumnya, pencarian di situs duniaIlkom bisa dilakukan dengan mengubah *query string*.

Dari segi bahasa, “**get**” berarti *ambil* atau *mengambil*. Jika form yang kita buat akan mengambil suatu informasi dari server, sebaiknya gunakan method **get**. Sedangkan “**post**” bisa diartikan *memasukkan sesuatu*. Oleh karena itu jika form yang dirancang nantinya akan disimpan ke dalam database, gunakan method **post**.

## 20.3 Menampilkan Hasil Form Secara Individu

Contoh kode program yang kita praktekkan sebelumnya menampilkan isi form secara global melalui fungsi `print_r()`. Bagaimana cara mengambil hasil form ini secara individu? Caranya sama seperti variabel *superglobals* lain, yakni dengan menulis key-arraynya. Berikut perubahan kode di halaman `proses.php`:

**proses.php**

---

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar PHP</title>
6 </head>
7 <body>
8 <h1>Halaman Proses</h1>
9 <?php
10  echo "Nama : ". $_POST["nama"] . "<br>";
11  echo "Password : ". $_POST["password"] . "<br>";
12 ?>
13 </body>
14 </html>
```

---

Disini, saya menampilkan hasil form secara terpisah dengan perintah **echo**.

*Baik, saya sudah bisa menampilkan isi form dari text input. Tapi bagaimana dengan objek form lain seperti checkbox, radio button dan text area?*

Pada prinsipnya tetap sama, atribut **name** dari object form akan menjadi key dari variabel **\$\_GET** atau **\$\_POST**. Sebagai contoh, saya akan menampilkan form utuh yang terdiri dari berbagai objek form:

**index.php**

---

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar PHP</title>
6   <style>
7     h1 {
8       margin:0;
9     }
10    div {
11      width:400px;
12    }
13    p{
14      margin:0;
15    }
16    fieldset{
17      margin-top:10px;
18    }
19    input[type="text"], input[type="password"], select, textarea {
```

```
20      margin-left:10px;
21      margin-bottom:10px;
22  }
23  input[type="checkbox"] {
24      margin-left:120px;
25  }
26  label {
27      width:110px;
28      float:left;
29  }
30  label[for="html"], label[for="css"], label[for="php"] {
31      float: initial;
32  }
33  </style>
34 </head>
35 <body>
36
37 <div>
38 <h1>Form Registrasi</h1>
39
40 <form action="proses.php" method="post">
41 <fieldset>
42 <legend>Biodata</legend>
43 <p>
44     <label for="nama">Nama : </label>
45     <input type="text" name="nama" id="nama">
46 </p>
47 <p>
48     <label for="nim">NIM : </label>
49     <input type="text" name="nim" id="nim">
50 </p>
51 <p>
52     <label for="tgl" >Tanggal Lahir : </label>
53     <select name="tgl" id="tgl">
54         <?php
55             for ($i = 1; $i <= 31; $i++) {
56                 echo "<option value = $i >";
57                 echo str_pad($i, 2, "0", STR_PAD_LEFT);
58                 echo "</option>";
59             }
60         ?>
61     </select>
62     <select name="bln">
63         <option value=1>Januari</option>
64         <option value=2>Februari</option>
65         <option value=3>Maret</option>
```

```
66      <option value=4>April</option>
67      <option value=5>Mei</option>
68      <option value=6>Juni</option>
69      <option value=7>Juli</option>
70      <option value=8>Agustus</option>
71      <option value=9>September</option>
72      <option value=10>Oktober</option>
73      <option value=11>Nopember</option>
74      <option value=12>Desember</option>
75  </select>
76  <select name="thn">
77      <?php
78      for ($i = 1980; $i <= 2016; $i++) {
79          echo "<option value = $i > $i </option>";
80      }
81      ?>
82  </select>
83 </p>
84 <p>
85     <label for="alamat">Alamat </label>
86     <textarea name="alamat" id="alamat" cols="25"></textarea>
87 </p>
88 <p>
89     <label>Jenis kelamin</label>
90     <label><input type="radio" name="kel" value="laki2">
91     Laki-laki</label>
92     <label><input type="radio" name="kel" value="perempuan">
93     Perempuan</label>
94 </p>
95 </fieldset>
96
97 <fieldset>
98 <legend align="">User Account</legend>
99 <p>
100    <label for="username">Username</label>
101    <input type="text" name="username" id="username"/>
102 </p>
103 <p>
104    <label for="email">Email </label>
105    <input type="text" name="email" id="email" />
106 </p>
107 <p>
108    <label for="pass">Password</label>
109    <input type="password" name="password" id="pass" />
110 </p>
111 <p>
```

```
112     <label for="repass">Ulangi Password</label>
113     <input type="password" name="repassword" id="repass" />
114     </p>
115 </fieldset>
116
117 <fieldset>
118 <legend>Resolusi Tahun Ini</legend>
119     <p>
120         <input type="checkbox" name="target1" value="HTML" id="html">
121         <label for="html"> Menguasai HTML</label>
122     </p>
123     <p>
124         <input type="checkbox" name="target2" value="CSS" id="css">
125         <label for="css"> Paham CSS</label>
126     </p>
127     <p>
128         <input type="checkbox" name="target3" value="PHP" id="php">
129         <label for="php"> Mastering PHP</label>
130     </p>
131 </fieldset>
132     <br>
133     <p>
134         <input type="submit" value="Kirim Data">
135     </p>
136 </form>
137
138 </div>
139
140 </body>
141 </html>
```

---

Kode HTML diatas cukup panjang, selain itu saya juga menambahkan sedikit CSS untuk mempercantik tampilan form. Sebenarnya kode diatas adalah contoh kasus yang ada di buku **HTML Uncover**. Jika anda kurang paham tentang kode HTML yang ada, silahkan buka kembali buku **HTML Uncover** :)

Khusus untuk objek form tanggal dan tahun, saya menggunakan kode PHP untuk membuat angka secara otomatis. Yang perlu diperhatikan adalah, semua objek form memiliki atribut **name**. Atribut inilah yang akan menjadi *key* di variabel `$_POST`.

**Form Registrasi**

**Biodata**

Nama :

NIM :

Tanggal Lahir :

Alamat

Jenis kelamin  Laki-laki  Perempuan

**User Account**

Username

Email

Password

Ulangi Password

**Resolusi Tahun Ini**

Menguasai HTML  
 Paham CSS  
 Mastering PHP

**Kirim Data**

Gambar: Form yang dirancang dari buku HTML Uncover

Bagaimana cara menampilkan seluruh form ini? Berikut kode untuk halaman *proses.php*:

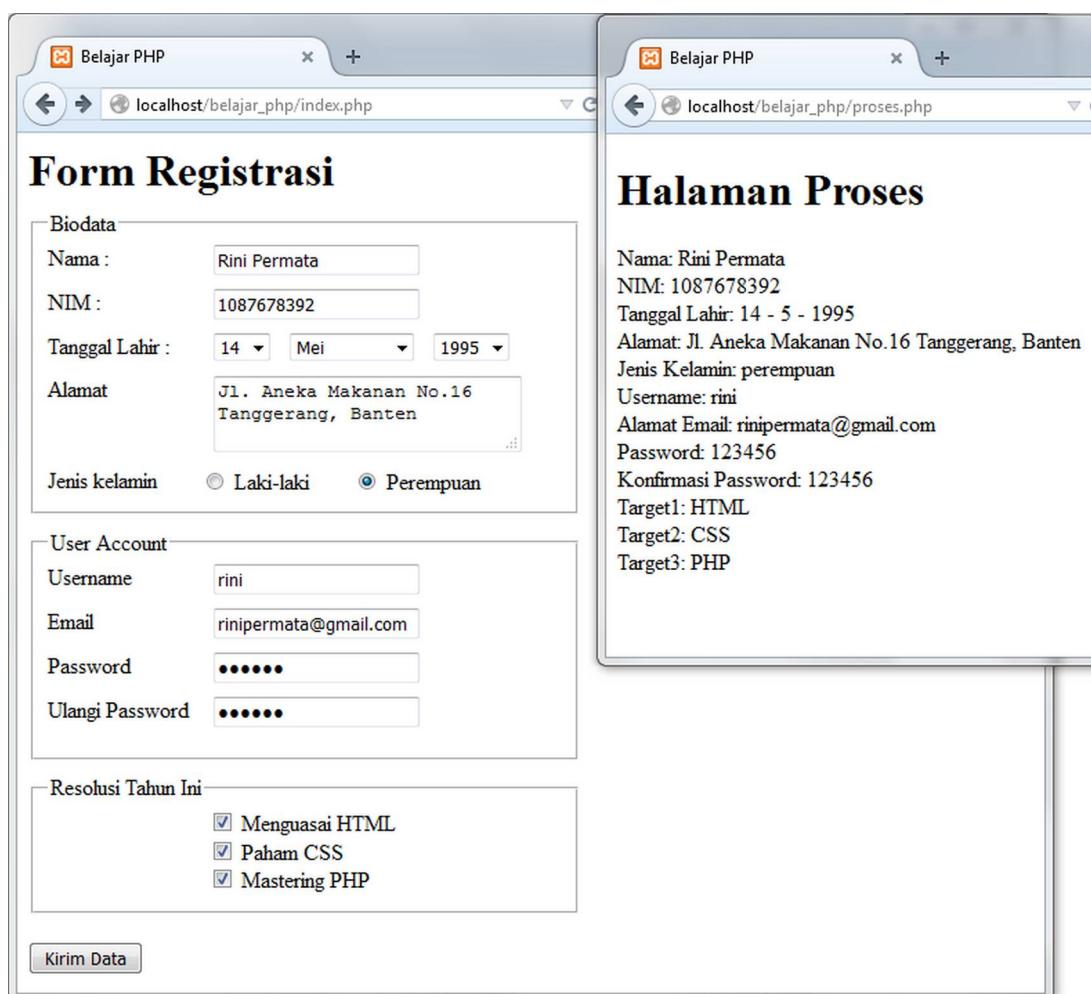
**proses.php**

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6  </head>
7  <body>
8  <h1>Halaman Proses</h1>
9  <?php
10     echo "Nama: ".$_POST["nama"]."<br>";
11     echo "NIM: ".$_POST["nim"]."<br>";
```

```

12 echo "Tanggal Lahir: ".$_POST["tgl"]." - ".$_POST["bln"]." .
13                                     " - ".$_POST["thn"]." <br> ";
14 echo "Alamat: ".$_POST["alamat"]. "<br> ";
15 echo "Jenis Kelamin: ".$_POST["kel"]. "<br> ";
16 echo "Username: ".$_POST["username"]. "<br> ";
17 echo "Alamat Email: ".$_POST["email"]. "<br> ";
18 echo "Password: ".$_POST["password"]. "<br> ";
19 echo "Konfirmasi Password: ".$_POST["repassword"]. "<br> ";
20 echo "Target1: ".$_POST["target1"]. "<br> ";
21 echo "Target2: ".$_POST["target2"]. "<br> ";
22 echo "Target3: ".$_POST["target3"]
23 ?>
24 </body>
25 </html>

```



Gambar: Hasil pemrosesan form

Silahkan anda samakan artibut **name** dari objek form dengan key dari variabel **\$\_POST**. Keduanya harus sama persis agar isi form dapat ditampilkan.

Khusus untuk objek form **select**, **checkbox** dan **radio button**, nilai yang dikirim adalah nilai

yang terdapat di dalam atribut **value**. Sebagai contoh, berikut adalah kode HTML untuk membuat *checkbox*:

```
<input type="checkbox" name="target1" value="HTML"> Menguasai HTML
```

Jika *checkbox* tersebut di klik (di checklist), hasil dari variabel `$_POST["target1"]` adalah `"HTML"`, bukan `"Menguasai HTML"`. Hal yang sama juga berlaku untuk *radio button* dan *select*.

Saya sangat sarankan anda mengutak-atik kode diatas, misalnya menambah beberapa object form lain, atau membuat form yang sepenuhnya baru.

-  Khusus untuk objek form *radio button* dan *checkbox*, jika tidak dipilih akan menampilkan **Notice: Undefined index**. Pesan error ini terjadi karena ketika *radio button* dan *checkbox* tidak dipilih, variabel `$_POST`-nya juga tidak akan dikirim (sehingga tidak terdefenisi). Solusi untuk masalah ini akan kita bahas sesaat lagi.

## 20.4 Pengertian Validasi Form

**Validasi form** adalah proses untuk memastikan apakah isian form sudah sesuai dengan keinginan atau belum. Sebagai contoh, apakah isian alamat *email* memang berupa email atau tidak, apakah isian jumlah barang diisi dengan angka atau malah dengan huruf.

Secara umum, terdapat beberapa kriteria yang ingin kita pastikan pada sebuah form:

- **Wajib diisi:** apakah sebuah isian form wajib diisi atau tidak.
- **Panjang karakter:** apakah panjang isian form harus dibatasi, misalnya maksimal diisi 5 karakter, atau minimal 5 karakter.
- **Tipe data:** isian form bisa ditentukan apakah berupa angka bulat (integer), angka desimal (float) atau harus berupa sebuah kalimat (string).
- **Dalam sebuah kelompok data:** isian form hanya bisa diisi dengan karakter tentu saja, misalnya "L" untuk laki-laki dan "P" untuk perempuan. Selain karakter ini, dianggap tidak valid.
- **Pola khusus:** isian form harus memenuhi syarat pola / format khusus, misalnya untuk alamat email, harus memiliki karakter "@" dan memiliki alamat domain.

Selain kelima validasi diatas, masih banyak syarat validasi lain tergantung dengan kebutuhan kita. Validasi form *sangat amat penting* untuk menghindari data yang salah. Mengisi data yang tidak seharusnya juga akan menyebabkan error. Misalnya apakah boleh isian tanggal lahir diisi dengan tahun 2020, atau apakah boleh kolom usia diisi dengan 200 tahun, dsb.

Dalam merancang form, sedapat mungkin kita sudah bisa memikirkan hal-hal seperti ini. Tidak semua pengguna form mengisi data sesuai dengan instruksi yang diberikan. Beberapa iseng mengisi data-data yang tidak benar, atau bahkan mencoba membobol isian form. Sepanjang sisa bab ini, saya akan menjelaskan beberapa teknik yang bisa digunakan untuk proses validasi form ini.

## 20.5 Function isset()

Materi pertama yang akan saya bahas terkait validasi form adalah fungsi `isset()`. Fungsi `isset()` digunakan untuk memeriksa apakah sebuah variabel telah terdefenisi atau tidak. Fungsi ini berguna untuk memastikan apakah sebuah isian form siap untuk diproses.

Mari kita lihat contoh kasusnya:

index.php

---

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar PHP</title>
6 </head>
7 <body>
8 <h1>Halaman Form</h1>
9   <form action="proses.php" method="post">
10    <p>Nama: <input type="text" name="nama" ></p>
11    <p>Email: <input type="text" name="email"></p>
12    <p><label><input type="checkbox" name="belajar" value="php">
13      Belajar PHP</p></label>
14    <input type="submit" value="Proses Data" >
15   </form>
16 </body>
17 </html>
```

---

Dari kode HTML diatas, dapatkah anda merancang isi dari halaman *proses.php*? Kita hanya perlu mengakses 3 variabel:

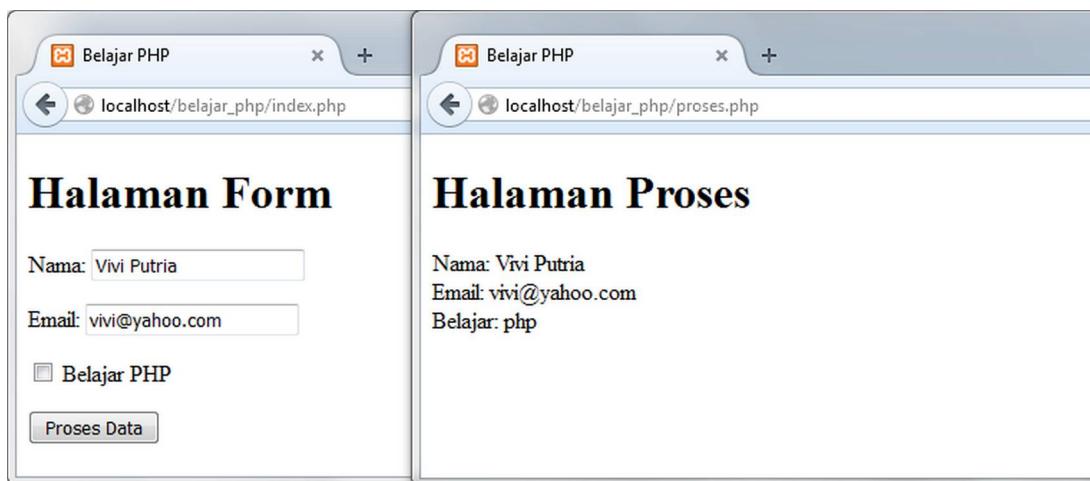
```
1 $_POST[ "nama" ]
2 $_POST[ "email" ]
3 $_POST[ "belajar" ]
```

Berikut kode lengkap untuk halaman *proses.php*:

**proses.php**

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6  </head>
7  <body>
8  <h1>Halaman Proses</h1>
9  <?php
10     echo "Nama: ".$_POST["nama"]."<br>";
11     echo "Email: ".$_POST["email"]."<br>";
12     echo "Belajar: ".$_POST["belajar"]."<br>";
13 ?>
14 </body>
15 </html>
```

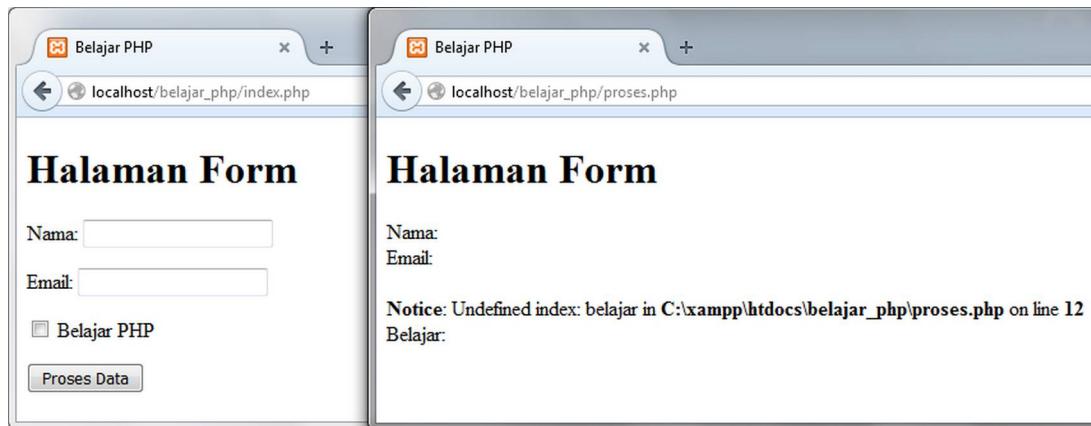
Sekarang, mari kita coba. Isi semua form, lalu **submit**. Hasilkan akan tampak seperti gambar berikut:



Gambar: Hasil form dengan mengisi seluruh isian

Semuanya tampak normal dan sesuai dengan yang dirancang. Tidak ada masalah.

Buka kembali ke halaman *index.php*, sekarang pastikan untuk TIDAK mengisi form (hapus kalau sudah tersisi), lalu langsung klik tombol “**Proses Data**”:



Gambar: Error karena tidak mengisi form

Kenapa terdapat error? Dan apabila anda perhatikan kode errornya, di baris ke 12 adalah baris kode untuk menampilkan `$_POST["belajar"]` yang berasal dari *checkbox*. Mengapa hanya ini yang error? Bukankah variabel `$_POST["nama"]` dan `$_POST["email"]` juga tidak kita isi?

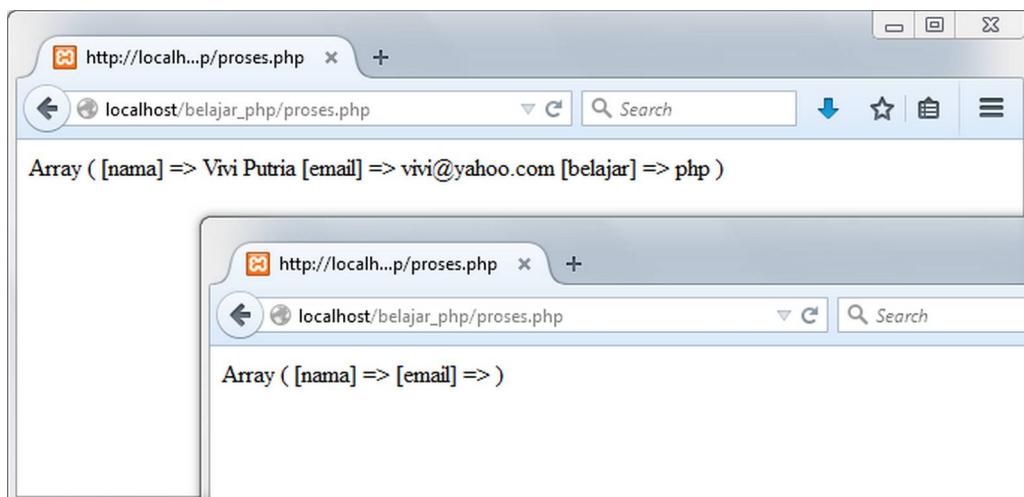
Untuk mencari letak masalahnya, saya akan lakukan proses *debugging*. Langkah pertama adalah mencari tahu apa yang sebenarnya dikirim dari halaman *index.php*. Caranya bisa dengan fungsi `print_r()` seperti yang telah kita lakukan di awal bab ini. Silahkan ubah kode program untuk halaman *proses.php* menjadi sebagai berikut:

```
1 <?php
2   print_r($_POST);
3 ?>
```

Jika butuh informasi yang lebih detail, bisa juga menggunakan fungsi `var_dump()`:

```
1 <?php
2   var_dump($_POST);
3 ?>
```

Kali ini saya akan menggunakan fungsi `print_r()` dan melakukan 2 percobaan: mengisi seluruh form, dan tidak mengisi form sama sekali. Berikut hasil yang saya dapat:



Gambar: Hasil dari mengisi seluruh form (atas), dan tanpa mengisi form (bawah)

Tampilan paling atas adalah ketika seluruh form diisi, sedangkan tampilan dibawah ketika saya langsung men-klik tombol **submit** tanpa mengisi form. Bisakah anda melihat perbedaannya?

Untuk isian form yang berupa *textbox*, yakni `<input type="text">`, variabel `$_POST` tetap akan dikirim, walaupun isinya kosong. Namun untuk isian form *checkbox*, variabel `$_POST`-nya tidak akan dikirim jika *checkbox* tersebut tidak dipilih. Inilah yang menyebabkan terjadi error **Notice: Undefined index**.

Dengan kata lain, ketika *checkbox* tidak dipilih, di halaman *proses.php* variabel `$_POST["belajar"]` tidak akan ada atau tidak terdefinisi. Pemanggilan variabel yang tidak terdefinisi inilah yang menyebabkan error.

Ketika anda menemukan masalah dengan menampilkan form, trik seperti ini sangat berguna. Kita bisa memastikan apakah isi form memang sudah dikirim atau tidak.

Jadi, bagaimana solusinya?

Sebelum melakukan sesuatu kepada `$_POST["belajar"]` kita bisa mengecek keberadaannya dengan fungsi `isset()`. Fungsi `isset()` akan mengembalikan nilai boolean `TRUE` jika variabel itu ada, dan `FALSE` jika variabel tersebut tidak ada.

Mari kita revisi kembali kode program untuk halaman *proses.php*:

### proses.php

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6  </head>
7  <body>
8  <h1>Halaman Proses</h1>
9  <?php
10     echo "Nama: " . $_POST["nama"] . "<br>";
```

```
11 echo "Email: ".$_POST["email"]."<br>";  
12 if (isset($_POST["belajar"])) {  
13     echo "Belajar: ".$_POST["belajar"]."<br>";  
14 }  
15 ?>  
16 </body>  
17 </html>
```

---

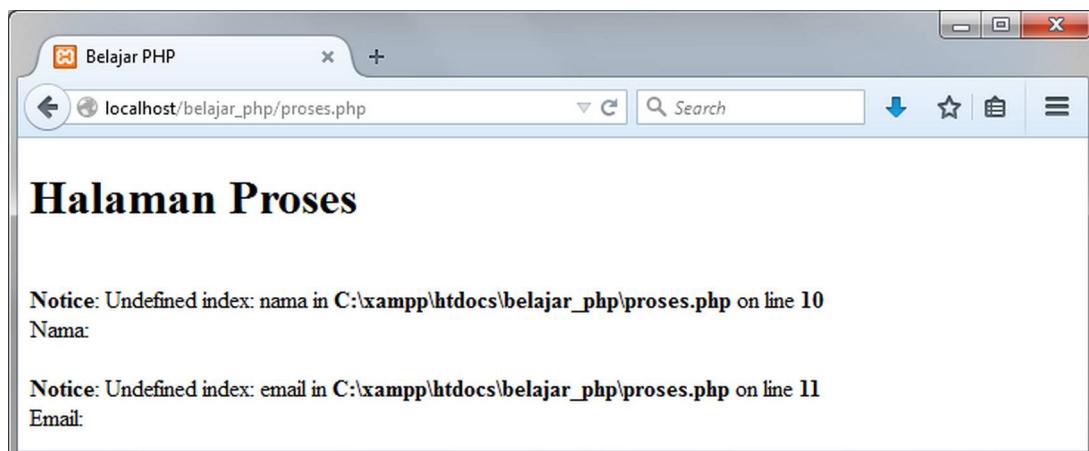
Dengan kode program diatas, variabel `$_POST["belajar"]` akan diperiksa terlebih dahulu. Jika ada, tampilkan dengan perintah `echo`. Jika tidak ada, perintah `echo` tidak akan dijalankan.

Selain *checkbox*, nilai dari *radio button* juga tidak akan dikirim jika tidak dipilih. Oleh karena itu, apabila di dalam form terdapat *radio button*, sebaiknya juga di “bentengi” menggunakan fungsi `isset()`.

## 20.6 Mencegah Akses ke Halaman proses.php

Dari contoh yang sudah kita lakukan sejak awal bab ini, saya menggunakan 2 buah halaman, yakni *index.php* dan *proses.php*. Halaman *index.php* berisi form HTML, sedangkan halaman *proses.php* berfungsi untuk memproses form dan menampilkan hasilnya. Ini juga berarti bahwa halaman *proses.php* diakses secara tidak langsung hanya ketika form di-submit.

Namun tidak ada yang bisa menghalangi seseorang mengakses halaman *proses.php* tanpa dari form. Menggunakan contoh kode sebelumnya, inilah hasil yang saya dapat ketika mengakses halaman *proses.php* secara langsung:



Gambar: Error ‘Notice: Undefined index’ ketika halaman proses.php diakses langsung

Sama seperti pada kasus *checkbox*, error ini terjadi karena di dalam *proses.php* saya mengakses variabel `$_POST["nama"]` dan `$_POST["email"]` secara langsung. Saat halaman ini diakses tanpa melalui *form*, kedua variabel ini memang tidak ada. Jadi, bagaimana menanggulanginya? Salah satunya bisa seperti ini:

**proses.php**

---

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6  </head>
7  <body>
8  <h1>Halaman Proses</h1>
9  <?php
10     if (isset($_POST["nama"])) {
11         echo "Nama: ".$_POST["nama"]."<br>";
12     }
13     if (isset($_POST["email"])) {
14         echo "Email: ".$_POST["email"]."<br>";
15     }
16     if (isset($_POST["belajar"])) {
17         echo "Belajar: ".$_POST["belajar"]."<br>";
18     }
19 ?>
20 </body>
21 </html>
```

---

Dengan kode diatas, setiap variabel `$_POST` akan diperiksa terlebih dahulu apakah ada atau tidak. Jika variabel tersebut ditemukan, baru kita mulai memproses isian form.

Apabila objek form yang dikirim cukup banyak, memeriksa satu-satu variabel ini terasa tidak praktis (walaupun tidak salah). Sebagai alternatif, terdapat satu metode yang bisa kita gunakan. Kembali ke halaman `index.php`, tambahkan atribut `name` pada tombol submit:

```
<input type="submit" name="submit" value="Proses Data" >
```

Ini artinya ketika form disubmit, `$_POST["submit"]` juga ikut dikirim. Variabel inilah yang bisa dijadikan “bukti” bahwa form telah dikirim ke halaman `proses.php`. Berikut kode lengkap untuk halaman `index.php`:

## index.php

---

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar PHP</title>
6 </head>
7 <body>
8 <h1>Halaman Form</h1>
9   <form action="proses.php" method="post">
10    <p>Nama: <input type="text" name="nama" ></p>
11    <p>Email: <input type="text" name="email"></p>
12    <p><label><input type="checkbox" name="belajar" value="php">
13      Belajar PHP</p></label>
14    <input type="submit" value="Proses Data" name="submit">
15  </form>
16 </body>
17 </html>
```

---

Ketika menggunakan fungsi `print_r($_POST)` dari halaman `proses.php`, berikut hasil yang saya dapat:

```
1 Array
2 (
3   [nama] => Vivi Putria
4   [email] => vivi@yahoo.com
5   [belajar] => php
6   [submit] => Proses Data
7 )
```

Selain isian form **nama**, **email** dan **belajar**, ikut juga dikirim “**submit**” dengan nilai “*proses data*”. Dengan demikian, di halaman `proses.php` saya bisa menulis kode seperti ini:

## proses.php

---

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar PHP</title>
6 </head>
7 <body>
8 <h1>Halaman Proses</h1>
9 <?php
```

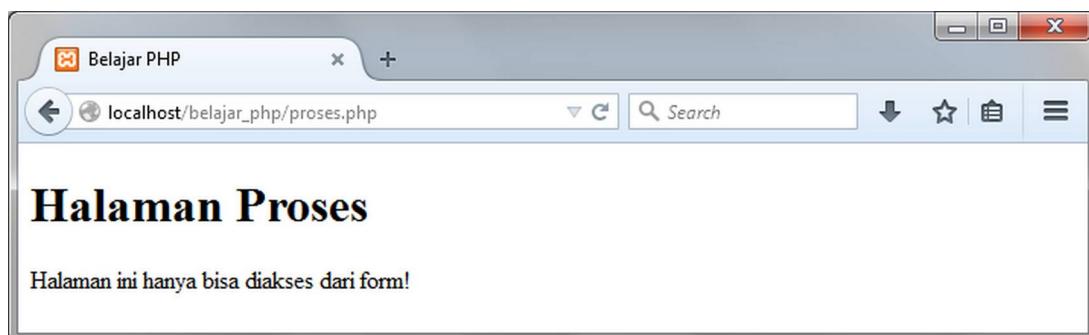
```

10 if (isset($_POST["submit"])) {
11     echo "Nama: ".$_POST["nama"]."<br>";
12     echo "Email: ".$_POST["email"]."<br>";
13     if (isset($_POST["belajar"])) {
14         echo "Belajar: ".$_POST["belajar"]."<br>";
15     }
16 }
17 else {
18     echo "Halaman ini hanya bisa diakses dari form!";
19 }
20 ?>
21 </body>
22 </html>

```

---

Kali ini saya memeriksa apakah variabel `$_POST["submit"]` berisi sebuah nilai atau tidak. Jika iya, ini artinya form telah di-submit dan kita bisa memproses isian form. Jika tidak ada, berarti form belum dikirim dan kemungkinan besar halaman `proses.php` diakses secara langsung. Apabila ini yang terjadi, bagian `else` akan jalan dan menampilkan *“Halaman ini hanya bisa diakses dari form!”*.



Gambar: Tampilan halaman `proses.php` ketika diakses secara langsung

Daripada menampilkan keterangan error seperti ini, kita juga bisa me-*redirect* pengunjung ke halaman lain, misalnya langsung ke `index.php`:

#### proses.php

---

```

1 <?php
2 if (!isset($_POST["submit"])) {
3     header("Location: index.php");
4     die();
5 }
6 ?>
7 <!DOCTYPE html>
8 <html>
9 <head>
10    <meta charset="UTF-8">
11    <title>Belajar PHP</title>

```

```

12  </head>
13  <body>
14  <h1>Halaman Proses</h1>
15  <?php
16  echo "Nama: ".$_POST["nama"]."<br>";
17  echo "Email: ".$_POST["email"]."<br>";
18
19  if (isset($_POST["belajar"])) {
20      echo "Belajar: ".$_POST["belajar"]."<br>";
21  }
22 ?>
23 </body>
24 </html>

```

---

Sekarang, logika program sedikit berubah. Jika halaman *proses.php* tidak menemukan variabel `$_POST["submit"]`, maka akan langsung di *redirect* ke *index.php* (tempat dimana form berada). Perhatikan cara penulisan fungsi `isset()`, saya menulis operator negasi “!” untuk pemberbalik logika agar alur if berjalan saat variabel `$_POST["submit"]` tidak ditemukan.

Kode program ini juga harus ditulisi di bagian paling awal, karena fungsi `header()` harus dijalankan sebelum output dikirim dari web server. Penjelasan lengkapnya telah kita bahas pada bab tentang **HTTP header**.

## 20.7 Function empty()

Validasi berikutnya yang akan saya bahas adalah memeriksa apakah sebuah objek form sudah diisi atau tidak. Untuk kebutuhan ini, kita bisa menggunakan fungsi `empty()`.

Fungsi `empty()` akan menghasilkan nilai TRUE ketika sebuah variabel tidak berisi apa-apa. Namun karena PHP mengkonversi sebuah tipe data menjadi tipe data lain, “tidak berisi apa-apa” ini mencakup beberapa kondisi, berikut isi variabel yang dianggap “kosong” oleh fungsi `empty()`:

```

1  <?php
2      // sebuah string kosong
3      $test="";
4      var_dump(empty($test)); echo "<br>";    // bool(true)
5
6      // string kosong
7      $test="";
8      var_dump(empty($test)); echo "<br>";    // bool(true)
9
10     // integer 0
11     $test=0;
12     var_dump(empty($test)); echo "<br>";    // bool(true)

```

```
13
14 // float 0.0
15 $test="";
16 var_dump(empty($test)); echo "<br>"; // bool(true)
17
18 // string "0"
19 $test="0";
20 var_dump(empty($test)); echo "<br>"; // bool(true)
21
22 // NULL
23 $test=NULL;
24 var_dump(empty($test)); echo "<br>"; // bool(true)
25
26 // boolean FALSE
27 $test="";
28 var_dump(empty($test)); echo "<br>"; // bool(true)
29
30 // array kosong
31 $test=array();
32 var_dump(empty($test)); echo "<br>"; // bool(true)
33
34 // variabel yang belum berisi nilai
35 $test;
36 var_dump(empty($test)); // bool(true)
37 ?>
```

Sebagai contoh praktik, saya akan menggunakan form yang sama seperti sebelumnya. Kali ini saya ingin agar isian *nama* dan *email* tidak boleh kosong (harus diisi). Kode form untuk halaman *index.php* tidak perlu kita ubah, halaman *proses.php*-lah yang harus dimodifikasi. Berikut perubahannya:

#### proses.php

---

```
1 <?php
2 if (!isset($_POST["submit"])) {
3     header("Location: index.php");
4 }
5 ?>
6 <!DOCTYPE html>
7 <html>
8 <head>
9     <meta charset="UTF-8">
10    <title>Belajar PHP</title>
11 </head>
12 <body>
13 <h1>Halaman Proses</h1>
14 <?php
```

```
15 $nama=$_POST["nama"];
16 $email=$_POST["email"];
17
18 if (empty($nama)) {
19     echo "Nama wajib diisi <br>";
20 }
21 else {
22     echo "Nama: $nama <br>";
23 }
24
25 if (empty($email)) {
26     echo "Email wajib diisi <br>";
27 }
28 else {
29     echo "Email: $email <br>";
30 }
31
32 if (isset($_POST["belajar"])) {
33     echo "Belajar: ".$_POST["belajar"]."<br>";
34 }
35 ?>
36 </body>
37 </html>
```

Sekarang apabila objek form *nama* dan *email* tidak diisi, akan tampil pesan “*Nama wajib diisi*” atau “*Email wajib diisi*”. Silahkan anda coba untuk mengosongkan salah satu (atau keduanya) dan lihat pesan error yang tampil.

Selain itu, dalam kode diatas saya juga memindahkan isi variabel `$_POST["nama"]` dan `$_POST["email"]` ke dalam variabel `$nama` dan `$email` agar lebih mudah diakses. Cara seperti ini akan saya gunakan dalam contoh-contoh selanjutnya.



Gambar: Pesan kesalahan bahwa nama dan email harus diisi

Saya yakin anda akan langsung berfikir, lebih baik pesan ini tampil di atas form sehingga pengguna bisa langsung memperbaiknya dengan mengisi kolom *nama* dan *email*. Saya akan

membahas cara membuatnya beberapa saat lagi. Karena masih ada sedikit “problem” dengan fungsi `empty()` yang kita gunakan.

Sekilas, tidak ada yang salah dari kode sebelumnya, namun bagaimana jika seseorang menginput ‘spasi’ atau ‘tab’ ke dalam kolom *nama* dan *email*. Apakah yang terjadi? Silahkan anda coba sebentar.

Yup, ‘spasi’ dan ‘tab’ (serta karakter *whitespace* lain) dianggap sebagai sebuah karakter oleh fungsi `empty()`, sehingga dianggap ‘berisi’. Bagaimana solusinya? Gunakan fungsi `trim()`.

Jika anda masih ingat, fungsi `trim()` berguna untuk menghapus karakter *whitespace* di sisi kiri dan kanan sebuah string. Ini bisa kita manfaatkan untuk men-filter isian form sebelum di periksa dengan fungsi `empty()`. Berikut perubahan kode programnya:

```
1 <?php
2     //... kode halaman proses.php disini
3     $nama=trim($_POST["nama"]);
4     $email=trim($_POST["email"]);
5     //... kode halaman proses.php disini
6 ?>
```

Dengan melewatkannya variabel `$_POST` kedalam fungsi `trim()`, karakter *whitespace* akan dihapus terlebih dahulu. Silahkan anda coba lagi dengan menginput spasi atau tab ke isian *nama* dan *email*.

## 20.8 Menampilkan Pesan Kesalahan Form

Pesan kesalahan form akan lebih rapi jika ditampilkan di atas form pada halaman *index.php*. Sehingga pengguna bisa langsung memperbaiki form dengan lebih cepat tanpa perlu men-klik tombol *back* di web browser.

Terdapat berbagai teknik yang bisa digunakan. Kali ini saya akan me-*redirect* pesan error ke halaman *index.php* melalui *query string*.

Silahkan pelajari sebentar kode untuk halaman *proses.php* berikut ini:

*proses.php*

---

```
1 <?php
2     if (!isset($_POST["submit"])) {
3         header("Location: index.php");
4     }
5
6     // ambil nilai form
7     $nama=trim($_POST["nama"]);
8     $email=trim($_POST["email"]);
9
10    // cek apakah "nama" sudah diisi atau tidak
```

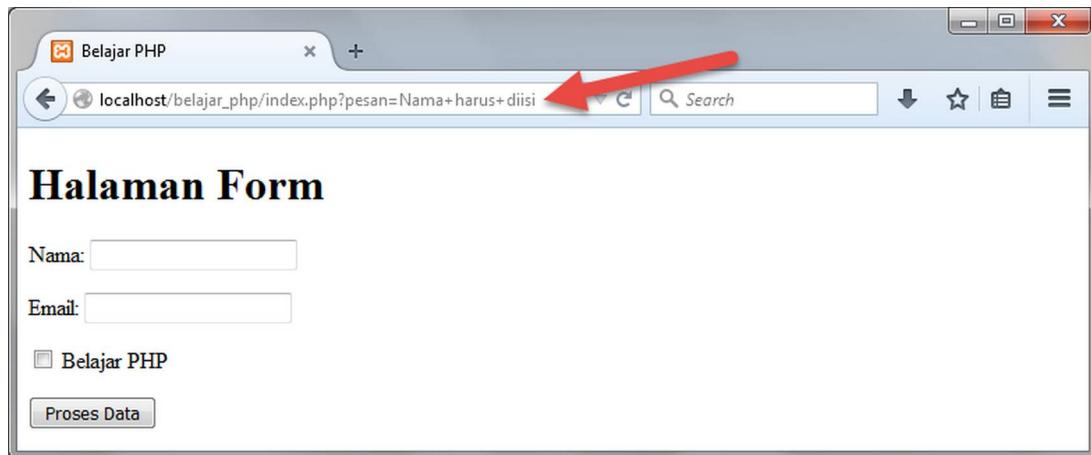
```
11  if (empty($nama)) {
12      $pesan = urlencode("Nama harus diisi");
13      header("Location: index.php?pesan=$pesan");
14      die();
15  }
16
17  // cek apakah "email" sudah diisi atau tidak
18  if (empty($email)) {
19      $pesan = urlencode("Email harus diisi");
20      header("Location: index.php?pesan=$pesan");
21      die();
22  }
23  ?>
24  <!DOCTYPE html>
25  <html>
26  <head>
27      <meta charset="UTF-8">
28      <title>Belajar PHP</title>
29  </head>
30  <body>
31  <h1>Halaman Proses</h1>
32  <?php
33      echo "Nama: $nama <br>";
34      echo "Email: $email <br>";
35      if (isset($_POST["belajar"])) {
36          echo "Belajar: ".$_POST["belajar"]."<br>";
37      }
38  ?>
39  </body>
40  </html>
```

---

Saya memindahkan pemrosesan form ke bagian atas karena akan menggunakan fungsi **header()**. Apabila anda telah memahami bab tentang **HTTP header**, saya yakin sudah bisa memahami apa yang di kerjakan kode program ini.

Ketika isian *nama* atau *email* kosong, saya menyiapkan pesan error di dalam variabel **\$pesan**. Pesan error ini lalu di-*redirect* ke halaman *index.php* (tempat dimana form berada).

Jika anda menjalankan halaman *index.php* dengan mengosongkan salah satu kotak form, akan langsung kembali ke halaman *index.php*, namun tidak tampil pesan apa-apa. Kenapa? Karena saya belum membuat kode program untuk “mengambil” variabel **\$pesan** dari *query string*.



Gambar: Pesan error dikirim melalui query string

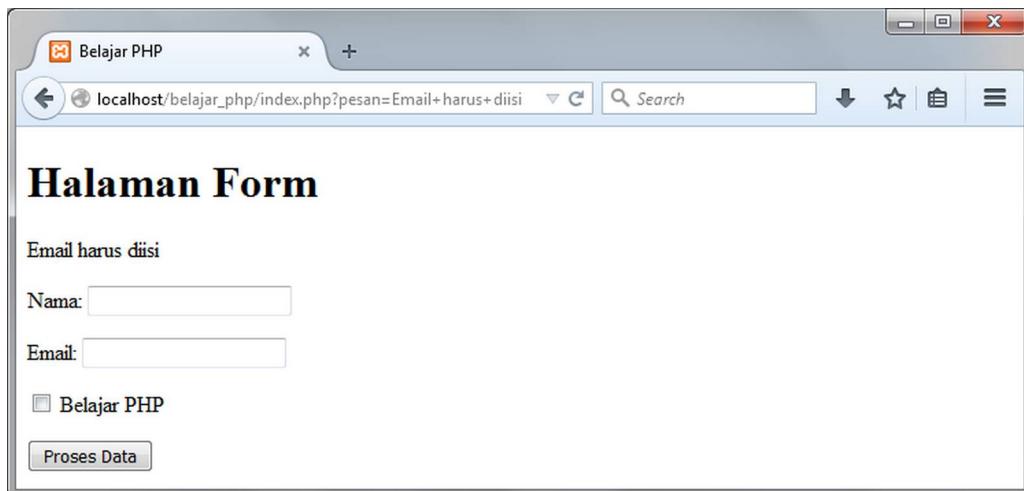
Untuk dapat menampilkan pesan error ini, kita bisa menggunakan cara yang sama seperti bab sebelumnya, yakni melalui variabel `$_GET["pesan"]`. Berikut perubahan untuk halaman `index.php`:

#### index.php

```
1 <?php
2     if (isset($_GET["pesan"])){
3         $pesan = "<p> {$_GET["pesan"]} <p>";
4     }
5     else {
6         $pesan = "";
7     }
8 ?>
9 <!DOCTYPE html>
10 <html>
11 <head>
12     <meta charset="UTF-8">
13     <title>Belajar PHP</title>
14 </head>
15 <body>
16     <h1>Halaman Form</h1>
17     <?php echo $pesan ?>
18     <form action="proses.php" method="post">
19         <p>Nama: <input type="text" name="nama" ></p>
20         <p>Email: <input type="text" name="email"></p>
21         <p><label><input type="checkbox" name="belajar" value="php">
22             Belajar PHP</p></label>
23         <input type="submit" value="Proses Data" name="submit">
24     </form>
25 </body>
26 </html>
```

Pada bagian awal *index.php*, saya memeriksa apakah variabel `$_GET["pesan"]` terdefinisi atau tidak. Jika iya, pindahkan isinya ke dalam variabel `$pesan`. Jika tidak, variabel `$pesan` diisi dengan string kosong (ini juga yang terjadi ketika halaman *index.php* dijalankan pertama kali). Variabel `$pesan` selanjutnya ditampilkan sebelum form.

Sekarang, jika isian form *nama* atau *email* kosong, akan tampil pesan error.



Gambar: Pesan error yang tampil ketika form tidak diisi

## 20.9 Mengisi Kembali Form (form re-populate)

Selain menampilkan pesan error, sebuah form akan lebih *user friendly* jika terisi kembali secara otomatis.

Sebagai contoh, misalkan saya telah menulis *nama*, namun lupa mengisi kolom *email*. Pesan error akan tampil untuk memberitahukan bahwa kolom *email* wajib diisi, namun kolom *nama* juga kembali kosong. Ini terjadi karena efek *redirect* halaman akan menghapus seluruh isian form. Akan jauh lebih baik jika kolom *nama* langsung terisi dengan nilai yang saya input sebelumnya.

Untuk membuat hal ini, kita bisa menggunakan beberapa teknik, misalnya dengan **cookie** (yang akan kita pelajari pada bab berikutnya), atau bisa juga dengan cara menyisipkan hasil form ke dalam *query string*. Idenya sama seperti cara menampilkan pesan error yang telah kita pelajari. Mari kita coba, berikut kode program untuk halaman *proses.php*:

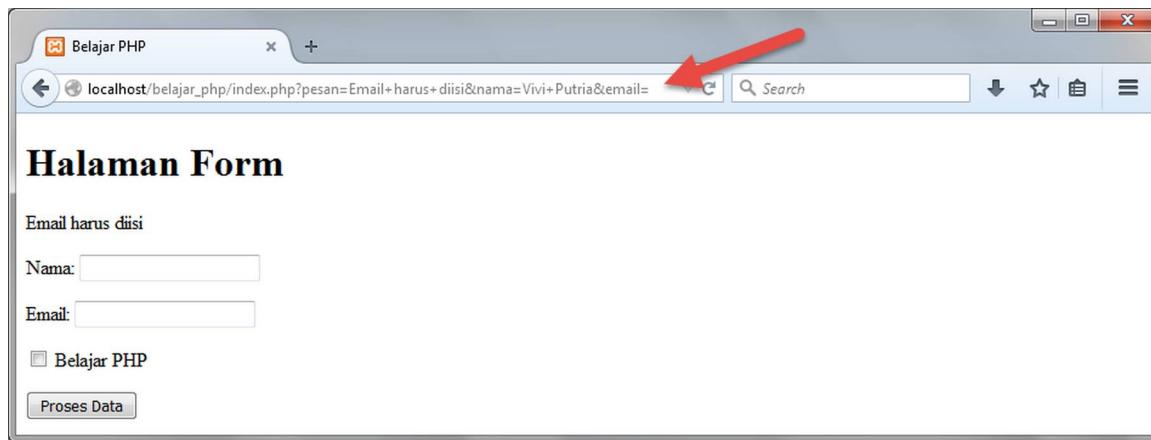
## proses.php

```
1 <?php
2     if (!isset($_POST["submit"])) {
3         header("Location: index.php");
4     }
5
6     // ambil nilai form
7     $nama=trim($_POST["nama"]);
8     $email=trim($_POST["email"]);
9
10    // siapkan nilai untuk dikirim kembali
11    $query_nama = "nama=".urlencode($nama);
12    $query_email = "email=".urlencode($email);
13    $isi_form = "&$query_nama&$query_email";
14
15    // cek apakah "nama" sudah diisi atau tidak
16    if (empty($nama)) {
17        $pesan = urlencode("Nama harus diisi");
18        header("Location: index.php?pesan={$pesan}{$isi_form}");
19        die();
20    }
21
22    // cek apakah "email" sudah diisi atau tidak
23    if (empty($email)) {
24        $pesan = urlencode("Email harus diisi");
25        header("Location: index.php?pesan={$pesan}{$isi_form}");
26        die();
27    }
28 ?>
29 <!DOCTYPE html>
30 <html>
31 <head>
32     <meta charset="UTF-8">
33     <title>Belajar PHP</title>
34 </head>
35 <body>
36 <h1>Halaman Proses</h1>
37 <?php
38     echo "Nama: $nama <br>";
39     echo "Email: $email <br>";
40     if (isset($_POST["belajar"])) {
41         echo "Belajar: ".$_POST["belajar"]."<br>";
42     }
43 ?>
44 </body>
```

45 </html>

---

Sebelum di *redirect* ke halaman *index.php*, saya mempersiapkan variabel **\$query\_nama** dan **\$query\_email** untuk disambung ke variabel **\$isi\_form**. Variabel ini kemudian dikirim bersama **\$pesan** ke halaman *index.php* melalui *query string*.



Gambar: Query string berisi pesan error, variabel nama dan email

Dapat terlihat bahwa selain pesan error, di dalam *query string* juga terdapat nilai *nama* dan *email*. Langkah selanjutnya adalah mengubah kode di halaman *index.php* agar bisa mengambil nilai ini. Berikut modifikasinya:

#### index.php

```
1 <?php
2     // ambil pesan jika ada
3     if (isset($_GET["pesan"])){
4         $pesan = "<p> {$_GET["pesan"]} <p>";
5     }
6     else {
7         $pesan = "";
8     }
9
10    // ambil nilai nama jika ada
11    if (isset($_GET["nama"])){
12        $nilai_nama = $_GET["nama"];
13    }
14    else {
15        $nilai_nama = "";
16    }
17
18    // ambil nilai email jika ada
19    if (isset($_GET["email"])){
20        $nilai_email = $_GET["email"];
21    }
```

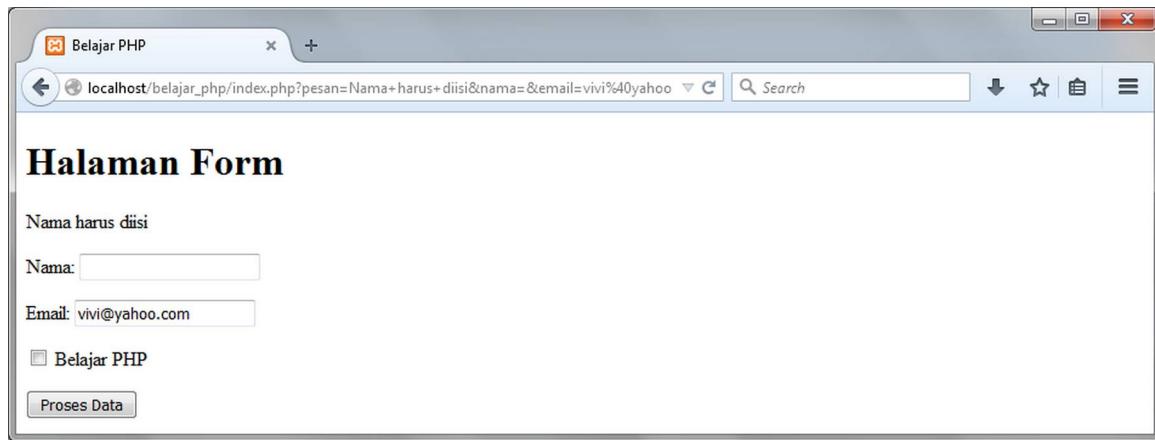
```
22  else {
23      $nilai_email = "";
24  }
25 ?>
26 <!DOCTYPE html>
27 <html>
28 <head>
29     <meta charset="UTF-8">
30     <title>Belajar PHP</title>
31 </head>
32 <body>
33 <h1>Halaman Form</h1>
34 <?php echo $pesan ?>
35     <form action="proses.php" method="post">
36         <p>Nama: <input type="text" name="nama"
37             value="<?php echo $nilai_nama ?>" ></p>
38         <p>Email: <input type="text" name="email"
39             value="<?php echo $nilai_email ?>" ></p>
40         <p><label><input type="checkbox" name="belajar" value="php">
41             Belajar PHP</p></label>
42         <input type="submit" value="Proses Data" name="submit">
43     </form>
44 </body>
45 </html>
```

---

Pada prinsipnya, untuk form re-*populate* sama seperti cara menampilkan pesan error. Saya mengambil nilai *nama* dan *email* dari query string menggunakan variabel `$_GET["nama"]` dan `$_GET["email"]`.

Agar keduanya ditampilkan di dalam form, variabel ini saya tempatkan ke dalam atribut `value`. Sebagaimana yang kita pelajari di HTML, atribut `value` untuk tag `<input>` berfungsi sebagai nilai awal form:

```
<p>Nama: <input type="text" name="nama"
    value="<?php echo $nilai_nama ?>" ></p>
<p>Email: <input type="text" name="email"
    value="<?php echo $nilai_email ?>" ></p>
```



Gambar: Isi form akan ditampilkan kembali ketika terdapat error

## 20.10 Memproses Form Dalam Halaman Yang Sama

Sampai saat ini, saya memproses form menggunakan 2 buah halaman, yakni *index.php* (tempat form berada), dan halaman *proses.php* (tempat proses dilakukan). Pembagian halaman ini membuat kode PHP lebih terstruktur dan lebih rapi.

Namun seperti yang telah kita praktikkan sebelumnya, "sarana komunikasi" antara halaman *proses.php* dengan *index.php* harus melalui *query string*. Untuk form yang panjang, *query string* juga memiliki batasan, sehingga bisa menjadi masalah tersendiri.

Solusinya, kita bisa membuat kode program dimana tampilan form dan pemrosesan form dilakukan pada halaman yang sama. Triknya adalah dengan mengisi atribut **action** di tag `<form>` dengan halaman itu sendiri. Sebagai contoh, pada halaman *index.php*, saya bisa membuat tag `<form>` seperti berikut:

```
<form action="index.php" method="post">
```

Ini artinya, ketika form disubmit, data form akan dikirim kembali ke halaman *index.php*.

Namun, kita perlu suatu cara untuk memisahkan kapan proses form di lakukan, dan kapan halaman *index.php* tampil pertama kali. Teknik ini sebenarnya sudah kita pelajari, yakni dengan memeriksa kondisi dari variabel `$_POST["submit"]`. Jika variabel ini ada, jalankan pemrosesan form. Jika tidak ada, tampilkan form.

Berikut cara memproses form pada halaman yang sama, yakni *index.php*:

## index.php

---

```
1 <?php
2 // cek apakah form telah di submit
3 if (isset($_POST["submit"])) {
4
5     // form telah disubmit, proses data
6     echo "<h1>Form Berhasil di Proses </h1>";
7     echo "Nama : {" . $_POST["nama"] . "}<br>";
8     echo "Email : {" . $_POST["email"] . "}";
9     die();
10 }
11 ?>
12 <!DOCTYPE html>
13 <html>
14 <head>
15     <meta charset="UTF-8">
16     <title>Belajar PHP</title>
17 </head>
18 <body>
19 <h1>Form Register</h1>
20     <form action="index.php" method="post">
21         <p>Nama: <input type="text" name="nama"></p>
22         <p>Email: <input type="text" name="email"></p>
23         <input type="submit" name="submit" value="Proses Data" >
24     </form>
25 </body>
26 </html>
```

---

Ketika kode program `isset($_POST["submit"])` menghasilkan TRUE, ini artinya form sudah di-*submit*, maka tampilkan nilai `$_POST["nama"]` dan `$_POST["email"]`. Untuk mengakhiri proses, saya menggunakan fungsi `die()`.

Namun jika variabel `$_POST["submit"]` tidak ada, kode program untuk pemrosesan form akan dilewatkan dan form HTML ditampilkan seperti kondisi awal.

Dalam kode diatas saya sengaja tidak menambahkan validasi apapun agar mudah dipahami. Baik, mari kita tambahkan validasi yang telah kita pelajari sebelumnya, termasuk menampilkan pesan error:

## index.php

```
1 <?php
2 // cek apakah form telah di submit
3 if (isset($_POST["submit"])) {
4
5     // ambil nilai form
6     $nama=trim($_POST["nama"]);
7     $email=trim($_POST["email"]);
8
9     // siapkan variabel untuk menampung pesan error
10    $pesan_error="";
11
12    // cek apakah "nama" sudah diisi atau tidak
13    if (empty($nama)) {
14        $pesan_error .= "Nama belum diisi <br>";
15    }
16
17    // cek apakah "email" sudah diisi atau tidak
18    if (empty($email)) {
19        $pesan_error .= "Email belum diisi <br>";
20    }
21
22    // jika tidak ada error, tampilkan isi form
23    if ($pesan_error == "") {
24        echo "<h1>Form Berhasil di Proses </h1>";
25        echo "Nama : $nama <br>";
26        echo "Email : $email";
27        die();
28    }
29 }
30 else {
31     $pesan_error = "";
32     $nama = "";
33     $email = "";
34 }
35 ?>
36 <!DOCTYPE html>
37 <html>
38 <head>
39     <meta charset="UTF-8">
40     <title>Belajar PHP</title>
41 </head>
42 <body>
43 <h1>Form Register</h1>
44 <?php echo $pesan_error; ?>
```

```
45  <form action="index.php" method="post">
46    <p>Nama: <input type="text" name="nama"
47      value="<?php echo $nama ?>" ></p>
48    <p>Email: <input type="text" name="email"
49      value="<?php echo $email ?>" ></p>
50    <input type="submit" name="submit" value="Proses Data" >
51  </form>
52 </body>
53 </html>
```

---

Kode diatas cukup panjang, silahkan anda luangkan waktu sejenak untuk mempelajarinya.

Inilah keuntungan jika kode untuk form dan pemrosesan form ada di halaman yang sama. Saya bisa dengan mudah menampilkan kode error dan mengisi kembali nilai form (*re-populate*).

Selain itu, saya juga bisa menampilkan 2 pesan error sekaligus, yakni ketika isian *nama* dan *email* tidak diisi. Dalam contoh menggunakan *query string*, pesan error ini saya kirim satu persatu.

Khusus untuk bagian yang menampilkan hasil form, dalam praktek sebenarnya kita bisa menyimpan data ini ke database. Atau bisa juga menggunakan fungsi **include()** untuk membuat tampilan hasil form yang lebih pas (akan saya bahas dalam *case study* di akhir bab ini).

## 20.11 Proteksi Form dari Cross-Site Scripting dan HTML Injection

Salah satu konsep dasar yang harus selalu kita ingat sebagai web developer adalah: jangan percaya dengan apa yang diinput user (pengguna). Setiap "hal" yang memungkinkan seseorang untuk menginput sesuatu (seperti form) dapat menjadi celah membobol web kita atau menyisipkan kode-kode aneh.

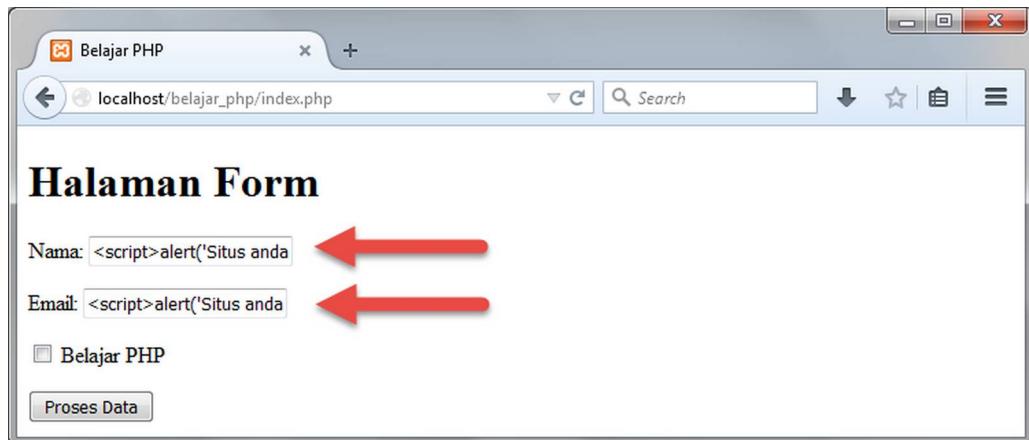
**Cross-site Scripting** atau sering disingkat dengan **XSS** adalah jenis serangan ke sebuah situs dengan cara menyisipkan kode *script* (biasanya **JavaScript**). Ini hanya akan berhasil jika situs tersebut memiliki fitur untuk menampilkan kembali isian form ke web browser, seperti yang terdapat di dalam form komentar.

Sedangkan **HTML injection** adalah istilah yang lebih spesifik kepada cara 'menyisipkan' kode HTML ke dalam sebuah situs.

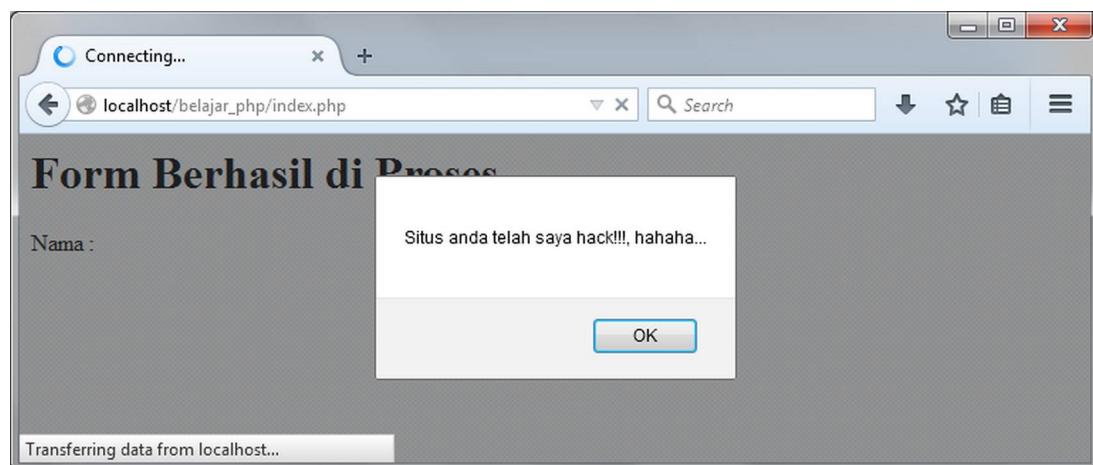
Sebagai programmer web, penanganan *Cross-site Scripting* maupun *HTML injection* merupakan hal yang sangat penting, terutama dalam pemrosesan form. Pada dasarnya, form dapat diinput oleh siapa saja, oleh karena itu kita perlu memproteksi web dari kode-kode berbahaya yang bisa diinput melalui form.

Sebagai contoh, silahkan ketik kode berikut ke dalam kolom *nama* dan *email* dari kode program kita sebelumnya:

```
<script>alert('Situs anda telah saya hack!!!, hahaha...')</script>
```



Gambar: Input script ke dalam form



Gambar: Contoh hasil dari Cross-site Scripting

Tampilan diatas adalah hasil dari kode **JavaScript** yang baru saja saya input melalui form. Kode ini dapat berjalan karena pada halaman *index.php* saya langsung menampilkan data yang diinput oleh user tanpa melakukan proses *filter* lebih lanjut.

Hal ini sangat berbahaya, karena dengan kode **JavaScript** seseorang bisa melakukan 'hampir segalanya' dengan situs kita. Tidak hanya sekedar menampilkan pesan seperti kode diatas, tetapi juga bisa mengubah background, tampilan seluruh web, bahkan mengarahkan pengunjung ke situs lain (*redirect*).



Hasil diatas saya peroleh menggunakan web browser *Mozilla Firefox*. Ketika saya menggunakan *Google Chrome*, tampilan javascript tersebut tidak muncul. Ini karena *Google Chrome* memiliki mekanisme 'pencegahan' kode javascript internal ketika diinput melalui form.

Sebagai contoh untuk **HTML injection**, seseorang bisa menulis kode berikut ke dalam kotak input nama:

```
Dunia <br> Ilkom
```

Walaupun tidak mengandung script, namun kode ini membuat tampilan hasil form menjadi berantakan (bayangkan jika kita memiliki halaman yang berisi tabel berisi nama-nama seluruh user, dan kode diatas akan menghancurkan desain web yang telah dirancang).

Untuk memproteksi dari **XXS** dan **HTML injection**, kita bisa menggunakan fungsi seperti **htmlspecialchars()**, **htmlentities()** atau **strip\_tags()**. Penjelasan serta perbedaan masing-masing fungsi ini telah kita bahas dalam bab-bab sebelumnya.

Saya akan menggunakan fungsi **htmlentities()**, dan ini bisa ditempatkan pada saat mengambil variabel **\$\_POST**:

```
$nama = htmlentities(trim($_POST["nama"]));  
$email = htmlentities(trim($_POST["email"]));
```

Atau jika anda memutuskan untuk tidak membolehkan seseorang menulis tag HTML, bisa menambahkan satu lagi fungsi **strip\_tags()**:

```
$nama = htmlentities(strip_tags(trim($_POST["nama"])));  
$email = htmlentities(strip_tags(trim($_POST["email"])));
```

Dengan fungsi ini setidaknya form kita menjadi lebih terlindungi.



Pembahasan tentang pengamanan kode PHP (*PHP Security*) merupakan materi lanjutan yang cukup kompleks. Teknik penggunaan fungsi **htmlspecialchars()**, **htmlentities()** atau **strip\_tags()** hanyalah sedikit pengaman untuk form.

Sebagai contoh lain, ada lagi metode yang disebut sebagai **Cross-Site Request Forgery** yang disingkat menjadi **CSRF** atau **XSRF**. Disini web dibobol dengan cara menjebak kode program seolah-olah di akses dari sumber yang terpercaya (untuk web yang menggunakan metode login). Materi seperti ini termasuk ke dalam bahasan PHP lanjutan, dan untuk saat ini tidak akan saya bahas.

## 20.12 Membuat Batasan untuk Validasi Form

Setelah membahas teknik untuk menampilkan pesan kesalahan dan *re-populate* form, kita bisa fokus untuk membuat validasi form lainnya. Kali ini saya akan membahas cara membatasi jumlah karakter form.

Misalkan kita ingin isian form *nama* hanya bisa berisi maksimum 5 karakter saja, atau minimum 10 karakter untuk *password*. Ini bisa dilakukan dengan memeriksa panjang variabel menggunakan fungsi **strlen()**.

Perhatikan potongan kode berikut:

```
1 // nama harus 5 digit atau lebih
2 if (strlen($nama) <= 5) {
3     $pesan_error .= "Nama minimal 5 digit <br>";
4 }
```

Kode diatas bisa saya tambahkan untuk membatasi isian *nama* yang harus diisi dengan minimal 5 digit.

Atau bisa juga sebagai berikut:

```
1 // nama maksimal 10 digit
2 if (strlen($nama) > 10) {
3     $pesan_error .= "Nama maksimal 10 digit <br>";
4 }
```

Kali ini **\$nama** harus berisi kurang dari atau sama dengan 10 digit.

Ada kalanya kita juga ingin isian form harus memiliki sebuah karakter agar bisa disebut valid. Sebagai contoh, untuk alamat email, harus ada karakter '@' di dalam string yang diinput. Bagaimana cara memeriksanya? Kita bisa memanfaatkan fungsi **strpos()**. Seperti contoh berikut:

```
1 // alamat email harus memiliki karakter '@'
2 if (strpos($email, "@") === false) {
3     $pesan_error .= "Penulisan alamat email salah <br>";
4 }
```

Jika anda masih ingat, fungsi **strpos()** akan mengembalikan posisi suatu karakter di dalam string atau FALSE jika karakter tersebut tidak ditemukan.

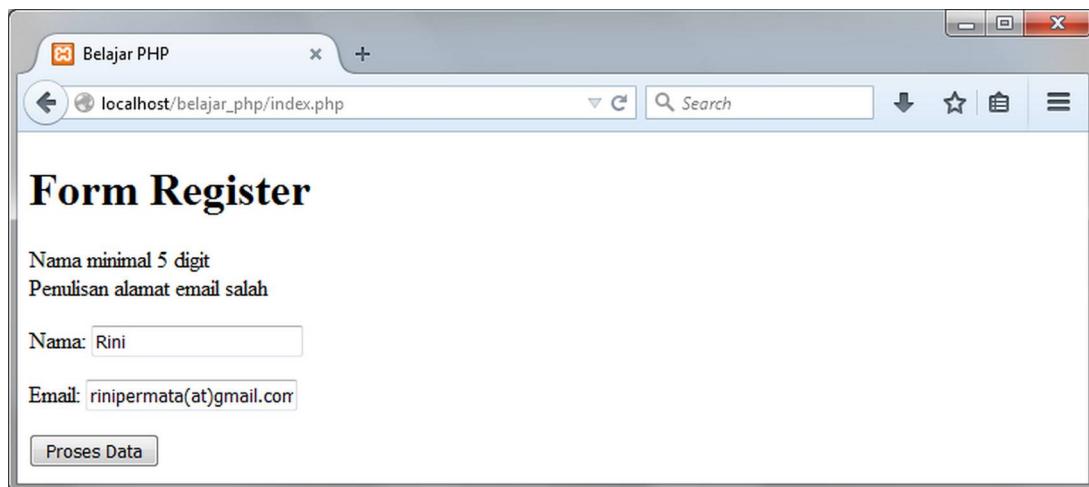
Menggabungkan kode program untuk validasi **strlen()** dan **strpos()**, berikut kode lengkap halaman *index.php*:

**index.php**

---

```
1 <?php
2 // cek apakah form telah di submit
3 if (isset($_POST["submit"])) {
4
5     // ambil nilai form
6     $nama = htmlentities(strip_tags(trim($_POST["nama"])));
7     $email = htmlentities(strip_tags(trim($_POST["email"])));
8
9     // siapkan variabel untuk menampung pesan error
10    $pesan_error="";
11
12    // cek apakah "nama" sudah diisi dan panjang karakter >= 5
13    if (empty($nama)) {
```

```
14     $pesan_error .= "Nama belum diisi <br>";
15 }
16 else if (strlen($nama) < 5) {
17     $pesan_error .= "Nama minimal 5 digit <br>";
18 }
19
20 // cek apakah "email" sudah diisi dan memiliki karakter @@
21 if (empty($email)) {
22     $pesan_error .= "Email belum diisi <br>";
23 }
24 else if (strpos($email, "@") === false) {
25     $pesan_error .= "Penulisan alamat email salah <br>";
26 }
27
28 // jika tidak ada error, tampilkan isi form
29 if ($pesan_error == "") {
30     echo "<h1>Form Berhasil di Proses </h1>";
31     echo "Nama : $nama <br>";
32     echo "Email : $email";
33     die();
34 }
35 }
36 else {
37     $pesan_error = "";
38     $nama = "";
39     $email = "";
40 }
41 ?>
42 <!DOCTYPE html>
43 <html>
44 <head>
45     <meta charset="UTF-8">
46     <title>Belajar PHP</title>
47 </head>
48 <body>
49 <h1>Form Register</h1>
50     <?php echo $pesan_error; ?>
51     <form action="index.php" method="post">
52         <p>Nama: <input type="text" name="nama"
53             value="<?php echo $nama ?>" ></p>
54         <p>Email: <input type="text" name="email"
55             value="<?php echo $email ?>" ></p>
56         <input type="submit" name="submit" value="Proses Data" >
57     </form>
58 </body>
59 </html>
```



Gambar: proses validasi gagal karena nama kurang dari 5 karakter dan tidak ada karakter ‘@’ di dalam email

Perhatikan bagaimana cara menyatukan validasi ini. Saya menggunakan kondisi IF ELSE, sehingga hanya 1 pesan error yang ditampilkan untuk setiap isian form. Ketika *nama* belum diisi, pesan error yang tampil adalah “Nama belum diisi”, dan hanya ketika nama sudah diisi tapi kurang dari 5 karakter, barulah pesan error “*Nama minimal 5 digit*” yang tampil. Begitu juga dengan pesan error di *email*.

Jika anda ingin menambahkan validasi yang lain, misalnya nama tidak boleh kurang dari 5 digit, tapi tidak boleh lebih dari 10 digit, bisa dengan menggabungkan logikanya, seperti:

```

1 else if ((strlen($nama) < 5) OR (strlen($nama) >= 10)) {
2     $pesan_error .= "Nama minimal 5 digit dan maksimal 10 digit <br>";
3 }
```

Untuk objek form lain, kadang kita ingin nilai yang diinput harus salah satu dari *string* yang telah didefinisikan. Sebagai contoh, saya punya kotak isian form **buku** yang hanya bisa diisi dengan nama buku: “**html uncover**”, “**css uncover**”, dan “**php uncover**”. Jika user menginput judul lain, akan tampil pesan kesalahan. Bagaimana cara membuatnya?

Kita bisa memasukkan pilihan ini kedalam sebuah array, lalu memeriksanya dengan fungsi **in\_array()**, seperti contoh berikut:

```

1 // cek apakah "buku" ada di pilihan
2 $array_buku = ["html uncover", "css uncover", "php uncover"];
3 $buku = strtolower($buku);
4
5 if (!in_array($buku, $array_buku)) {
6     $pesan_error .= "Buku tidak tersedia<br>";
7 }
```

Fungsi `in_array()` akan menghasilkan TRUE apabila menemukan string di dalam array. Penjelasan fungsi ini juga telah kita bahas di dalam bab tentang *array function*. Karena saya hanya menampilkan pesan error ketika string tidak ada di dalam array, maka saya membutuhkan operator negasi ‘!’ untuk membalik logika fungsi `in_array()`.

Fungsi `strtolower()` berguna untuk menyeragamkan variabel, karena bisa saja user menulis “HTML Uncover” yang dianggap tidak sama dengan “html uncover”. Fungsi `strtolower()` akan mengkonversi inputan user menjadi huruf kecil.

Validasi selanjutnya, bagaimana jika cara kita memastikan bahwa yang diinput user adalah sebuah angka? Misalnya kita membuat kotak isian untuk harga barang, tentunya hanya bisa diisi dengan angka dan tidak boleh huruf. Untuk keperluan ini kita bisa menggunakan fungsi `is_numeric()`. Seperti contoh berikut:

```
1 // jumlah pesanan harus berupa angka
2 if (!is_numeric($harga)) {
3     $pesan_error .= "Harga barang harus dalam angka<br>";
4 }
```

Fungsi `is_numeric()` akan menghasilkan TRUE jika variabel yang diperiksa adalah sebuah “angka”, kembali saya membalikkan logika dengan operator “!“.

Satu hal yang sering membuat pusing bagi programmer pemula adalah memeriksa isian form dengan fungsi `is_integer()` atau `is_float()`. Kedua fungsi ini digunakan untuk memeriksa apakah sebuah variabel bertipe data *integer* atau *float*. Tapi kita tidak bisa menggunakan fungsi ini di dalam pemrosesan form.

Kenapa? Karena walaupun user menginput angka di dalam objek form, yang akan dikirim adalah *string*, bukan *integer* atau *float*. Misalkan saya mengisi kolom harga dengan 10000, yang akan dikirim di dalam variabel `$_POST` adalah string “10000”. Oleh karena itulah fungsi `is_integer()` dan `is_float()` akan menghasilkan nilai FALSE.

Berikut contoh penggunaan fungsi `in_array()` dan `is_numeric()` untuk membuat validasi form:

index.php

---

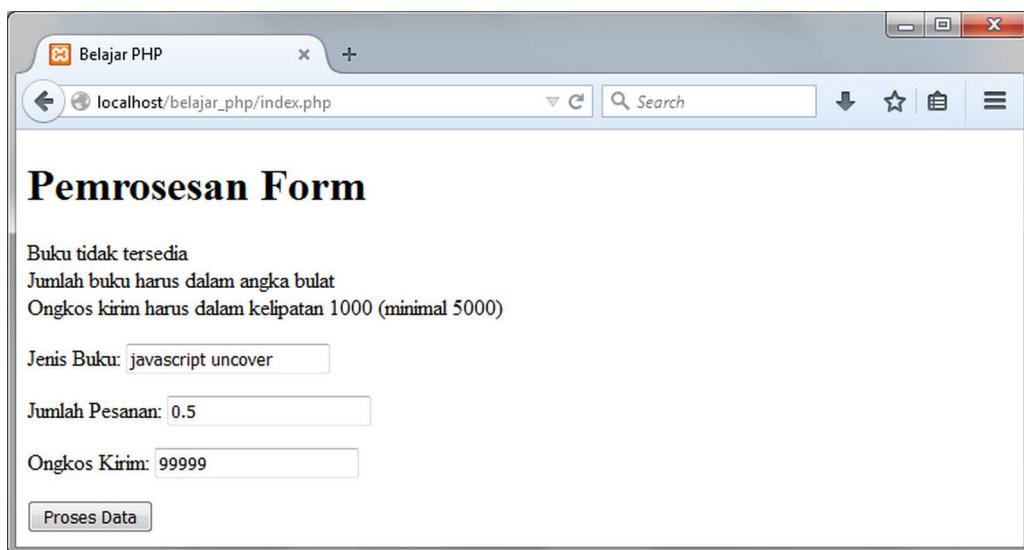
```
1 <?php
2     // cek apakah form telah di submit
3     if (isset($_POST["submit"])) {
4         // form telah disubmit, proses data
5
6         // ambil nilai form
7         $buku = htmlentities(strip_tags(trim($_POST["buku"])));
8         $jumlah = htmlentities(strip_tags(trim($_POST["jumlah"])));
9         $ongkir = htmlentities(strip_tags(trim($_POST["ongkir"])));
10
11        // siapkan variabel untuk menampung pesan error
12        $pesan_error="";
13 }
```

```
14 // cek apakah "buku" ada di pilihan
15 $array_buku = ["html uncover", "css uncover", "php uncover"];
16 $buku = strtolower($buku);
17
18 if (!in_array($buku, $array_buku)) {
19     $pesan_error .= "Buku tidak tersedia<br>";
20 }
21
22 // jumlah pesanan harus berupa angka
23 if (!is_numeric($jumlah)) {
24     $pesan_error .= "Jumlah buku harus dalam satuan angka<br>";
25 }
26 // jumlah pesanan harus >= 1 dan <= 10
27 elseif ($jumlah <= 0 OR $jumlah >= 10) {
28     $pesan_error .= "Jumlah buku antara 1-10<br>";
29 }
30 // jumlah pesanan harus angka bulat
31 elseif ($jumlah != round($jumlah)) {
32     $pesan_error .= "Jumlah buku harus dalam angka bulat<br>";
33 }
34
35 // ongkos kirim harus berupa angka
36 if (!is_numeric($ongkir) OR ($ongkir < 5000) OR ((($ongkir % 1000) != 0)) {
37     $pesan_error .= "Ongkos kirim harus dalam kelipatan 1000 (minimal 5000)";
38 }
39
40 // jika tidak ada error, tampilkan isi form
41 if ($pesan_error === "") {
42     echo "Form berhasil di proses <br>";
43     echo "Nama Buku : $buku <br>";
44     echo "Jumlah Pesanan : $jumlah <br>";
45     echo "Ongkos Kirim : $ongkir <br>";
46     die();
47 }
48 }
49 else {
50     $pesan_error = "";
51     $buku = "";
52     $jumlah = "";
53     $ongkir = "";
54 }
55 ?>
56
57 <!DOCTYPE html>
58 <html>
59 <head>
```

```

60  <meta charset="UTF-8">
61  <title>Belajar PHP</title>
62  </head>
63  <body>
64  <h1>Pemrosesan Form</h1>
65  <?php echo $pesan_error; ?>
66  <form action="index.php" method="post">
67  <p>Jenis Buku: <input type="text" name="buku"
68  value="<?php echo $buku ?>" ></p>
69  <p>Jumlah Pesanan: <input type="text" name="jumlah"
70  value="<?php echo $jumlah ?>" ></p>
71  <p>Ongkos Kirim: <input type="text" name="ongkir"
72  value="<?php echo $ongkir ?>" ></p>
73
74  <input type="submit" name="submit" value="Proses Data" >
75  </form>
76  </body>
77  </html>

```



Gambar: Pesan kesalahan tampil akibat gagal melewati validasi form

Mudah-mudahan anda bisa memahami kode program yang cukup panjang ini. Saya menyatakan berbagai teknik validasi, misalnya untuk memastikan yang diinput adalah angka bulat, saya menggunakan kode `elseif ($jumlah != round($jumlah))`.

Untuk memastikan angka yang diinput harus kelipatan 1000, saya menggunakan kondisi `($ongkir % 1000) != 0`. Masih ingat dengan operator ‘%’? Operator ‘%’ atau *modulus* akan mengembalikan sisa hasil bagi. Kode program `($ongkir % 1000 != 0)` hanya akan menghasilkan TRUE ketika variabel `$ongkir` habis dibagi 1000.

Teknik validasi seperti ini memang cukup susah untuk dirancang. Terlebih jika anda baru memulai belajar programming. Sering-sering latihan studi kasus adalah tips paling ampuh untuk melatih logika program.

## 20.13 Regular Expression

Apabila fungsi-fungsi bawaan PHP sudah tidak mencukupi untuk proses validasi, saatnya menggunakan *regular expression*. **Regular expression** (sering disingkat sebagai **regex**) adalah kumpulan huruf atau karakter yang digunakan untuk pencocokan pola (*pattern matching*). Fitur *regular expression* tidak hanya ada di dalam PHP. Hampir seluruh bahasa pemrograman modern mendukung *regular expression*.

Kita bisa menggunakan *regex* untuk berbagai keperluan, seperti membuat aplikasi pencarian atau memeriksa apakah sebuah string sama dengan pola yang kita rancang (tes validasi).

Sebagai contoh, jika saya ingin form isian “**username**” harus terdiri dari 3 buah angka dan 2 buah huruf (total 5 karakter), saya bisa merancang kode programnya menggunakan *regex*.



*Regular expression* sangat powerfull dan materinya cukup kompleks. Apa yang akan saya bahas disini hanya sedikit dari apa yang bisa dilakukan dengan *regex*.

*Regular Expression* sebenarnya bisa dijadikan bab tersendiri. Tapi karena kita fokus pada penggunaan *regex* untuk validasi form, saya memutuskan untuk tetap memasukkannya ke dalam bab ini.

Secara teknis, di dalam PHP terdapat 2 kelompok fungsi yang terkait *Regular Expression*, yakni **POSIX Regular Expressions**<sup>1</sup> dan **PERL Style Regular Expressions (PCRE)**<sup>2</sup>.

**POSIX** sudah berstatus *deprecated* dan disarankan untuk tidak digunakan lagi. Dari cara *penulisan pola*, keduanya tidak banyak berbeda. Bisa dibilang **PCRE** adalah versi yang lebih update daripada **POSIX**. Baik **POSIX** maupun **POSIX** ada di dalam PHP Manual, jika anda ingin mempelajari fungsi-fungsi *regular expression* di PHP, gunakan **PERL Style Regular Expressions (PCRE)**.

Untuk membuat proses validasi menggunakan *regular expression*, fungsi yang akan kita gunakan adalah `preg_match()`.

Fungsi `preg_match()` membutuhkan 2 buah argumen. Argumen pertama adalah pola *regular expression* dalam bentuk string, dan yang kedua adalah *string* yang ingin diperiksa. Apabila string sesuai dengan pola, fungsi `preg_match()` akan mengembalikan integer 1. Jika pola tidak sesuai, akan mengembalikan integer 0. Jika terdapat error, akan mengembalikan nilai FALSE.

Berikut contoh penggunaan fungsi `preg_match()`:

---

<sup>1</sup><http://php.net/manual/en/ref.regex.php>

<sup>2</sup><http://php.net/manual/en/book.pcre.php>

```
1 <?php
2   $hasil = preg_match("/[0-9]{5}/", "abcde");
3   echo $hasil."<br>";    // 0
4
5   $hasil = preg_match("/^([0-9]{5})$/", "12345");
6   echo $hasil."<br>";    // 1
7 ?>
```

Mari kita bahas bagaimana aturan penulisan pola *regular expression* ini.

Pertama, string *regular expression* harus di batasi oleh dua buah karakter. Dalam contoh diatas, saya menggunakan karakter ‘ / ’ sebagai pembatas. Kita juga bisa menggunakan karakter lain seperti ‘ # ’ atau ‘ | ’, namun yang sering digunakan di PHP adalah ‘ / ’.

Berikut contoh penggunaan karakter lain sebagai pembatas string *regular expression*:

```
1 <?php
2   $hasil = preg_match("#[0-9]{5}#", "abcde");
3   echo $hasil."<br>";    // 0
4
5   $hasil = preg_match("|^([0-9]{5})|", "12345");
6   echo $hasil."<br>";    // 1
7 ?>
```

Di dalam karakter pembatas inilah pola *regular expression* ditulis. Yang paling sederhana, kita bisa membuat langsung pola yang ingin diperiksa oleh fungsi `preg_match()`. Sebagai contoh, pola `/php/` akan memeriksa apakah ada kata “php” di dalam sebuah string (yang diletakkan sebagai argumen kedua). Berikut percobaannya:

```
1 <?php
2   $hasil = preg_match("/php/", "belajarphp");
3   echo $hasil."<br>";    // 1
4
5   $hasil = preg_match("/php/", "belajarhtml");
6   echo $hasil."<br>";    // 0
7
8   $hasil = preg_match("/php/", "belajar php");
9   echo $hasil."<br>";    // 1
10
11  $hasil = preg_match("/php/", "php itu asyik");
12  echo $hasil."<br>";    // 1
13
14  $hasil = preg_match("/php/", "sedang serius belajar");
15  echo $hasil."<br>";    // 0
16
17  $hasil = preg_match("/php/", "...ph...p");
```

```

18 echo $hasil."<br>"; // 0
19
20 $hasil = preg_match("/php/", "1234php1234");
21 echo $hasil."<br>"; // 1
22 ?>

```

Seperti yang terlihat, regular expression /php/ akan cocok dengan string apapun selama ada “php”, tidak peduli di posisi mana “php” berada.

Bagaimana jika ingin mengatur agar kata yang dicari harus berada di depan, ditengah atau diakhiri kalimat? *Regular expression* menyediakan 2 karakter khusus untuk membuat pengaturan ini:

- ^ : karakter harus berada di awal string
- \$ : karakter harus berada di akhir string

Sekarang, jika saya ingin memastikan bahwa kata “php” harus berada di awal kalimat, polanya ditulis sebagai: /^php/. Berikut contohnya:

```

1 <?php
2   $hasil = preg_match("/^php/", "belajarphp");
3   echo $hasil."<br>"; // 0
4
5   $hasil = preg_match("/^php/", "belajar php");
6   echo $hasil."<br>"; // 0
7
8   $hasil = preg_match("/^php/", "php itu asyik");
9   echo $hasil."<br>"; // 1
10
11  $hasil = preg_match("/^php/", "aaaaphpaaa");
12  echo $hasil."<br>"; // 0
13
14  $hasil = preg_match("/^php/", "ph..p");
15  echo $hasil."<br>"; // 0
16
17  $hasil = preg_match("/^php/", "phpaaaa");
18  echo $hasil."<br>"; // 1
19 ?>

```

Sedangkan jika saya menggunakan pola /php\$/, maka hanya akan cocok dengan “belajarphp”, “aaaphp”, atau “apa itu php”, yakni string atau kalimat yang diakhiri dengan kata “php”.

Bagaimana jika saya menulis /^php\$/ ? pola ini akan cocok hanya dengan kata “php”, karena “php” harus berada di awal string sekaligus diakhiri string.

Untuk menghindari pola yang bermakna ambigu, kita bisa menambah tanda kurung. Misalnya menjadi /^(php)\$/.

Selain menentukan posisi, regular expression juga menyediakan mekanisme untuk menentukan berapa kali sebuah pola bisa muncul. Caranya adalah dengan menggunakan karakter-karakter berikut:

- $(p)^+$ : cocok dengan string yang terdiri dari 1 atau lebih pola. Sebagai contoh, pola  $/(aku)^+/-$  akan cocok dengan: “*ada aku*”, “*aku bisa*”, “*aku aku aku*”. Tapi tidak cocok dengan “*boleh saja*”. Pola  $/(aku)^+/-$  artinya, harus ada string “*aku*” minimal 1 kali.
- $(p)^*$ : cocok dengan string yang terdiri dari 0 atau lebih pola. Sebagai contoh, pola  $/(aku)^*/-$  akan cocok dengan: “*ada aku*”, “*aku bisa*”, “*aku aku aku*”, maupun “*boleh saja*”. Sebenarnya, pola  $/(aku)^*/-$  akan cocok dengan string apapun, baik yang memiliki karakter “*aku*” maupun tidak.
- $(p)^?$ : cocok dengan string yang terdiri dari 0 atau 1 pola (tidak boleh lebih). Sebagai contoh, pola  $/(aku)^?/-$  akan cocok dengan: “*ada aku*”, “*bisa*” namun tidak dengan “*aku aku aku*”.
- $(p)^{a}$ : cocok dengan string yang memiliki karakter “*p*” berurutan sebanyak “*a*”. Sebagai contoh, pola  $/(a)^{4}/-$  cocok dengan: “*bbaaaabb*” atau “*aaaaaaaa*”, namun tidak cocok dengan “*aaabbaa*”.  $/(a)^{4}/-$  artinya harus terdapat 4 karakter “*a*” yang saling tersambung di dalam string.
- $(p)^{a,b}$ : cocok dengan string yang memiliki karakter *p* sebanyak “*a*” atau sebanyak “*b*”. Sebagai contoh, pola  $/(sa)^{2,3}/-$  cocok dengan “*sasa*”, “*qqsasasaqq*”, “*sasa sa*”, maupun “*sasasasasasa*”.  $/(sa)^{2,3}/-$  artinya harus terdapat setidaknya 2 atau 3 karakter “*sa*”.

Untuk kumpulan karakter atau *range*, kita bisa menggunakan tanda kurung siku. Sebagai contoh, untuk membuat range karakter angka, bisa ditulis sebagai  $/[0123456789]/$  atau disingkat menjadi  $/[0-9]/$ .

Untuk huruf A-Z, ditulis menjadi  $/[A-Z]/$ . Range ini mengikuti urutan karakter dalam tabel ASCII, sehingga  $/[A-z]/$  terdiri dari seluruh alphabet huruf besar, karakter ^ \ [ ] \_ ', serta alphabet huruf kecil.

Selain membuat range secara manual, *regular expression* juga menyediakan range karakter khusus:

- . : tanda titik berarti mewakili sebuah karakter apapun.
- \s: mewakili karakter whitespace (spasi, tab, newline)
- \S: mewakili karakter selain whitespace, perhatikan kali ini adalah S besar
- \d: mewakili karakter angka (0-9)
- \D: mewakili karakter selain angka
- \w: mewakili karakter yang umum dalam sebuah kata (*word character* : a-z, A-Z, 0-9, \_)
- \W: mewakili *non-word character*, terdiri dari karakter-karakter khusus seperti \$, %, \*, dll.

Untuk membuat logika “atau”, bisa menggunakan karakter *pipe*: “ | “. Sebagai contoh, pola  $/ma(kan|ndi|kassar)/$  akan cocok dengan string “*makan*”, “*mandi*”, dan “*makassar*”.

Untuk logika “not” atau negasi, bisa menggunakan karakter “^” namun harus di dalam sebuah himpunan (*range*). Sebagai contoh, pola  $/[^aeiou]/$  akan cocok dengan karakter apa saja selain huruf fokal.

Namun bagaimana cara menggunakan karakter khusus \* atau ^ di dalam pola? Untuk keperluan ini kita bisa menambahkan *backslash* sebagai *escape character*. Karakter khusus yang ada di dalam regular expression PHP adalah: ^ \$ + \* ? . | ( ) { } [ ] \

Misalnya saya ingin membuat pola angka perkalian seperti 5\*6, 3\*3, atau 1\*9. Saya bisa menggunakan regular expression /[0-9]\*[0-9]/. Namun tanda bintang ini bukan mewakili karakter “\*”, tapi bermakna bahwa angka [0-9] bisa ditulis sebanyak 0 atau lebih. Seharusnya pola diatas ditulis menjadi /[0-9]\\*[0-9]/.

Baik, cukup tentang teorinya (mudah-mudahan anda belum ngantuk). Mari kita praktek membuat pola *regular expression*.

Contoh kasus pertama, saya ingin membuat pola yang diawali dengan 3 digit huruf, lalu diikuti oleh 2 digit angka, seperti: AAA01, BCE99, JIK12, atau PPP89. Karakter ini harus dalam huruf besar. Bagaimana pola *regular expression*-nya?

Untuk range 3 digit huruf besar, saya bisa menggunakan [A-Z]{3}. Untuk 2 digit huruf, bisa dengan [0-9]{2}. Jadi pola *regular expression*-nya adalah: /[A-Z]{3}[0-9]{2}/. Mari kita coba:

```
1 <?php
2     $hasil = preg_match("/[A-Z]{3}[0-9]{2}/", "AAA01");
3     echo $hasil."<br>"; // 1
4
5     $hasil = preg_match("/[A-Z]{3}[0-9]{2}/", "BCE99");
6     echo $hasil."<br>"; // 1
7
8     $hasil = preg_match("/[A-Z]{3}[0-9]{2}/", "JIK12");
9     echo $hasil."<br>"; // 1
10
11    $hasil = preg_match("/[A-Z]{3}[0-9]{2}/", "aaa01");
12    echo $hasil."<br>"; // 0
13
14    $hasil = preg_match("/[A-Z]{3}[0-9]{2}/", "JIKAA");
15    echo $hasil."<br>"; // 0
16
17    $hasil = preg_match("/[A-Z]{3}[0-9]{2}/", "zzAAA01zz");
18    echo $hasil."<br>"; // 1
19 ?>
```

Untuk 5 contoh pertama, sudah sesuai dengan keinginan. Tapi perhatikan contoh terakhir. String zzAAA01zz dianggap valid. Ini terjadi karena pola *regular expression* yang saya gunakan belum menyertakan posisi dari string. Oleh karena itu, saya harus mengubahnya menjadi: /^[A-Z]{3}[0-9]{2}\$/:

```

1 <?php
2   $hasil = preg_match("/^ [A-Z]{3}[0-9]{2}$/, "AAA01");
3   echo $hasil."<br>"; // 0
4
5   $hasil = preg_match("/^ [A-Z]{3}[0-9]{2}$/, "zzAAA01zz");
6   echo $hasil."<br>"; // 0
7 ?>

```

Penambahan tanda ^ diawal dan tanda \$ diakhir pola memastikan bahwa kata harus berada di awal dan akhir string (tidak boleh ada karakter lain).

Contoh kasus kedua, bisakah anda membuat pola *regular expression* untuk nomor polisi kendaraan sipil di Indonesia?

Sebuah nomor polisi diawal dengan 1 atau 2 huruf, lalu diikuti dengan 1 sampai 4 angka, dan diakhiri dengan 1 sampai 3 huruf, contohnya: B123ZZ, BK1N, BB2222RZY, dst. Untuk sementara tidak perlu ada karakter spasi diantara huruf dan angka.

Inilah pola yang saya gunakan: /^[A-Z]{1,2}[0-9]{1,4}[A-Z]{1,3}\$. Bisakah anda menerjemahkan pola ini? Berikut percobaan yang saya lakukan:

```

1 <?php
2   $hasil = preg_match("/^ [A-Z]{1,2}[0-9]{1,4}[A-Z]{1,3}$/, "BB123ZZZ");
3   echo $hasil."<br>"; // 1
4
5   $hasil = preg_match("/^ [A-Z]{1,2}[0-9]{1,4}[A-Z]{1,3}$/, "D78A");
6   echo $hasil."<br>"; // 1
7
8   $hasil = preg_match("/^ [A-Z]{1,2}[0-9]{1,4}[A-Z]{1,3}$/, "B1RI");
9   echo $hasil."<br>"; // 1
10
11  $hasil = preg_match("/^ [A-Z]{1,2}[0-9]{1,4}[A-Z]{1,3}$/, "BA9US");
12  echo $hasil."<br>"; // 1
13
14  $hasil = preg_match("/^ [A-Z]{1,2}[0-9]{1,4}[A-Z]{1,3}$/, "1BA9US");
15  echo $hasil."<br>"; // 0
16
17  $hasil = preg_match("/^ [A-Z]{1,2}[0-9]{1,4}[A-Z]{1,3}$/, "BA 9 US");
18  echo $hasil."<br>"; // 0
19 ?>

```

Dalam contoh terakhir, string BA 9 US dianggap tidak valid karena terdapat karakter spasi. Bagaimana cara mengakomodir pola ini? saya ingin agar baik BA9US maupun BA 9 US tetap dianggap valid. Berikut revisi pola regular expressionnya: /^[A-Z]{1,2} ?[0-9]{1,4} ?[A-Z]{1,3}\$.

Perhatikan penambahan “ ?” (sebuah spasi dan sebuah tanda ?) diantara huruf dan angka. Ini artinya boleh terdapat 0 atau 1 spasi. Berikut prakteknya:

```

1 <?php
2   $hasil = preg_match("/^ [A-Z]{1,2} [0-9]{1,4} [A-Z]{1,3}$/, "BB123ZZZ");
3   echo $hasil."<br>"; // 1
4
5   $hasil = preg_match("/^ [A-Z]{1,2} [0-9]{1,4} [A-Z]{1,3}$/, "D 78 A");
6   echo $hasil."<br>"; // 1
7
8   $hasil = preg_match("/^ [A-Z]{1,2} [0-9]{1,4} [A-Z]{1,3}$/, "B 1 RI");
9   echo $hasil."<br>"; // 1
10
11  $hasil = preg_match("/^ [A-Z]{1,2} [0-9]{1,4} [A-Z]{1,3}$/, "BA9US");
12  echo $hasil."<br>"; // 1
13
14  $hasil = preg_match("/^ [A-Z]{1,2} [0-9]{1,4} [A-Z]{1,3}$/, "BA 9 US");
15  echo $hasil."<br>"; // 0
16 ?>

```

Sebagai contoh kasus terakhir (sebelum kita kembali ke pemrosesan form), dapatkah anda membuat sebuah pola *regular expression* untuk mengecek apakah sebuah sting valid untuk alamat email?

Berikut beberapa kemungkinan pola yang saya gunakan:

```

1 <?php
2   $hasil = preg_match("/@/", "duniailkom@gmail.com");
3   echo $hasil."<br>"; // 1
4
5   $hasil = preg_match("/.+@.+/", "duniailkom@gmail.com");
6   echo $hasil."<br>"; // 1
7
8   $hasil = preg_match("/.+@.+\\..+/", "duniailkom@gmail.com");
9   echo $hasil."<br>"; // 1
10 ?>

```

Pola pertama: `/@/` adalah yang paling sederhana. Sebuah string dianggap valid selama memiliki karakter '@'. Ini tidak berbeda dari fungsi `strpos($email, "@")`.

Pola kedua: `/.+@.+/` berarti sebelum dan sesudah karakter '@' harus terdapat 1 atau lebih karakter. Sudah sedikit mendingan.

Pola ketiga: `/.+@.+\\..+/"` saya menambahkan syarat lagi untuk bagian setelah tanda '@'. Sebagaimana yang kita tahu, sebuah alamat email memiliki domain di belakangnya, seperti `@gmail.com`, `@yahoo.com`, atau `@ui.ac.id`.

Pada bagian domain ini, harus ada sebuah tanda titik “.”. Oleh karena itu ini bisa dijadikan syarat tambahan untuk *regex* email. Saya membuat `“.+\\..+”` yang bisa dibaca: minimal 1 karakter apa saja, tanda titik, lalu minimal satu karakter lagi. Saya harus men-*escape* tanda “titik” kedua, karena merupakan karakter khusus di dalam *regular expression*.

Berikut pengujian regex untuk email:

```
1 <?php
2     $hasil = preg_match("/.+\@.+\..+/", "test");
3     echo $hasil."<br>"; // 0
4
5     $hasil = preg_match("/.+\@.+\..+/", "test@");
6     echo $hasil."<br>"; // 0
7
8     $hasil = preg_match("/.+\@.+\..+/", "test@hehe");
9     echo $hasil."<br>"; // 0
10
11    $hasil = preg_match("/.+\@.+\..+/", "@hehe.com");
12    echo $hasil."<br>"; // 0
13
14    $hasil = preg_match("/.+\@.+\..+/", "a@a.a");
15    echo $hasil."<br>"; // 1
16
17    $hasil = preg_match("/.+\@.+\..+/", "j0n1@d1sin1.bagus");
18    echo $hasil."<br>"; // 1
19
20    $hasil = preg_match("/.+\@.+\..+/", "01@00.01");
21    echo $hasil."<br>"; // 1
22
23    $hasil = preg_match("/.+\@.+\..+/", "s@hehe.com.net.org");
24    echo $hasil."<br>"; // 1
25 ?>
```

Dalam kenyataanya, pola *regular expresion* untuk alamat email sangat kompleks. Pola yang saya gunakan diatas masih terhitung sederhana dan belum mengakomodasi semua kemungkinan alamat email yang tidak valid.

Sebagai contoh, alamat “\*%@%%.()” akan dianggap valid. Jika anda ingin melihat versi lengkap *regular expression* untuk alamat email, bisa mengunjungi: [emailregex.com](http://emailregex.com)<sup>3</sup>. Namun siapkan mental sebelum melihatnya :)

## 20.14 Validasi Form Menggunakan Regular Expression

Kembali ke pemrosesan form dengan PHP, mari kita praktikkan konsep *regular expression* yang telah dipelajari ke dalam validasi form. Berikut contohnya:

---

<sup>3</sup><http://emailregex.com>

```
1 // email harus sesuai dengan format
2 elseif (!preg_match("/.+@.+\..+/", $email) ) {
3     $pesan_error .= "Format email tidak sesuai <br>";
4 }
```

Potongan kode diatas mengikuti format validasi yang sudah kita pelajari. Disini saya membalik logika hasil fungsi `preg_match()`. Artinya, hanya ketika fungsi ini menghasilkan nilai 0 atau FALSE, pesan error akan ditampilkan.

Berikut contoh kode programnya lengkapnya:

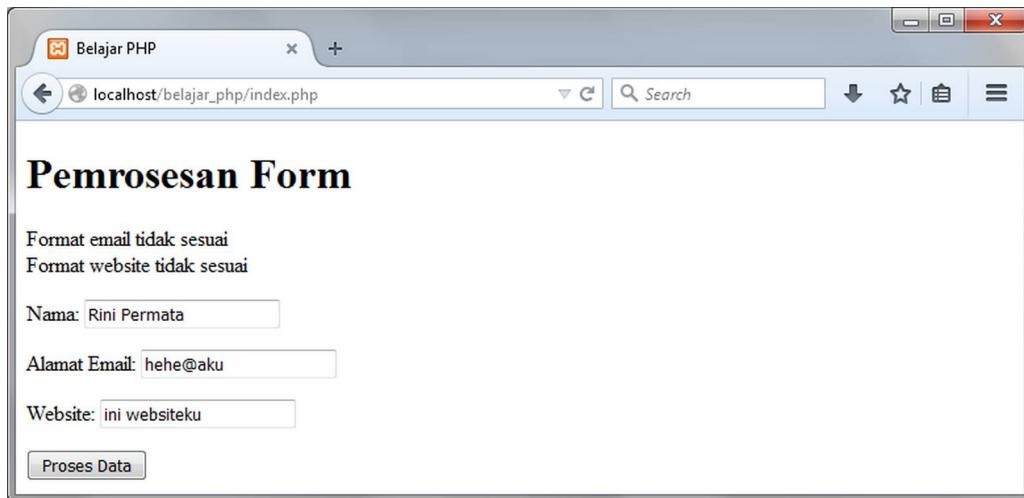
index.php

---

```
1 <?php
2     // cek apakah form telah di submit
3     if (isset($_POST["submit"])) {
4         // form telah disubmit, proses data
5
6         // ambil nilai form
7         $nama = htmlentities(strip_tags(trim($_POST["nama"])));
8         $email = htmlentities(strip_tags(trim($_POST["email"])));
9         $website = htmlentities(strip_tags(trim($_POST["website"])));
10
11        // siapkan variabel untuk menampung pesan error
12        $pesan_error="";
13
14        // cek apakah "nama" sudah diisi atau tidak
15        if (empty($nama)) {
16            $pesan_error .= "Nama belum diisi <br>";
17        }
18
19        // cek apakah "email" sudah diisi atau tidak
20        if (empty($email)) {
21            $pesan_error .= "Email belum diisi <br>";
22        }
23        // email harus sesuai dengan format
24        elseif (!preg_match("/.+@.+\..+/", $email) ) {
25            $pesan_error .= "Format email tidak sesuai <br>";
26        }
27
28        // email harus sesuai dengan format
29        if (!empty($website) AND !preg_match("/.+@.+\..+/", $website) ) {
30            $pesan_error .= "Format website tidak sesuai";
31        }
32
33        // jika tidak ada error, tampilkan isi form
34        if ($pesan_error === "") {
```

```
35     echo "Form berhasil di proses <br>";
36     echo "Nama : $nama <br>";
37     echo "Email : $email <br>";
38     echo "Website : $website <br>";
39     die();
40 }
41 }
42 else {
43     $pesan_error = "";
44     $nama = "";
45     $email = "";
46     $website = "";
47 }
48 ?>
49
50 <!DOCTYPE html>
51 <html>
52 <head>
53     <meta charset="UTF-8">
54     <title>Belajar PHP</title>
55 </head>
56 <body>
57 <h1>Pemrosesan Form</h1>
58 <?php echo $pesan_error; ?>
59 <form action="index.php" method="post">
60     <p>Nama: <input type="text" name="nama"
61             value="<?php echo $nama ?>" ></p>
62     <p>Alamat Email: <input type="text" name="email"
63             value="<?php echo $email ?>" ></p>
64     <p>Website: <input type="text" name="website"
65             value="<?php echo $website ?>" ></p>
66
67     <input type="submit" name="submit" value="Proses Data" >
68 </form>
69 </body>
70 </html>
```

---



Gambar: Validasi gagal karena tidak cocok dengan pola regular expression

Dalam contoh ini, terdapat 3 form isian yang terdiri dari: *nama*, *email*, dan *website*. Saya menggunakan *regular expression* untuk memeriksa pola karakter nilai *email* dan *website*, jika tidak valid, tampilkan pesan kesalahan.

Seperti yang terlihat, *regular expression* membuat proses validasi semakin mudah. Silahkan anda coba praktek menggunakan validasi *regular expression* ini. Misalnya untuk kolom isian untuk NIM (*Nomor Induk Mahasiswa*), nomor KTP, dll. Semakin banyak latihan akan membuat anda semakin paham cara penggunaannya.

## 20.15 Repopulate Isian Form Select

Cara mengisi kembali isian form (*form re-populate*) sudah kita pelajari dalam materi sebelumnya, namun hanya untuk isian form yang berupa kotak input, atau tag `<input type="text">`. Bagaimana dengan isian form lain? seperti *select*, *checkbox*, dan *radio button*? Inilah yang akan saya bahas kali ini.

Yang pertama, saya akan membahas cara *re-populate* isian form untuk list, atau tag `<select>`. Sebagai contoh form, silahkan jalankan kode berikut:

### index.php

```

1 <?php
2 // cek apakah form telah di submit
3 if (isset($_POST["submit"])) {
4 // form telah disubmit, proses data
5
6 // ambil nilai form
7 $buku = htmlentities(strip_tags(trim($_POST["buku"])));
8
9 // siapkan variabel untuk menampung pesan error
10 $pesan_error="";
11

```

```
12  // cek apakah "buku" sudah dipilih atau tidak
13  if (empty($buku)) {
14      $pesan_error .= "Buku belum dipilih <br>";
15  }
16
17  // cek buku = JavaScript Uncover atau MySQL Uncover
18  if ($buku == "JavaScript Uncover" OR $buku == "MySQL Uncover") {
19      $pesan_error .= "Maaf, buku belum tersedia <br>";
20  }
21
22  // jika tidak ada error, tampilkan isi form
23  if ($pesan_error === "") {
24      echo "Form berhasil di proses <br>";
25      echo "Buku : $buku <br>";
26      die();
27  }
28 }
29 else {
30     $pesan_error = "";
31 }
32 ?>
33 <!DOCTYPE html>
34 <html>
35 <head>
36     <meta charset="UTF-8">
37     <title>Belajar PHP</title>
38 </head>
39 <body>
40 <h1>Pemrosesan Form</h1>
41     <?php echo $pesan_error; ?>
42     <form action="index.php" method="post">
43         <p>Jenis Buku :
44             <select name="buku" id="buku">
45                 <option value="HTML Uncover">HTML Uncover</option>
46                 <option value="CSS Uncover">CSS Uncover</option>
47                 <option value="PHP Uncover">PHP Uncover</option>
48                 <option value="JavaScript Uncover">JavaScript Uncover</option>
49                 <option value="MySQL Uncover">MySQL Uncover</option>
50             </select>
51         </p>
52         <input type="submit" name="submit" value="Proses Data" >
53     </form>
54 </body>
55 </html>
```

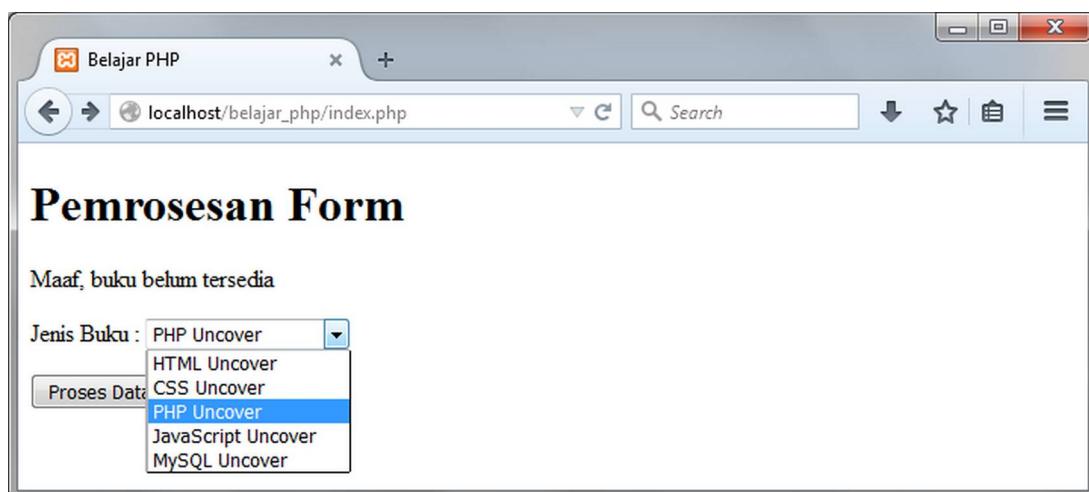
Kode diatas akan menampilkan sebuah form yang terdiri dari list pilihan nama buku. List ini

dibuat menggunakan tag `<select>` dan `<option>`. Pada bagian atas, saya membuat kode PHP untuk memproses hasil dari form. Semua ini sudah kita pelajari sebelumnya.

Saya membuat sebuah validasi:

```
1 // cek buku = JavaScript Uncover atau MySQL Uncover
2 if ($buku == "JavaScript Uncover" OR $buku == "MySQL Uncover") {
3     $pesan_error .= "Maaf, buku belum tersedia <br>";
4 }
```

Ini artinya, ketika user memilih buku *JavaScript Uncover* atau *MySQL Uncover*, akan tampil pesan error: "Maaf, buku belum tersedia".



Gambar: Pesan error jika memilih buku JavaScript atau MySQL

Validasi ini sengaja saya rancang sebagai bahan praktek kita. Perhatikan, ketika saya memilih buku *JavaScript Uncover* atau *MySQL Uncover* dan men-submit form, halaman akan *reload* dan pilihan buku di reset ke kondisi awal. Bagaimana cara agar pilihan ini tetap dan tidak berubah?

Sebagaimana yang kita tahu, agar kotak isian `select` bisa terpilih, harus menambahkan atribut `"selected"`. Sebagai contoh, agar pilihan buku "PHP Uncover" bisa terpilih, saya bisa menulis:

```
<option value="PHP Uncover" selected>PHP Uncover</option>
```

Trik yang akan saya gunakan adalah dengan memeriksa isi variabel `$_POST["buku"]`, lalu menampilkan atribut `"selected"` kepada tag `<option>` yang sesuai. Caranya, dengan menyiapkan 5 variabel untuk masing-masing tag `<option>` lalu membuat sebuah struktur `case`, seperti berikut ini:

```
1 // siapkan variabel untuk re-populate pilihan buku
2 $select_html=""; $select_css=""; $select_php="";
3 $select_javascript=""; $select_mysql="";
4
5 switch($buku) {
6     case "HTML Uncover" : $select_html = "selected"; break;
7     case "CSS Uncover" : $select_css = "selected"; break;
8     case "PHP Uncover" : $select_php = "selected"; break;
9     case "JavaScript Uncover" : $select_javascript = "selected"; break;
10    case "MySQL Uncover" : $select_mysql = "selected"; break;
11 }
```

Misalkan saat ini user memilih buku “*JavaScript Uncover*” dan men-klik tombol **submit**. Variabel **\$buku** yang berasal dari `$_POST["buku"]` akan berisi nilai “*JavaScript Uncover*”. Dengan menggunakan kode diatas, variabel `$select_javascript` akan berisi string “`selected`”. Sedangkan 4 variabel lain akan berisi string kosong.

Kemudian, di dalam form, saya bisa menambahkan kode berikut:

```
1 <p>Jenis Buku :
2   <select name="buku" id="buku">
3     <option value="HTML Uncover" <?php echo $select_html ?>>
4       HTML Uncover </option>
5     <option value="CSS Uncover" <?php echo $select_css ?>>
6       CSS Uncover</option>
7     <option value="PHP Uncover" <?php echo $select_php ?>>
8       PHP Uncover</option>
9     <option value="JavaScript Uncover" <?php echo $select_javascript ?>>
10       JavaScript Uncover</option>
11     <option value="MySQL Uncover" <?php echo $select_mysql ?>>
12       MySQL Uncover</option>
13   </select>
14 </p>
```

Perhatikan bagaimana setiap tag `<option>` ini memiliki pasangan variabel `$select_`. Dari semua variabel ini, hanya 1 pilihan yang akan berisi string “`selected`”, yakni yang sudah diproses oleh kode kita sebelumnya.

Berikut kode lengkap untuk me-repopulate isian form select:

## index.php

```
1 <?php
2 // cek apakah form telah di submit
3 if (isset($_POST["submit"])) {
4     // form telah disubmit, proses data
5
6     // ambil nilai form
7     $buku = htmlentities(strip_tags(trim($_POST["buku"])));
8
9     // siapkan variabel untuk menampung pesan error
10    $pesan_error="";
11
12    // cek apakah "buku" sudah dipilih atau tidak
13    if (empty($buku)) {
14        $pesan_error .= "Buku belum dipilih <br>";
15    }
16
17    // cek buku = JavaScript Uncover atau MySQL Uncover
18    if ($buku == "JavaScript Uncover" OR $buku == "MySQL Uncover") {
19        $pesan_error .= "Maaf, buku belum tersedia <br>";
20    }
21
22    // siapkan variabel untuk re-populate pilihan buku
23    $select_html=""; $select_css=""; $select_php="";
24    $select_javascript=""; $select_mysql="";
25
26    switch($buku) {
27        case "HTML Uncover" : $select_html = "selected"; break;
28        case "CSS Uncover" : $select_css = "selected"; break;
29        case "PHP Uncover" : $select_php = "selected"; break;
30        case "JavaScript Uncover" : $select_javascript = "selected"; break;
31        case "MySQL Uncover" : $select_mysql = "selected"; break;
32    }
33
34    // jika tidak ada error, tampilkan isi form
35    if ($pesan_error === "") {
36        echo "Form berhasil di proses <br>";
37        echo "Buku : $buku <br>";
38        die();
39    }
40 }
41 else {
42     $pesan_error = "";
43     $select_html="selected";
44     $select_css=""; $select_php="";
```

```
45     $select_javascript="" ; $select_mysql="";
46 }
47 ?>
48 <!DOCTYPE html>
49 <html>
50 <head>
51   <meta charset="UTF-8">
52   <title>Belajar PHP</title>
53 </head>
54 <body>
55 <h1>Pemrosesan Form</h1>
56 <?php echo $pesan_error; ?>
57 <form action="index.php" method="post">
58 <p>Jenis Buku :
59   <select name="buku" id="buku">
60     <option value="HTML Uncover" <?php echo $select_html ?>>
61     HTML Uncover </option>
62     <option value="CSS Uncover" <?php echo $select_css ?>>
63     CSS Uncover</option>
64     <option value="PHP Uncover" <?php echo $select_php ?>>
65     PHP Uncover</option>
66     <option value="JavaScript Uncover" <?php echo $select_javascript ?>>
67     JavaScript Uncover</option>
68     <option value="MySQL Uncover" <?php echo $select_mysql ?>>
69     MySQL Uncover</option>
70   </select>
71 </p>
72   <input type="submit" name="submit" value="Proses Data" >
73 </form>
74 </body>
75 </html>
```

---

Silahkan anda coba dengan memilih buku “*JavaScript Uncover*” atau “*MySQL Uncover*”. Validasi form akan menampilkan pesan error seperti sebelumnya, namun kali ini list pilihan buku akan *tetap* dan mengikuti pilihan terakhir dari user.

Teknik yang saya gunakan diatas cocok untuk tag `<select>` yang isinya dibuat manual, dimana saya menulis langsung tag `<option>` di dalam kode program. Namun bagaimana jika isian ini juga di generate oleh PHP? Berikut contoh kasusnya:

## index.php

```
1 <?php
2     // buat array untuk tanggal
3     $arr_bln = array( "1"=>"Januari",
4                         "2"=>"Februari",
5                         "3"=>"Maret",
6                         "4"=>"April",
7                         "5"=>"Mei",
8                         "6"=>"Juni",
9                         "7"=>"Juli",
10                        "8"=>"Agustus",
11                        "9"=>"September",
12                        "10"=>"Oktober",
13                        "11"=>"Nopember",
14                        "12"=>"Desember" );
15
16     // cek apakah form telah di submit
17     if (isset($_POST["submit"])) {
18         // form telah disubmit, proses data
19
20         // ambil nilai form
21         $tgl = htmlentities(strip_tags(trim($_POST["tgl"])));
22         $bln = htmlentities(strip_tags(trim($_POST["bln"])));
23         $thn = htmlentities(strip_tags(trim($_POST["thn"])));
24
25         // siapkan variabel untuk menampung pesan error
26         $pesan_error="";
27
28         // cek apakah "tgl" < 10
29         if ($tgl <= 10) {
30             $pesan_error .= "Maaf, tanggal harus > 10 <br>";
31         }
32
33         // jika tidak ada error, tampilkan isi form
34         if ($pesan_error === "") {
35             echo "Form berhasil di proses <br>";
36             echo "Tanggal dikirim : $tgl - $bln - $thn <br>";
37             die();
38         }
39     }
40     else {
41         $pesan_error = "";
42     }
43 ?>
44 <!DOCTYPE html>
```

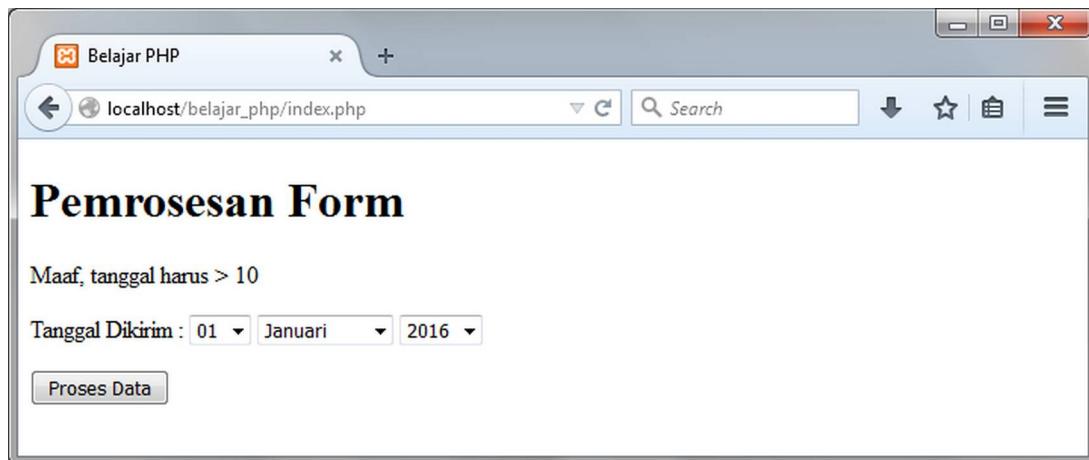
```
45 <html>
46 <head>
47   <meta charset="UTF-8">
48   <title>Belajar PHP</title>
49 </head>
50 <body>
51 <h1>Pemrosesan Form</h1>
52   <?php echo $pesan_error; ?>
53   <form action="index.php" method="post">
54     <p>
55       <label for="tgl" >Tanggal Dikirim : </label>
56       <select name="tgl" id="tgl">
57         <?php
58           for ($i = 1; $i <= 31; $i++) {
59             echo "<option value = $i >";
60             echo str_pad($i, 2, "0", STR_PAD_LEFT);
61             echo "</option>";
62           }
63         ?>
64       </select>
65       <select name="bln">
66         <?php
67           foreach ($arr_bln as $key => $value) {
68             echo "<option value=\"$key\">{$value}</option>";
69           }
70         ?>
71       </select>
72       <select name="thn">
73         <?php
74           for ($i = 2016; $i <= 2020; $i++) {
75             echo "<option value = $i >";
76             echo "$i </option>";
77           }
78         ?>
79       </select>
80     </p>
81     <input type="submit" name="submit" value="Proses Data" >
82   </form>
83 </body>
84 </html>
```

Kali ini saya memiliki form yang berisi tanggal. Terdapat 3 tag `<select>`, masing-masing untuk pilihan *tanggal*, *bulan*, dan *tahun*. Untuk inputan *tanggal* dan *tahun*, digenerate menggunakan perulangan PHP. Sedangkan untuk inputan bulan, di generate dari sebuah array `$arr_bln` yang berisi nama-nama bulan dalam bahasa indonesia.

Silahkan anda pelajari sejenak bagaimana kode PHP dan HTML ini saling bekerja sama. Seperti pembahasan sebelumnya, tujuan kita adalah mempelajari cara *re-populate form*. Pesan kesalahan saya buat dengan kode validasi:

```
1 if ($tgl <= 10) {
2     $pesan_error .= "Maaf, tanggal harus > 10 <br>";
3 }
```

Artinya, apabila tanggal dipilih kurang dari 10, tampilkan pesan error. Ketika ini yang terjadi, halaman *reload* dan tampilan tanggal kembali ke settingan awal (01 Januari 2016).



Gambar: Isian tanggal akan kembali ke pilihan awal apabila terjadi error

Jadi, bagaimana cara membuat isian form ini tetap mempertahankan nilainya? Sama seperti sebelumnya, untuk membuat pilihan tag `<option>` menjadi nilai *default* (pilihan yang terpilih ketika form di load), kita perlu menambahkan atribut `selected`.

Karena seluruh isian form digenerate menggunakan PHP, kita bisa memanfaatkan hal ini. Sewaktu form diproses, variabel `$tgl` akan berisi nilai dari `$_POST["tgl"]`. Ketika memproses perulangan untuk tanggal, saya bisa menggunakan kode berikut:

```
1 <select name="tgl" id="tgl">
2     <?php
3         for ($i = 1; $i <= 31; $i++) {
4             if ($i==$tgl){
5                 echo "<option value = $i selected>";
6             }
7             else {
8                 echo "<option value = $i >";
9             }
10            echo str_pad($i,2,"0",STR_PAD_LEFT);
11            echo "</option>";
12        }
13    ?>
14 </select>
```

Di dalam proses perulangan, saya menambahkan sebuah struktur if untuk memeriksa apakah angka yang di-generate saat ini sama dengan isi variabel `$tgl`. Jika sama, tampilkan pilihan `<option>` dengan tambahan nilai “`selected`”:

```
echo "<option value = $i selected>";
```

Jika tidak sama, tampilkan tag `<option>` seperti biasa:

```
echo "<option value = $i >";
```

Teknik ini juga bisa diterapkan untuk pilihan form tahun:

```
1 <select name="thn">
2   <?php
3     for ($i = 2016; $i <= 2020; $i++) {
4       if ($i==$thn){
5         echo "<option value = $i selected>";
```

Namun bagaimana dengan pilihan bulan? Caranya hampir mirip, karena nama bulan juga dihasilkan oleh perulangan `foreach`. Berikut kode yang saya gunakan:

```
1 <select name="bln">
2   <?php
3     foreach ($arr_bln as $key => $value) {
4       if ($key==$bln){
5         echo "<option value=\"$key\" selected>{$value}</option>";
```

Berikut kode lengkap untuk re-populate isian form dengan 3 tag `<select>`:

## index.php

```
1 <?php
2     // buat array untuk tanggal
3     $arr_bln = array( "1"=>"Januari",
4                         "2"=>"Februari",
5                         "3"=>"Maret",
6                         "4"=>"April",
7                         "5"=>"Mei",
8                         "6"=>"Juni",
9                         "7"=>"Juli",
10                        "8"=>"Agustus",
11                        "9"=>"September",
12                        "10"=>"Oktober",
13                        "11"=>"Nopember",
14                        "12"=>"Desember" );
15
16     // cek apakah form telah di submit
17     if (isset($_POST["submit"])) {
18         // form telah disubmit, proses data
19
20         // ambil nilai form
21         $tgl = htmlentities(strip_tags(trim($_POST["tgl"])));
22         $bln = htmlentities(strip_tags(trim($_POST["bln"])));
23         $thn = htmlentities(strip_tags(trim($_POST["thn"])));
24
25         // siapkan variabel untuk menampung pesan error
26         $pesan_error="";
27
28         // cek apakah "tgl" < 10
29         if ($tgl <= 10) {
30             $pesan_error .= "Maaf, tanggal harus > 10 <br>";
31         }
32
33         // jika tidak ada error, tampilkan isi form
34         if ($pesan_error === "") {
35             echo "Form berhasil di proses <br>";
36             echo "Tanggal dikirim : $tgl - $bln - $thn <br>";
37             die();
38         }
39     }
40     else {
41         $pesan_error = "";
42         $tgl=1; $bln="1"; $thn=2016;
43     }
44 ?>
```



```
91         }
92         echo "$i </option>";
93     }
94     ?>
95   </select>
96 </p>
97   <input type="submit" name="submit" value="Proses Data" >
98 </form>
99 </body>
100</html>
```

---

Untuk contoh praktik, silahkan pilih isian tanggal antara 1-10, lalu **submit**. Pesan kesalahan akan tampil, dan pilihan form tidak akan berubah (walaupun form sebenarnya sudah di-reload). Silahkan anda pelajari sebentar bagaimana kode ini bisa berjalan. Konsep tentang perulangan dan kondisi if sangat diperlukan disini.

## 20.16 Repopulate Isian Form Checkbox

Materi kali ini sama seperti sebelumnya, yakni cara me-*repopulate* isian form. Namun yang akan kita bahas adalah untuk isian form yang berupa **checkbox**. Langsung saja lihat contoh kode programnya:

index.php

```
1 <?php
2   // cek apakah form telah di submit
3   if (isset($_POST["submit"])) {
4     // form telah disubmit, proses data
5
6     // ambil nilai form
7     if (isset($_POST["dvd"])) {
8       $tambahan_dvd = "+ DVD eBook";
9     }
10    else {
11      $tambahan_dvd = "";
12    }
13
14    if (isset($_POST["kado"])) {
15      $tambahan_kado = "+ Bungkus Kado";
16    }
17    else {
18      $tambahan_kado = "";
19    }
20
21    // siapkan variabel untuk menampung pesan error
```

```
22     $pesan_error="";
23
24     // validasi untuk test
25     if (true) {
26         $pesan_error .= "Maaaf, ini hanya percobaan <br>";
27     }
28
29     // jika tidak ada error, tampilkan isi form
30     if ($pesan_error === "") {
31         echo "Form berhasil di proses <br>";
32         echo "Tambah : $tambahan_dvd $tambahan_kado <br>";
33         die();
34     }
35 }
36 else {
37     $pesan_error = "";
38 }
39 ?>
40
41 <!DOCTYPE html>
42 <html>
43 <head>
44     <meta charset="UTF-8">
45     <title>Belajar PHP</title>
46 </head>
47 <body>
48 <h1>Pemrosesan Form</h1>
49     <?php echo $pesan_error; ?>
50     <form action="index.php" method="post">
51         <p>
52             <label>Tambah : </label>
53             <input type="checkbox" name="dvd" value="DVD eBook" id="dvd">
54             <label for="dvd"> DVD eBook</label>
55             <input type="checkbox" name="kado" value="Bungkus Kado" id="kado">
56             <label for="kado"> Bungkus Kado</label>
57         </p>
58         <input type="submit" name="submit" value="Proses Data" >
59     </form>
60 </body>
61 </html>
```

Disini saya membuat 2 buah isian form **checkbox**. Berbeda dengan tag `<input type="text">` maupun `<select>`, nilai **checkbox** baru akan dikirim apabila sudah diceklist. Jika tidak dipilih, tidak ada nilai apapun yang dikirim.

Untuk mengatasi hal ini, saya harus melakukan pemeriksaan terlebih dahulu sebelum melakukan

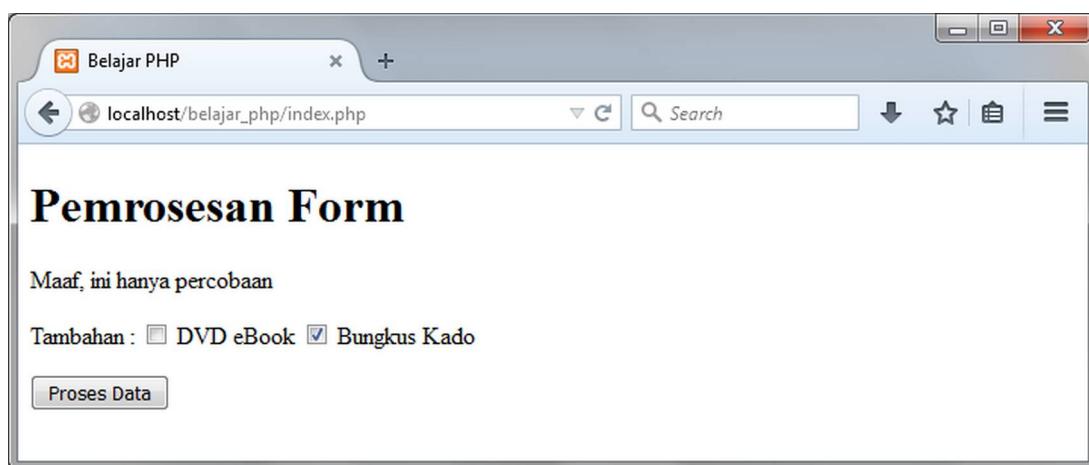
proses apapun. Inilah dilakukan dengan baris perintah `if (isset($_POST["dvd"]))` dan `if (isset($_POST["kado"]))`.

Variabel `$tambahan_dvd` dan `$tambahan_kado` saya buat untuk ditampilkan setelah form sukses diproses. Ini tidak berhubungan dengan *re-populate* form.

Khusus untuk percobaan ini, saya membuat kasus validasi yang selalu salah:

```
1 // validasi untuk test
2 if (true) {
3     $pesan_error .= "Maaf, ini hanya percobaan <br>";
4 }
```

Artinya, apapun nilai form yang dikirim, pesan kesalahan akan selalu tampil. Ini saya lakukan semata-mata untuk menguji kode program kita.



Gambar: pesan kesalahan akan tetap tampil walaupun kedua checkbox sudah dipilih

Kembali ke tujuan awal, bagaimana cara agar ketika pesan kesalahan tampil (halaman *reload*), pilihan **checkbox** tetap terpilih seperti semula?

Agar sebuah isian **checkbox** bisa terpilih, kita harus menambahkan atribut “**checked**” ke dalam tag `<input>`, seperti contoh berikut:

```
1 <input type="checkbox" name="dvd" value="DVD eBook" id="dvd" checked>
2 <label for="dvd"> DVD eBook</label>
```

Untuk membuat *re-populate checkbox*, caranya hampir sama seperti *re-populate tag <select>*. Berikut kode program yang saya gunakan:

## index.php

```
1 <?php
2
3 // cek apakah form telah di submit
4 if (isset($_POST["submit"])) {
5     // form telah disubmit, proses data
6
7     // ambil nilai form
8     if (isset($_POST["dvd"])) {
9         $checked_dvd = "checked";
10        $tambahan_dvd = "+ DVD eBook";
11    }
12    else {
13        $checked_dvd = "";
14        $tambahan_dvd = "";
15    }
16
17    if (isset($_POST["kado"])) {
18        $checked_kado = "checked";
19        $tambahan_kado = "+ Bungkus Kado";
20    }
21    else {
22        $checked_kado = "";
23        $tambahan_kado = "";
24    }
25
26 // siapkan variabel untuk menampung pesan error
27 $pesan_error="";
28
29 // validasi untuk test
30 if (true) {
31     $pesan_error .= "Maaf, ini hanya percobaan <br>";
32 }
33
34 // jika tidak ada error, tampilkan isi form
35 if ($pesan_error === "") {
36     echo "Form berhasil di proses <br>";
37     echo "Tambahan : $tambahan_dvd $tambahan_kado <br>";
38     die();
39 }
40 }
41 else {
42     $pesan_error = "";
43     $checked_dvd = ""; $checked_kado = "";
44 }
```

```
45  ?>
46
47  <!DOCTYPE html>
48  <html>
49  <head>
50      <meta charset="UTF-8">
51      <title>Belajar PHP</title>
52  </head>
53  <body>
54      <h1>Pemrosesan Form</h1>
55      <?php echo $pesan_error; ?>
56      <form action="index.php" method="post">
57          <p>
58              <label>Tambah : </label>
59              <input type="checkbox" name="dvd" value="DVD eBook" id="dvd"
60                  <?php echo $checked_dvd; ?>
61              <label for="dvd"> DVD eBook</label>
62              <input type="checkbox" name="kado" value="Bungkus Kado" id="kado"
63                  <?php echo $checked_kado; ?>
64              <label for="kado"> Bungkus Kado</label>
65          </p>
66          <input type="submit" name="submit" value="Proses Data" >
67      </form>
68  </body>
69  </html>
```

---

Ketika form di proses, saya mempersiapkan 2 variabel: `$checked_dvd` dan `$checked_kado`. Kedua variabel ini akan berisi nilai “checked” apabila checkbox tersebut dipilih.

Sebagai contoh, jika saya memilih checkbox DVD eBook, kondisi `if (isset($_POST["dvd"]))` akan menghasilkan `TRUE`, sehingga variabel `$checked_dvd` berisi string “checked”.

Kemudian, variabel ini di echo di dalam kode:

```
1  <input type="checkbox" name="dvd" value="DVD eBook" id="dvd"
2  <?php echo $checked_dvd; ?>
```

Ini sama artinya dengan:

```
1  <input type="checkbox" name="dvd" value="DVD eBook" id="dvd"
2  checked >
```

Dengan demikian, pilihan checkbox akan langsung terpilih pada saat pesan error ditampilkan.

## 20.17 Repopulate Isian Form Radio Button

Setelah form *select* dan *checkbox*, berikutnya kita akan membahas cara repopulate pilihan **radio button**.

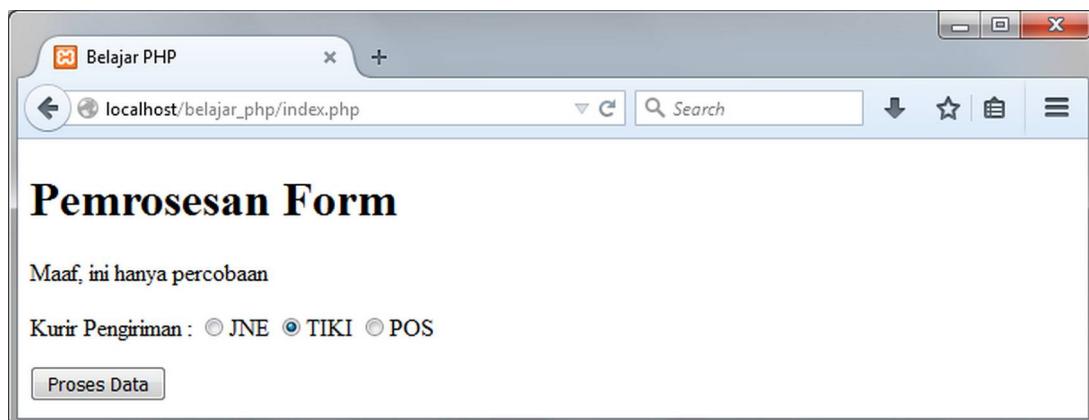
Pada dasarnya, teknik yang digunakan sama seperti sebelumnya. Jika isi form radio button ini ditulis manual, kita perlu menggunakan struktur case. Jika isinya di generate oleh PHP, kita bisa menggunakan kondisi if.

Langsung saja kode program lengkap dari contoh kasus *re-populate* form radio button:

index.php

```
1 <?php
2     // cek apakah form telah di submit
3     if (isset($_POST["submit"])) {
4         // form telah disubmit, proses data
5
6         // ambil nilai form
7         $kurir = htmlentities(strip_tags(trim($_POST["kurir"])));
8
9         // siapkan variabel untuk menampung pesan error
10        $pesan_error="";
11
12        // validasi untuk test
13        if (true) {
14            $pesan_error .= "Maaf, ini hanya percobaan <br>";
15        }
16
17        // siapkan variabel untuk re-populate pilihan kurir
18        $checked_jne="";$checked_tiki="";$checked_pos="";
19
20        switch($kurir) {
21            case "JNE" : $checked_jne = "checked"; break;
22            case "TIKI" : $checked_tiki = "checked"; break;
23            case "POS" : $checked_pos = "checked"; break;
24        }
25
26        // jika tidak ada error, tampilkan isi form
27        if ($pesan_error === "") {
28            echo "Form berhasil di proses <br>";
29            echo "Tambah : $tambahan_dvd $tambahan_kado <br>";
30            die();
31        }
32    }
33    else {
34        $pesan_error = "";
```

```
35     $checked_jne="checked";
36     $checked_tiki=""; $checked_pos="";
37 }
38 ?>
39 <!DOCTYPE html>
40 <html>
41 <head>
42   <meta charset="UTF-8">
43   <title>Belajar PHP</title>
44 </head>
45 <body>
46 <h1>Pemrosesan Form</h1>
47   <?php echo $pesan_error; ?>
48   <form action="index.php" method="post">
49     <p>
50       Kurir Pengiriman :
51       <label><input type="radio" name="kurir" value="JNE"
52         <?php echo $checked_jne ?>>JNE</label>
53       <label><input type="radio" name="kurir" value="TIKI"
54         <?php echo $checked_tiki ?>>TIKI</label>
55       <label><input type="radio" name="kurir" value="POS"
56         <?php echo $checked_pos ?>>POS</label>
57     </p>
58     <input type="submit" name="submit" value="Proses Data" >
59   </form>
60 </body>
61 </html>
```



Gambar: Pesan error untuk radio button + repopulate form

Disini saya membuat radio button dengan 3 pilihan kurir: JNE, TIKI dan POS. Berbeda dengan *checkbox*, ketiga isian ini memiliki nama yang sama, `name="kurir"`. Variabel `$_POST["kurir"]` akan berisi nilai kurir yang dipilih.

Untuk memilih kembali tombol radio button, saya menggunakan struktur `case`. Cara penggu-

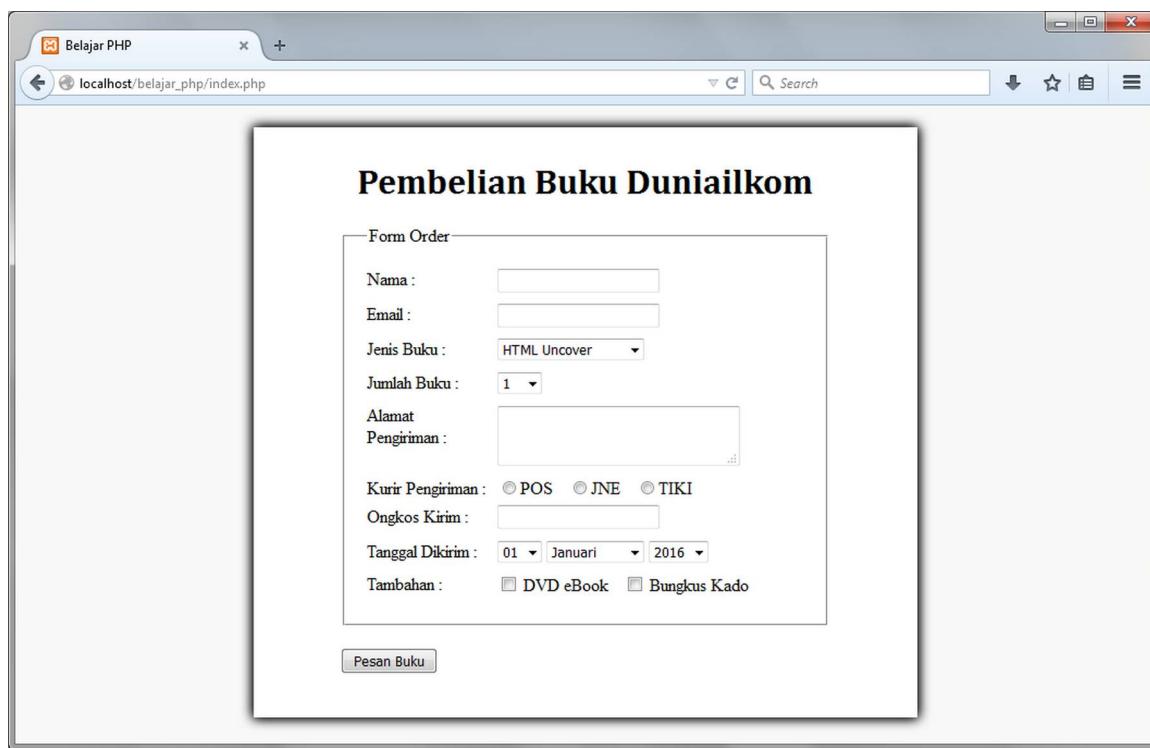
naannya sama seperti memproses tag <select> yang telah kita bahas sebelumnya. Salah satu dari variabel \$checked\_jne, \$checked\_tiki, atau \$checked\_pos akan berisi string "checked". Variabel ini kemudian ditampilkan ke dalam tag <input type="radio">.

Silahkan anda coba memilih salah satu pilihan yang ada, lalu submit. Halaman akan reload, namun pilihan form sebelumnya tetap terpilih.

## 20.18 Case Study: Sebuah Form Utuh

Bab tentang pemrosesan form ini saya buat cukup detail sehingga menjadi sangat panjang. Sebagai penutup dan untuk memastikan pemahaman anda tentang cara memproses form, saya akan membuat sebuah *case study* atau contoh kasus yang merangkum seluruh bahasan yang telah kita pelajari.

Saya membuat sebuah form untuk pemesanan buku duniaIlkom. Form ini terdiri dari berbagai isian, seperti gambar berikut:



The screenshot shows a web browser window with the title 'Belajar PHP'. The address bar shows 'localhost/belajar\_php/index.php'. The main content is a form titled 'Pembelian Buku DuniaIlkom'. The form is labeled 'Form Order' and contains the following fields:

- Nama :
- Email :
- Jenis Buku :
- Jumlah Buku :
- Alamat Pengiriman :
- Kurir Pengiriman :  POS  JNE  TIKI
- Ongkos Kirim :
- Tanggal Dikirim :
- Tambahan :  DVD eBook  Bungkus Kado

At the bottom is a 'Pesan Buku' button.

Gambar: Form pemesanan buku DuniaIlkom

Latihan pertama, dapatkah anda membuat tampilan form yang sama? Di dalam halaman ini terdapat 9 isian form. Saya menggunakan sedikit kode CSS untuk merapikan tampilan form, tapi ini tidak harus, yang penting anda bisa merancang formnya.

Berikut rincian untuk masing-masing isian form:

- Nama: menggunakan atribut name="nama".
- Email: menggunakan atribut name="email".

- Jenis Buku: dibuat dengan tag `<select>` dengan 5 isian: HTML Uncover, CSS Uncover, PHP Uncover, JavaScript Uncover, dan MySQL Uncover. Menggunakan atribut `name="buku"`.
- Jumlah Buku: dibuat dengan tag `<select>` dengan 10 isian, yakni angka 0-10. Menggunakan atribut `name="jumlah"`.
- Alamat Pengiriman: dibuat dengan `<textarea>`, menggunakan atribut `name="alamat"`.
- Kurir Pengiriman: dibuat menggunakan radion button dengan 3 pilihan: JNE, TIKI, dan POS, menggunakan atribut `name="kurir"`.
- Ongkos Kirim: menggunakan atribut `name="ongkir"`.
- Tanggal Dikirim: terdiri dari 3 pilihan list. Untuk tanggal menggunakan atribut `name="tg1"`, nilai yang dapat dipilih antara 1-31. Untuk bulan menggunakan atribut `name="bln"`, nilai yang dapat dipilih adalah nama-nama bulan dalam Bahasa Indonesia. Untuk tahun menggunakan atribut `name="thn"`, nilai yang dapat dilihat antara 2016-2020.
- Tambahan: terdiri dari 2 pilihan checkbox. Menggunakan atribut `name="dvd"` dan `name="kado"`.

Berikut kode HTML, CSS, dan PHP yang saya gunakan untuk membuat form tersebut:

#### index.php

---

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar PHP</title>
6   <style>
7     body {
8       background-color: #F8F8F8;
9     }
10    div.container {
11      width: 450px;
12      padding: 10px 80px 30px;
13      background-color: white;
14      margin: 20px auto;
15      box-shadow: 1px 0px 10px, -1px 0px 10px ;
16    }
17    h1 {
18      text-align: center;
19      font-family: Cambria, "Times New Roman", serif;
20    }
21    p {
22      margin:0;
23    }
24    fieldset {
25      padding:20px;
26    }
27    input, select, textarea {
28      margin-bottom:10px;
```

```
29      }
30      label {
31          width:110px;
32          float:left;
33          margin-right:10px;
34      }
35      .radio {
36          width:55px;
37          float:none;
38      }
39      label[for="dvd"], label[for="kado"] {
40          float: none;
41      }
42  </style>
43 </head>
44 <body>
45
46 <div class="container">
47 <h1>Pembelian Buku DuniaIlkom</h1>
48
49 <form action="index.php" method="post">
50 <fieldset>
51 <legend>Form Order</legend>
52 <p>
53     <label for="nama">Nama : </label>
54     <input type="text" name="nama" id="nama">
55 </p>
56 <p>
57     <label for="email">Email : </label>
58     <input type="text" name="email" id="email">
59 </p>
60 <p>
61     <label for="buku" >Jenis Buku : </label>
62     <select name="buku" id="buku">
63         <option value="HTML Uncover">HTML Uncover</option>
64         <option value="CSS Uncover">CSS Uncover</option>
65         <option value="PHP Uncover">PHP Uncover</option>
66         <option value="JavaScript Uncover">JavaScript Uncover</option>
67         <option value="MySQL Uncover">MySQL Uncover</option>
68     </select>
69 </p>
70 <p>
71     <label for="jumlah">Jumlah Buku : </label>
72     <select name="jumlah" id="jumlah">
73         <?php
74             for ($i = 1; $i <= 10; $i++) {
```

```
75         echo "<option value = $i >";
76         echo $i;
77         echo "</option>";
78     }
79     ?>
80     </select>
81 </p>
82 <p>
83     <label for="alamat">Alamat Pengiriman : </label>
84     <textarea name="alamat" cols="25" name="alamat"></textarea>
85 </p>
86     <p>
87     <label>Kurir Pengiriman :</label>
88     <label class="radio"><input type="radio" name="kurir" value="pos">POS</la\
89 bel>
90     <label class="radio"><input type="radio" name="kurir" value="jne">JNE</la\
91 bel>
92     <label class="radio"><input type="radio" name="kurir" value="tiki">TIKI</\
93 label>
94     </p>
95     <p >
96         <label for="ongkir">Ongkos Kirim : </label>
97         <input type="text" name="ongkir" id="ongkir">
98     </p>
99     <p>
100        <label for="tgl" >Tanggal Dikirim : </label>
101        <select name="tgl" id="tgl">
102            <?php
103            for ($i = 1; $i <= 31; $i++) {
104                echo "<option value = $i >";
105                echo str_pad($i,2,"0",STR_PAD_LEFT);
106                echo "</option>";
107            }
108            ?>
109        </select>
110        <select name="bln">
111            <option value=1>Januari</option>
112            <option value=2>Februari</option>
113            <option value=3>Maret</option>
114            <option value=4>April</option>
115            <option value=5>Mei</option>
116            <option value=6>Juni</option>
117            <option value=7>Juli</option>
118            <option value=8>Agustus</option>
119            <option value=9>September</option>
120            <option value=10>Oktober</option>
```

```
121      <option value=11>Nopember</option>
122      <option value=12>Desember</option>
123  </select>
124  <select name="thn">
125      <?php
126      for ($i = 2016; $i <= 2020; $i++) {
127          echo "<option value = $i > $i </option>";
128      }
129      ?>
130  </select>
131  </p>
132  <p>
133      <label>Tambah : </label>
134      <input type="checkbox" name="dvd" value="ok" id="dvd">
135      <label for="dvd"> DVD eBook</label>
136      <input type="checkbox" name="kado" value="ok" id="kado">
137      <label for="kado"> Bungkus Kado</label>
138  </p>
139 </fieldset>
140  <br>
141  <p>
142      <input type="submit" name="submit" value="Pesanan Buku">
143  </p>
144 </form>
145
146 </div>
147
148 </body>
149 </html>
```

---

Silahkan anda luangkan waktu untuk mempelajari kode form diatas, terutama penggunaan atribut **name** dari setiap isian form. Jika bingung dengan tag HTML seperti `<label>`, `<fieldset>` atau `<legend>`, silahkan pelajari sebentar di buku **HTML Uncover**. Penggunaan ketiga tag ini telah saya bahas dengan cukup lengkap. Terkait kode CSS juga bisa dipelajari di buku **CSS Uncover** DuniaIlkom (promosi untuk yang belum beli bukunya, hehe..)

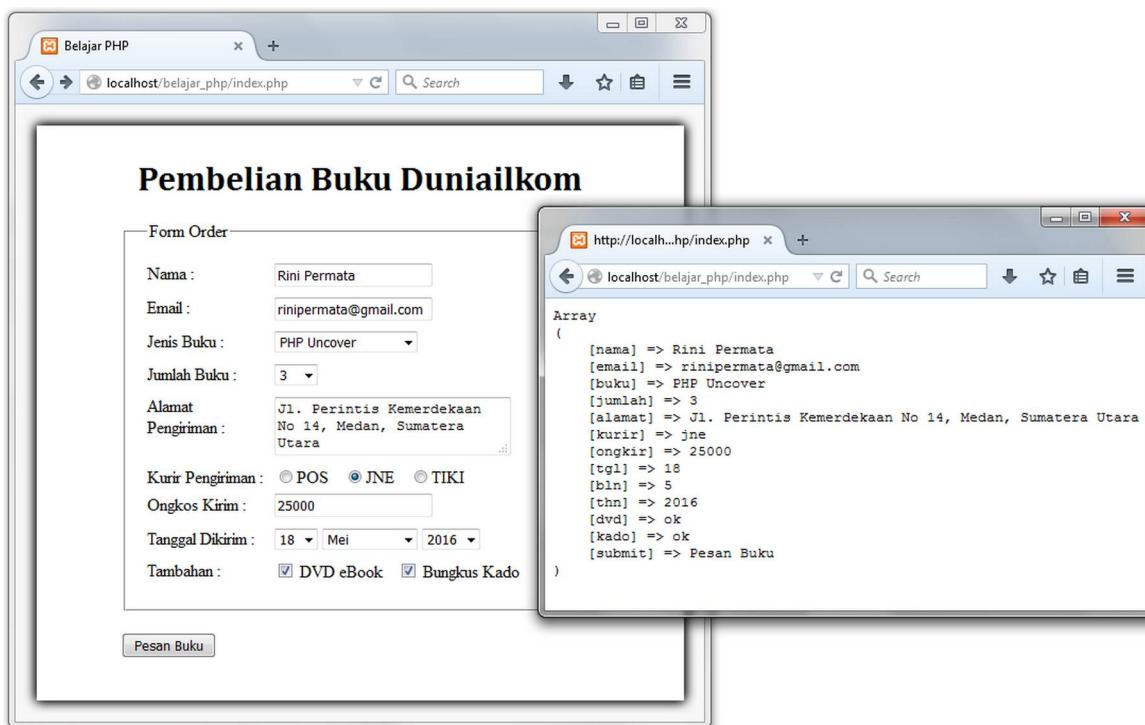
Baik, sebelum masuk ke dalam kode PHP untuk memproses nilai form, sangat disarankan kita melakukan pengecekan terlebih dahulu. Apakah seluruh isian form ini sudah terkirim seperti yang diharapkan, atau terdapat error. Caranya dengan menggunakan fungsi `var_dump($_POST)` atau `print_r($_POST)`. Silahkan tambahkan baris berikut di bagian paling atas halaman, yakni sebelum tag `<!DOCTYPE html>`:

```

1  <?php
2  // cek apakah form telah di submit
3  if (isset($_POST["submit"])) {
4      // form telah disubmit, cek seluruh data
5      echo "<pre>";
6      print_r($_POST);
7      echo "</pre>";
8      die();
9  }
10 ?>

```

Jika semua atribut name telah sesuai, seharusnya anda akan mendapat hasil seperti ini:



Gambar: Contoh testing isian form pemesanan buku DuniaIlkom

```

1  Array
2  (
3      [nama] => Rini Permata
4      [email] => rinipermata@gmail.com
5      [buku] => PHP Uncover
6      [jumlah] => 3
7      [alamat] => Jl. Perintis Kemerdekaan No 14, Medan, Sumatera Utara
8      [kurir] => jne
9      [ongkir] => 25000
10     [tgl] => 18
11     [bln] => 5
12     [thn] => 2016

```

```
13     [dvd] => ok
14     [kado] => ok
15     [submit] => Pesan Buku
16 )
```

Kalau boleh, saya mengharuskan anda untuk melakukan hal ini setiap kali membuat form. Ini agar kesalahan form dapat diketahui sejak awal. Bayangkan jika ternyata ada salah satu isian form yang tidak terkirim, namun kita sudah membuat ratusan baris kode PHP untuk memproses isian tersebut. Jika sudah seperti ini akan sangat sulit mencari asal penyebab error yang muncul.

Setelah memastikan seluruh isian form terkirim sempurna, langkah berikutnya adalah membuat halaman yang akan menampilkan isi form. Untuk merancang halaman ini, saya juga menggunakan data “dummy” agar lebih mudah dibuat.

Saya membuat sebuah halaman *pembelian.php*, dengan kode sebagai berikut:

### pembelian.php

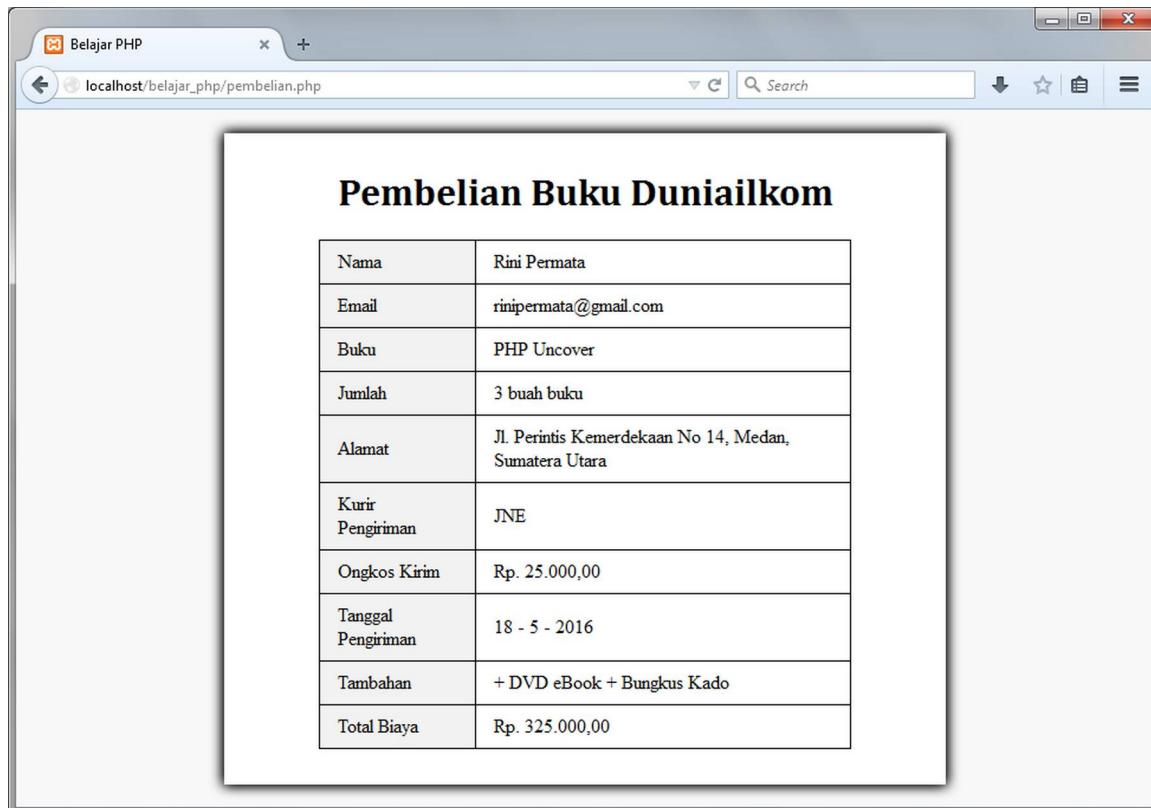
---

```
1 <?php
2     $nama = "Rini Permata";
3     $email = "rinipermata@gmail.com";
4     $buku = "PHP Uncover";
5     $jumlah=3;
6     $alamat="Jl. Perintis Kemerdekaan No 14, Medan, Sumatera Utara";
7     $kurir="JNE";
8     $ongkir="25000";
9     $tgl=18;$b1n="5";$thn=2016;
10    $tambahan_dvd = "+ DVD eBook";
11    $tambahan_kado = "+ Bungkus Kado";
12 ?>
13 <!DOCTYPE html>
14 <html>
15 <head>
16     <meta charset="UTF-8">
17     <title>Belajar PHP</title>
18     <style>
19         body {
20             background-color: #F8F8F8;
21         }
22         div.container {
23             width: 450px;
24             padding: 10px 80px 30px;
25             background-color: white;
26             margin: 20px auto;
27             box-shadow: 1px 0px 10px, -1px 0px 10px ;
28         }
29         h1 {
30             text-align: center;
```

```
31     font-family: Cambria, "Times New Roman", serif;
32 }
33 table {
34     border-collapse: collapse;
35     border-spacing: 0;
36     border: 1px black solid;
37     width: 100%
38 }
39 th, td {
40     padding: 8px 15px;
41     border: 1px black solid;
42 }
43 td:first-child {
44     background-color: #F2F2F2;
45 }
46 </style>
47 </head>
48 <body>
49
50 <div class="container">
51 <h1>Pembelian Buku DuniaIlkom</h1>
52 <table>
53 <tr>
54     <td>Nama</td>
55     <td><?php echo $nama; ?></td>
56 </tr>
57 <tr>
58     <td>Email</td>
59     <td><?php echo $email; ?></td>
60 </tr>
61 <tr>
62     <td>Buku</td>
63     <td><?php echo $buku; ?></td>
64 </tr>
65 <tr>
66     <td>Jumlah</td>
67     <td><?php echo $jumlah." buah buku"; ?></td>
68 </tr>
69 <tr>
70     <td>Alamat</td>
71     <td><?php echo $alamat; ?></td>
72 </tr>
73 <tr>
74     <td>Kurir Pengiriman</td>
75     <td><?php echo $kurir; ?></td>
76 </tr>
```

```
77 <tr>
78   <td>Ongkos Kirim</td>
79   <td><?php echo "Rp. ".number_format($ongkir,2,",","."); ?></td>
80 </tr>
81 <tr>
82   <td>Tanggal Pengiriman</td>
83   <td><?php echo "$tgl - $bln - $thn"; ?></td>
84 </tr>
85 <tr>
86   <td>Tambahkan</td>
87   <td><?php echo "$tambahan_dvd $tambahan_kado"; ?></td>
88 </tr>
89 <tr>
90   <td>Total Biaya</td>
91   <td>
92     <?php
93       $total=($jumlah*100000)+$ongkir;
94       echo "Rp. ".number_format($total,2,",",".");;
95     ?>
96   </td>
97 </tr>
98 </table>
99 </body>
100 </html>
```

---



Gambar: Tampilan akhir form menggunakan data “dummy”

Di awal kode program, terdapat 12 variabel yang merupakan data “dummy”. Saya sebut data “dummy” karena seharusnya data ini berasal dari form, bukan di ketik langsung.

Pada halaman *pembelian.php* saya menampilkan seluruh isi form, di baris terakhir ditambahkan kode program untuk menghitung *Total Biaya* dengan rumus = jumlah buku \* 100.000, lalu ditambah dengan ongkos kirim. Proses perhitungan ini hanya sekedar tambahan ‘pemanis’ form.

Untuk menampilkan nominal ongkos kirim dan total biaya, saya menggunakan fungsi *number\_format()* agar tampilan angka lebih rapi.

Sampai disini, saya memiliki 2 buah halaman: *index.php* yang berisi form, dan *pembelian.php* yang akan menampilkan hasil form. Agar proses validasi dan menampilkan error lebih mudah dirancang, saya memutuskan menggunakan 1 halaman untuk proses dan menampilkan form. Bagaimana cara menggabungkan dua halaman ini? gunakan fungsi *include()*.

Triknya adalah, ketika form valid dan data sudah sesuai, jalankan perintah *include("pembelian.php")*.

Silahkan hapus seluruh variabel dummy dari halaman *pembelian.php*, kemudian di baris paling atas halaman *index.php*, yakni tempat dimana fungsi *print\_r()* sebelumnya berada, ubah menjadi:

```

1  <?php
2  // cek apakah form telah di submit
3  if (isset($_POST["submit"])) {
4      // form telah disubmit, ambil nilai form
5      $nama = htmlentities(strip_tags(trim($_POST["nama"])));
6      $email = htmlentities(strip_tags(trim($_POST["email"])));
7      $buku = htmlentities(strip_tags(trim($_POST["buku"])));
8      $jumlah = htmlentities(strip_tags(trim($_POST["jumlah"])));
9      $alamat = htmlentities(strip_tags(trim($_POST["alamat"])));
10     $kurir = htmlentities(strip_tags(trim($_POST["kurir"])));
11     $ongkir = htmlentities(strip_tags(trim($_POST["ongkir"])));
12     $tgl = htmlentities(strip_tags(trim($_POST["tgl"])));
13     $bln = htmlentities(strip_tags(trim($_POST["bln"])));
14     $thn = htmlentities(strip_tags(trim($_POST["thn"])));
15
16     // untuk check box
17     $tambahan_dvd = "+ DVD eBook";
18     $tambahan_kado = "+ Bungkus Kado";
19
20     // panggil halaman pembelian untuk menampilkan form
21     include("pembelian.php");
22     die();
23 }
24 ?>

```

Ini artinya ketika form di submit, ambil seluruh isian form, kemudian gunakan fungsi `include()` untuk menampilkan hasil form. Fungsi `die()` saya tambahkan untuk memastikan kode PHP berhenti begitu selesai menampilkan halaman `pembelian.php`. Karena kita belum membuat kode validasi apapun, anda harus mengisi seluruh form agar tidak tampil error.

Sekarang, langkah terakhir yang sekaligus yang paling rumit, adalah membuat kode validasi dan repopulate form. Saya akan menampilkan kode akhir yang saya rancang. Berikut kode pembuatan form serta validasi “*Pembelian Buku DuniaIlkom*”:

#### index.php

---

```

1  <?php
2  // siapkan array untuk nama bulan
3  $arr_bln = array( "1"=>"Januari",
4                    "2"=>"Februari",
5                    "3"=>"Maret",
6                    "4"=>"April",
7                    "5"=>"Mei",
8                    "6"=>"Juni",
9                    "7"=>"Juli",
10                   "8"=>"Agustus",
11                   "9"=>"September",

```

```
12          "10"=> "Oktober",
13          "11"=> "Nopember",
14          "12"=> "Desember" );
15
16 // cek apakah form telah di submit
17 if (isset($_POST["submit"])) {
18     // form telah disubmit, proses data
19
20     // ambil nilai form kecuali checkbox
21     $nama = htmlentities(strip_tags(trim($_POST["nama"])));
22     $email = htmlentities(strip_tags(trim($_POST["email"])));
23     $buku = htmlentities(strip_tags(trim($_POST["buku"])));
24     $jumlah = htmlentities(strip_tags(trim($_POST["jumlah"])));
25     $alamat = htmlentities(strip_tags(trim($_POST["alamat"])));
26     $kurir = htmlentities(strip_tags(trim($_POST["kurir"])));
27     $ongkir = htmlentities(strip_tags(trim($_POST["ongkir"])));
28     $tgl = htmlentities(strip_tags(trim($_POST["tgl"])));
29     $bln = htmlentities(strip_tags(trim($_POST["bln"])));
30     $thn = htmlentities(strip_tags(trim($_POST["thn"])));
31
32     // siapkan variabel untuk menampung pesan error
33     $pesan_error="";
34
35     // cek apakah "nama" sudah diisi atau tidak
36     if (empty($nama)) {
37         $pesan_error .= "Nama belum diisi <br>";
38     }
39
40     // cek apakah "email" sudah diisi atau tidak
41     if (empty($email)) {
42         $pesan_error .= "Email belum diisi <br>";
43     }
44     // email harus sesuai dengan format
45     elseif (!preg_match("/.+@.+\.+/", $email) ) {
46         $pesan_error .= "Format email tidak sesuai <br>";
47     }
48
49     // siapkan variabel untuk menggenerate pilihan buku
50     $select_html=""; $select_css=""; $select_php="";
51     $select_javascript=""; $select_mysql="";
52
53     switch($buku) {
54         case "HTML Uncover" : $select_html      = "selected"; break;
55         case "CSS Uncover"  : $select_css       = "selected"; break;
56         case "PHP Uncover"  : $select_php        = "selected"; break;
57         case "JavaScript Uncover" : $select_javascript = "selected"; break;
```

```
58     case "MySQL Uncover"      : $select_mysql      = "selected"; break;
59 }
60
61 // cek apakah "alamat" sudah diisi atau tidak
62 if (empty($alamat)) {
63     $pesan_error .= "Alamat Pengiriman belum diisi <br>";
64 }
65
66 // siapkan variabel untuk menggenerate pilihan kurir
67 $checked_jne="";$checked_tiki="";$checked_pos="";
68
69 switch($kurir) {
70     case "JNE"  : $checked_jne = "checked"; break;
71     case "TIKI" : $checked_tiki = "checked"; break;
72     case "POS"  : $checked_pos = "checked"; break;
73 }
74
75 // ongkos kirim harus berupa angka dan kelipatan 1000
76 if (!is_numeric($ongkir) OR ($ongkir <=0) OR (($ongkir % 1000) != 0)) {
77     $pesan_error .= "Ongkos kirim harus dalam kelipatan 1000";
78 }
79
80 if (isset($_POST["dvd"])) {
81     $checked_dvd = "checked";
82     $tambahan_dvd = "+ DVD eBook";
83 }
84 else {
85     $checked_dvd = "";
86     $tambahan_dvd = "";
87 }
88
89 if (isset($_POST["kado"])) {
90     $checked_kado = "checked";
91     $tambahan_kado = "+ Bungkus Kado";
92 }
93 else {
94     $checked_kado = "";
95     $tambahan_kado = "";
96 }
97
98 // jika tidak ada error, tampilkan isi form
99 if ($pesan_error === "") {
100     include("pembelian.php"); // hasil form ditampilkan oleh halaman ini
101     die();
102 }
103 }
```

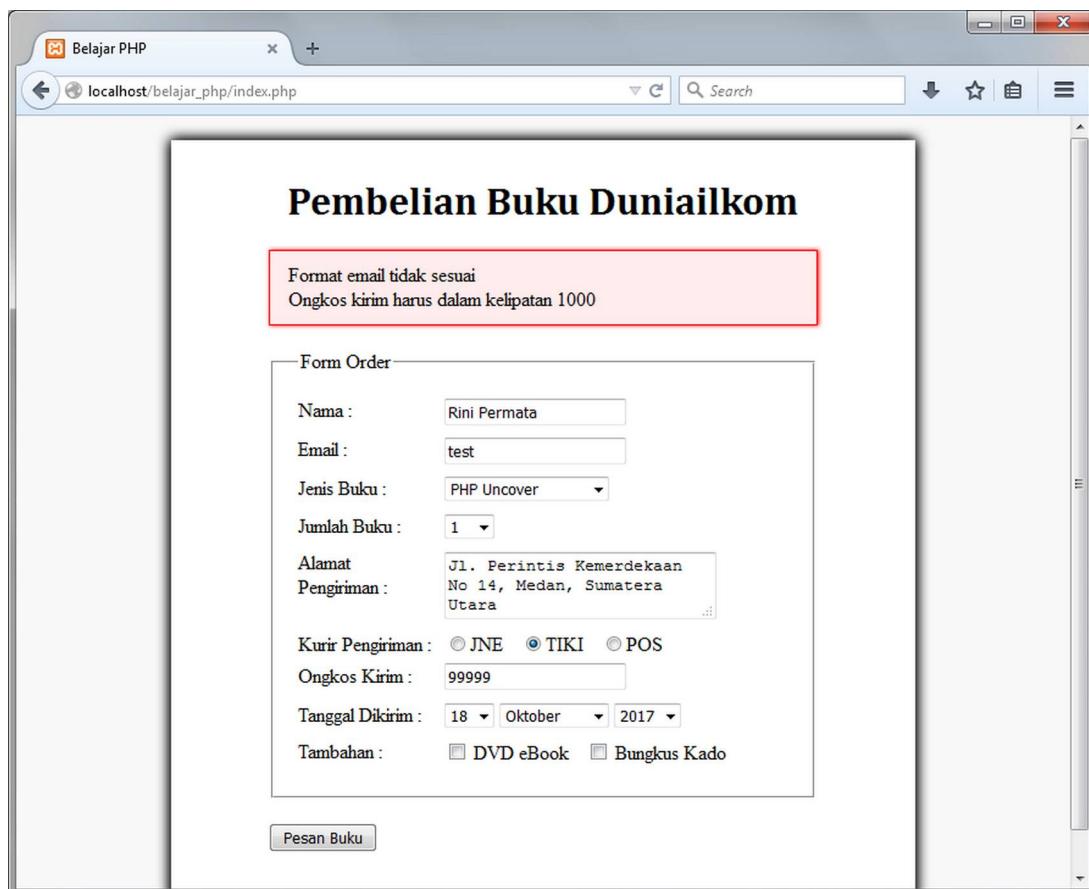
```
104 else {
105     // form belum disubmit atau halaman ini tampil untuk pertama kali
106     // berikan nilai awal untuk semua isian form
107     $pesan_error = "";
108     $nama = "";
109     $email = "";
110     $select_html="selected";
111     $select_css=""; $select_php="";
112     $select_javascript=""; $select_mysql="";
113     $jumlah=0;
114     $alamat="";
115     $checked_jne="checked";
116     $checked_tiki=""; $checked_pos="";
117     $ongkir="";
118     $tgl=1;$bln="1";$thn=2016;
119     $checked_dvd=""; $checked_kado="";
120     $tambahan_kado=""; $tambahan_dvd="";
121 }
122 ?>
123 <!DOCTYPE html>
124 <html>
125 <head>
126     <meta charset="UTF-8">
127     <title>Belajar PHP</title>
128     <style>
129         body {
130             background-color: #F8F8F8;
131         }
132         div.container {
133             width: 450px;
134             padding: 10px 80px 30px;
135             background-color: white;
136             margin: 20px auto;
137             box-shadow: 1px 0px 10px, -1px 0px 10px ;
138         }
139         h1 {
140             text-align: center;
141             font-family: Cambria, "Times New Roman", serif;
142         }
143         p {
144             margin:0;
145         }
146         fieldset {
147             padding:20px;
148         }
149         input, select, textarea {
```

```
150     margin-bottom:10px;
151 }
152 label {
153     width:110px;
154     float:left;
155     margin-right:10px;
156 }
157 .radio {
158     width:55px;
159     float:none;
160 }
161 label[for="dvd"], label[for="kado"] {
162     float: none;
163 }
164 .error {
165     background-color: #FFECEC;
166     padding: 10px 15px;
167     margin: 0 0 20px 0;
168     border: 1px solid red;
169     box-shadow: 1px 0px 3px red ;
170 }
171 </style>
172 </head>
173 <body>
174
175 <div class="container">
176 <h1>Pembelian Buku DuniaIlkom</h1>
177 <?php
178     // tampilkan error jika ada
179     if ($pesan_error !== "") {
180         echo "<div class=\"error\">$pesan_error</div>";
181     }
182 ?>
183 <form action="index.php" method="post">
184 <fieldset>
185 <legend>Form Order</legend>
186 <p>
187     <label for="nama">Nama : </label>
188     <input type="text" name="nama" id="nama" value="<?php echo $nama ?>">
189 </p>
190 <p>
191     <label for="email">Email : </label>
192     <input type="text" name="email" id="email" value="<?php echo $email ?>">
193 </p>
194 <p>
195     <label for="buku" >Jenis Buku : </label>
```

```
196     <select name="buku" id="buku">
197         <option value="HTML Uncover" <?php echo $select_html ?>>
198             HTML Uncover </option>
199         <option value="CSS Uncover" <?php echo $select_css ?>>
200             CSS Uncover</option>
201         <option value="PHP Uncover" <?php echo $select_php ?>>
202             PHP Uncover</option>
203         <option value="JavaScript Uncover" <?php echo $select_javascript ?>>
204             JavaScript Uncover</option>
205         <option value="MySQL Uncover" <?php echo $select_mysql ?>>
206             MySQL Uncover</option>
207     </select>
208 </p>
209 <p>
210     <label for="jumlah">Jumlah Buku : </label>
211     <select name="jumlah" id="jumlah">
212         <?php
213             for ($i = 1; $i <= 10; $i++) {
214                 if ($i==$jumlah){
215                     echo "<option value = $i selected>";
216                 }
217                 else {
218                     echo "<option value = $i >";
219                 }
220                 echo $i;
221                 echo "</option>";
222             }
223         <?>
224     </select>
225 </p>
226 <p>
227     <label for="alamat">Alamat Pengiriman : </label>
228     <textarea name="alamat" cols="25" name="alamat"><?php echo $alamat; ?></t\
229 extarea>
230 </p>
231     <p>
232         <label>Kurir Pengiriman :</label>
233         <label class="radio"><input type="radio" name="kurir" value="JNE"
234             <?php echo $checked_jne ?>>JNE</label>
235         <label class="radio"><input type="radio" name="kurir" value="TIKI"
236             <?php echo $checked_tiki ?>>TIKI</label>
237         <label class="radio"><input type="radio" name="kurir" value="POS"
238             <?php echo $checked_pos ?>>POS</label>
239     </p>
240     <p >
241         <label for="ongkir">Ongkos Kirim : </label>
```

```
242     <input type="text" name="ongkir" id="ongkir"
243     value="<?php echo $ongkir ?>">
244 </p>
245 <p>
246     <label for="tgl" >Tanggal Dikirim : </label>
247     <select name="tgl" id="tgl">
248         <?php
249         for ($i = 1; $i <= 31; $i++) {
250             if ($i==$tgl){
251                 echo "<option value = $i selected>";
252             }
253             else {
254                 echo "<option value = $i >";
255             }
256             echo str_pad($i,2,"0",STR_PAD_LEFT);
257             echo "</option>";
258         }
259         ?
260     </select>
261     <select name="bln">
262         <?php
263         foreach ($arr_bln as $key => $value) {
264             if ($key==$bln){
265                 echo "<option value=\"$key\" selected>{$value}</option>";
266             }
267             else {
268                 echo "<option value=\"$key\">{$value}</option>";
269             }
270         }
271         ?
272     </select>
273     <select name="thn">
274         <?php
275         for ($i = 2016; $i <= 2020; $i++) {
276             if ($i==$thn){
277                 echo "<option value = $i selected>";
278             }
279             else {
280                 echo "<option value = $i >";
281             }
282             echo "$i </option>";
283         }
284         ?
285     </select>
286 </p>
287 <p>
```

```
288     <label>Tambah : </label>
289     <input type="checkbox" name="dvd" value="DVD eBook" id="dvd"
290     <?php echo $checked_dvd; ?>
291     <label for="dvd"> DVD eBook</label>
292     <input type="checkbox" name="kado" value="Bungkus Kado" id="kado"
293     <?php echo $checked_kado; ?>
294     <label for="kado"> Bungkus Kado</label>
295     </p>
296 </fieldset>
297     <br>
298     <p>
299         <input type="submit" name="submit" value="Pesanan Buku">
300     </p>
301 </form>
302
303 </div>
304
305 </body>
306 </html>
```



Gambar: Kode error tampil karena terdapat isian form yang gagal validasi

Kode diatas mungkin akan membuat anda merasa “terintimidasi” (karena sangat panjang). Saya membutuhkan waktu beberapa hari untuk merancang kode ini. Oleh karena itu, tidak ada salahnya anda meluangkan waktu setengah jam lebih untuk mempelajari apa yang dilakukan kode ini. Semuanya saya tambahkan komentar agar fungsi tiap-tiap blok program bisa dipahami. Seluruh teknik yang saya gunakan telah kita pelajari dalam bab ini.

Saya juga tidak mengharuskan anda untuk bisa memahami kode program ini dalam sekali baca. Silahkan pelajari secara perlahan. Juga sangat disarankan anda menguji validasi dari kode diatas supaya bisa memahami bagian mana dari kode program yang bertanggung jawab untuk menampilkan error tersebut.

Ketika error terjadi, seluruh isian form juga di repopulate sesuai yang telah diisi sebelumnya. Kode error ditampilkan dengan background merah yang saya buat menggunakan kode CSS (selector `.error`).

Jika *case study* ini masih kurang menantang, anda bisa membuat proses validasi lanjutan. Misalnya, jika user memilih tanggal pengiriman buku yang sudah lewat dari tanggal hari ini, (misalnya memilih tanggal 01 jan 2016), tampilkan pesan error. Atau jika user memilih buku “JavaScript Uncover” dan “MySQL Uncover”, tampilkan pesan error bahwa buku belum tersedia.

---

*Form processing* merupakan salah satu skill terpenting dan wajib di kuasai di dalam PHP, semoga materi yang ada di dalam bab ini bisa membantu anda dalam merancang form untuk berbagai keperluan.

Dalam bab berikutnya saya masih membahas form, yakni cara memproses form upload dengan PHP.

# 21. Form Upload

Dalam bab sebelumnya kita telah membahas panjang lebar tentang cara pemrosesan form menggunakan PHP. Namun masih terdapat 1 konsep lagi yang belum saya bahas, yakni pemrosesan form upload.

## 21.1 Superglobals Variable `$_FILES`

Meskipun sama-sama digunakan untuk memproses form, informasi tentang file upload di simpan ke dalam superglobals `$_FILES`, bukan `$_GET` maupun `$_POST` sebagaimana yang kita gunakan untuk memproses form biasa.

Selain itu, di kode HTML harus ditambahkan atribut `enctype="multipart/form-data"` pada tag `<form>`. Berikut contoh kode HTML untuk menampilkan kotak input upload:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar PHP</title>
6 </head>
7 <body>
8 <h1>Upload File</h1>
9   <form action="index.php" method="post" enctype="multipart/form-data">
10    <p>Nama File : <input type="text" name="nama_file"></p>
11    <p>Upload File: <input type="file" name="file_upload"></p>
12    <input type="submit" name="submit" value="Proses Upload">
13  </form>
14 </body>
15 </html>
```

Perhatikan cara penulisan tag `<form>`:

```
<form action="index.php" method="post" enctype="multipart/form-data">
```

Jika di dalam form terdapat inputan upload, kita harus menambahkan atribut `enctype="multipart/form-data"`. Apabila tidak ditulis, PHP tidak bisa memproses file upload.

Secara sederhana, atribut `enctype="multipart/form-data"` menginformasikan kepada web server bahwa nilai yang dikirim dari form tidak hanya berupa teks, tetapi *mungkin* juga terdapat file. Kata “mungkin” artinya jika di dalam form tidak terdapat file yang diupload, form akan tetap diproses seperti biasa.

Untuk form upload, kita juga harus menggunakan `method="post"`. Method `get` tidak bisa digunakan untuk pemrosesan file upload.

Pada form diatas, saya menempatkan 2 inputan: sebuah *text box* yang dibuat dengan tag `<input type="text">`, serta inputan upload yang dibuat dengan tag `<input type="file">`. Kedua tag ini memiliki atribut **name** yang akan digunakan pada saat pemrosesan nanti. Form ini ditutup dengan sebuah tombol **submit**.

Jika anda menjalankan program diatas, tag `<input type="file">` akan ditampilkan dengan sebuah tombol “**Browse...**”. Tombol ini sering kita jumpai ketika ingin melakukan upload. Tampilan tombol ditentukan sepenuhnya oleh web browser, sehingga akan sedikit berbeda ketika dijalankan pada IE, Firefox maupun Chrome.



Gambar: Tampilan form upload dengan jendela “Browse...”

Form diatas belum melakukan proses apapun karena belum ada kode program PHPnya. Seperti biasa, mari tambahkan sebuah perintah `print_r()`:

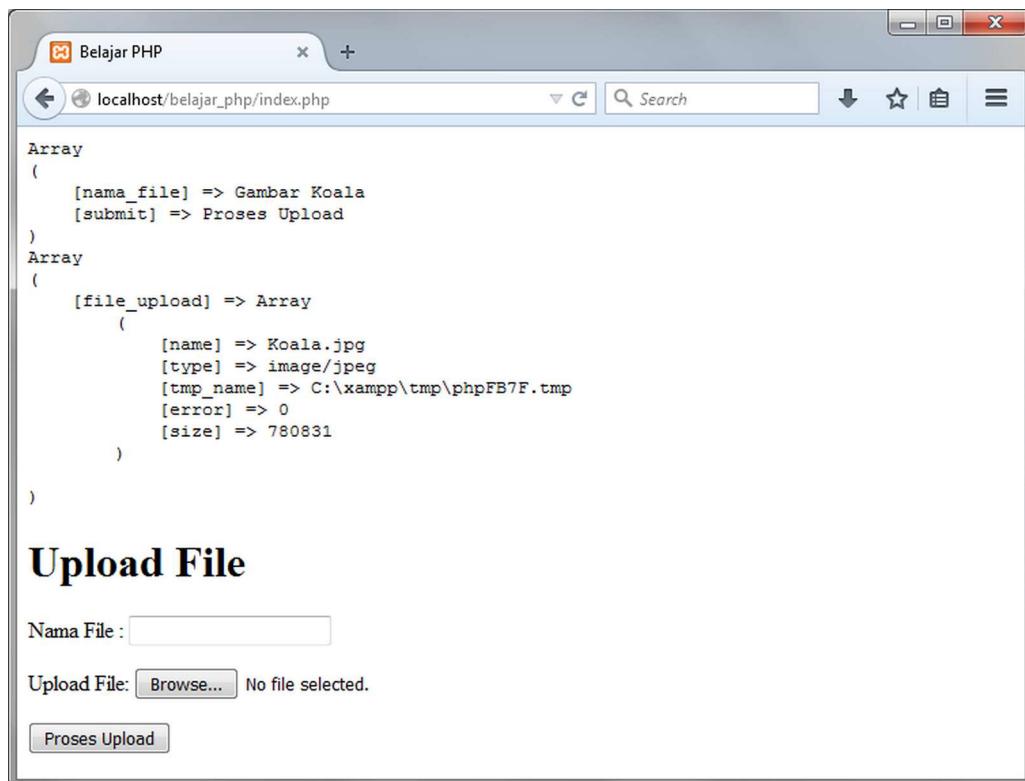
```

1 <?php
2 // cek apakah form telah di submit
3 if (isset($_POST["submit"])) {
4     // form telah disubmit, tampilkan nilai form
5     echo "<pre>";
6     print_r($_POST);
7     print_r($_FILES);
8     echo "</pre>";
9 }
10 ?>
11 <!DOCTYPE html>
12 <html>
13 <head>
14     <meta charset="UTF-8">
15     <title>Belajar PHP</title>
16 </head>

```

```
17 <body>
18 <h1>Upload File</h1>
19 <form action="index.php" method="post" enctype="multipart/form-data">
20   <p>Nama File : <input type="text" name="nama_file"></p>
21   <p>Upload File: <input type="file" name="file_upload"></p>
22   <input type="submit" name="submit" value="Proses Upload">
23 </form>
24 </body>
25 </html>
```

Form diatas diproses dalam halaman yang sama, yakni `index.php`. Saya menambahkan perintah `print_r($_POST)` dan `print_r($_FILES)` untuk melihat apa yang sebenarnya dikirim oleh form tersebut. Berikut contoh hasil yang saya dapat ketika mengupload sebuah gambar *Koala.jpg*:



Gambar: Contoh tampilan ketika form upload di proses



Apabila anda ingin menjalankan kode program diatas, harap upload file dengan ukuran 2MB ke bawah. Jika file yang di upload melebihi 2M kemungkinan besar akan terjadi error (akan kita bahas nanti).

Agar lebih jelas, berikut hasil perintah `print_r()`:

```
Array
(
    [nama_file] => Gambar Koala
    [submit] => Proses Upload
)
Array
(
    [file_upload] => Array
        (
            [name] => Koala.jpg
            [type] => image/jpeg
            [tmp_name] => C:\xampp\tmp\phpC9F.tmp
            [error] => 0
            [size] => 780831
        )
)
)
```

Mari kita bahas hasil kode ini.

Perhatikan bahwa hasil form dipecah ke dalam 2 *variabel superglobal*: `$_POST` dan `$_FILES`. Dalam contoh ini saya mengisi nilai text box dengan “Gambar Koala”. Isian form akan masuk ke variabel `$_POST["nama_file"]`.

Jika saya menambah objek form lain seperti *checkbox*, *radio button*, atau *textarea*, semuanya tetap masuk ke dalam variabel `$_POST`, bukan `$_FILES`. Contoh lain, tombol submit yang bernilai “Proses Upload” tetap berada di `$_POST["submit"]`.

Variabel `$_FILES` hanya akan berisi data tentang file yang diupload, yakni yang berasal dari tag `<input type="file">`. Karena di dalam form saya menulis atribut `name="file_upload"`, maka seluruh informasi tersimpan kedalam variabel `$_FILES["file_upload"]`.

Bukan itu saja, variabel `$_FILES` sebenarnya merupakan sebuah *associative array* 2 dimensi. Untuk mendapatkan nama file yang diupload, saya mengaksesnya dari variabel: `$_FILES["file_upload"]["name"]`.

Dari contoh diatas, kita juga bisa dilihat bahwa file upload memiliki 5 informasi:

```
[name] => Koala.jpg
[type] => image/jpeg
[tmp_name] => C:\xampp\tmp\phpC9F.tmp
[error] => 0
[size] => 780831
```

- **name**: Nama file yang sedang di upload. Dalam contoh ini, “*Koala.jpg*” merupakan nama file yang saya upload.
- **type**: MIME type dari file yang diupload. Karena saya mengupload gambar bertipe jpg, hasil MIME typenya adalah *image/jpeg*.

- **tmp\_name**: lokasi sementara (*temporary*) file yang diupload. Disinilah file upload disimpan untuk sementara. Selama kita tidak memindahkannya, file ini belum bisa diakses. Saya akan membahasnya sesaat lagi.
- **error**: kode error yang terjadi. Nilai 0 berarti tidak terdapat error. Jika berisi angka 1 - 8 berarti terjadi error.
- **size**: ukuran file yang diupload dalam satuan byte. 780831 Byte = 762 KB. Dalam perhitungan komputer, 1KB tidak sama dengan 1000 Byte, tapi 1024 Byte. Oleh karena itu, 780831 Byte = 780831/1024 KB = 762 KB.

Seluruh nilai ini dibutuhkan untuk memproses hasil upload dan untuk proses validasi. Saya bisa membuat kode program yang hanya menerima file gambar. Atau membuat kode program yang membatasi ukuran file agar tidak lebih dari 1MB. Semua ini akan kita bahas dengan lebih detail sesaat lagi.



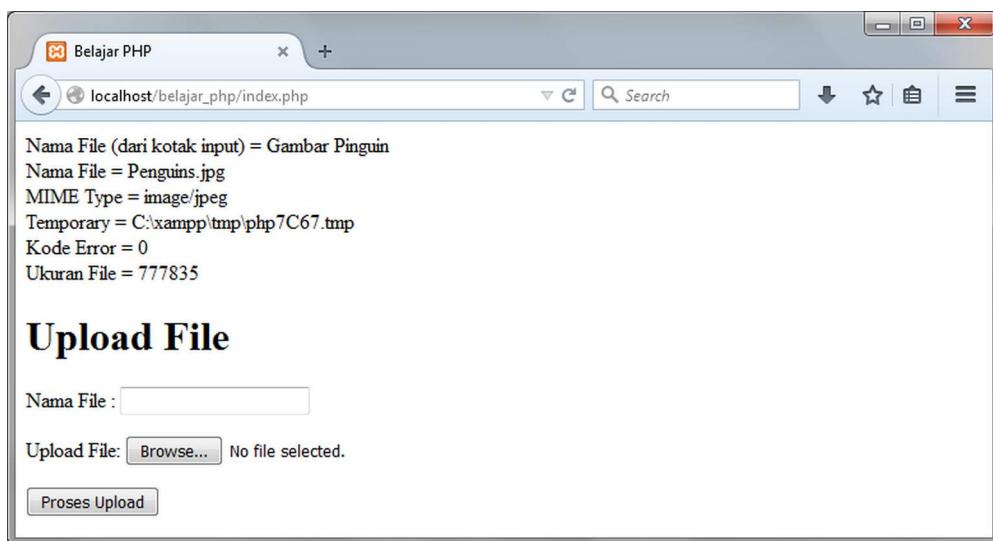
Jika anda ingin melakukan sedikit percobaan, silahkan hapus atribut `enctype="multipart/form-data"` dari tag `<form>`, dan lihat apa yang terjadi.

Sebelum kita masuk ke pembahasan berikutnya, dapatkah anda mengubah kode diatas agar menampilkan setiap informasi menggunakan perintah `echo` (bukan dari `print_r()`)? Ini artinya kita akan mengakses satu persatu nilai variabel `$_POST` dan `$_FILES`.

Berikut kode yang saya gunakan:

```
1 <?php
2 // cek apakah form telah di submit
3 if (isset($_POST["submit"])) {
4     // form telah disubmit, proses data
5     echo "Nama File (dari kotak input) = ".$_POST["nama_file"]."<br>";
6     // tampilkan informasi tentang file yang diupload
7     echo "Nama File = ".$_FILES["file_upload"]["name"]."<br>";
8     echo "MIME Type = ".$_FILES["file_upload"]["type"]."<br>";
9     echo "Temporary = ".$_FILES["file_upload"]["tmp_name"]."<br>";
10    echo "Kode Error = ".$_FILES["file_upload"]["error"]."<br>";
11    echo "Ukuran File = ".$_FILES["file_upload"]["size"];
12 }
13 ?>
14 <!DOCTYPE html>
15 <html>
16 <head>
17   <meta charset="UTF-8">
18   <title>Belajar PHP</title>
19 </head>
20 <body>
21 <h1>Upload File</h1>
22 <form action="index.php" method="post" enctype="multipart/form-data">
```

```
23  <p>Nama File : <input type="text" name="nama_file"></p>
24  <p>Upload File: <input type="file" name="file_upload"></p>
25  <input type="submit" name="submit" value="Proses Upload">
26  </form>
27 </body>
28 </html>
```

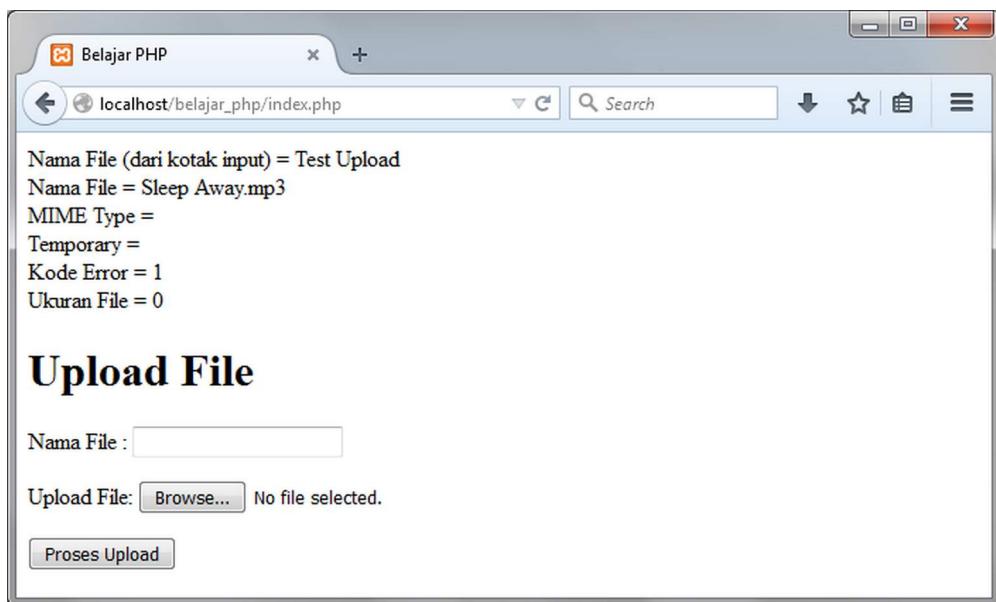


Gambar: Hasil informasi form ditampilkan dengan perintah echo()

Khusus untuk nama file, saya tulis 2 kali. Yang pertama berasal dari `$_POST["nama_file"]`, yakni dari kotak input text box, sedangkan yang kedua berasal dari `$_FILES["file_upload"]["name"]`, yakni dari file yang diupload.

## 21.2 Pengaturan Terkait File Upload (php.ini)

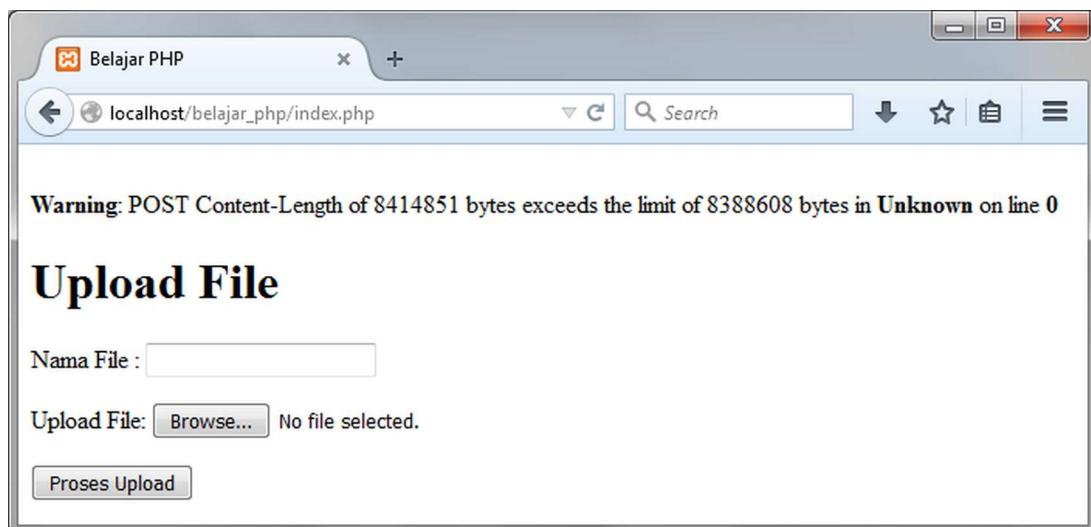
Sebelum kita mulai membahas pengaturan / setting terkait file upload, mari lakukan sedikit percobaan. Silahkan anda upload file dengan ukuran lebih besar dari 2MB, namun kurang dari 8MB. Berikut hasilnya:



Gambar: Upload file dengan ukuran 4MB

Tampilan diatas saya dapat ketika mengupload file Sleep Away.mp3 yang berukuran 4MB. Perhatikan apa yang terjadi. Seluruh keterangan mengenai file upload kosong, kecuali kode **error = 1**.

Sekarang, silahkan upload file dengan ukuran > 8MB. Berikut hasil yang saya dapat:



Gambar: Upload file dengan ukuran 8,02MB

Kali ini seluruh informasi tentang file upload dan juga variabel `$_POST` tidak tampil sama sekali. Yang keluar adalah pesan error: *Warning: POST Content-Length of 8414851 bytes exceeds the limit of 8388608 bytes in Unknown on line 0*.

### Kenapa ini bisa terjadi?

Semuanya terkait dengan pengaturan PHP tentang seberapa besar file yang boleh diupload. Semua ini berasal dari file konfigurasi **php.ini**. Mari kita lihat apa saja pengaturan yang

berhubungan dengan file upload.

Silahkan buka **phpinfo**. Jika anda masih ingat, tampilan phpinfo bisa diakses dari halaman awal localhost: <http://localhost/dashboard/phpinfo.php><sup>1</sup>, atau dari file PHP dengan menjalankan fungsi **php\_info()**.

## upload\_max\_filesize

sys_temp_dir	no value	no value
track_errors	On	On
unserialize_callback_func	no value	no value
upload_max_filesize	2M	2M
upload_tmp_dir	C:\xampp\tmp	C:\xampp\tmp
user_dir	no value	no value
user_ini.cache_ttl	300	300
user_ini.filename	.user.ini	.user.ini

Gambar: Pengaturan **upload\_max\_filesize** dari **phpinfo**

**upload\_max\_filesize** digunakan untuk membatasi ukuran maksimum file upload. Dalam settingan default XAMPP yang saya gunakan, nilainya adalah 2M. Ini artinya file yang diupload dibatasi maksimal 2MB untuk 1 buah file. Jika lebih dari itu, file upload akan gagal (error).

## post\_max\_size

output_encoding	no value	no value
output_handler	no value	no value
post_max_size	8M	8M
precision	14	14
realpath_cache_size	16K	16K
realpath_cache_ttl	120	120

Gambar: Pengaturan **post\_max\_size** dari **phpinfo**

**post\_max\_size** adalah batas maksimum total seluruh file dalam 1 kali pengiriman (1 kali POST-ing). Di dalam sebuah form, kita bisa membuat beberapa tag `<input type="file">`. Setiap tag ini dibatasi oleh **upload\_max\_size**. Total seluruh file ini tidak boleh melewati **post\_max\_size**. Dalam pengaturan default XAMPP yang saya gunakan, nilai **post\_max\_size** = 8M.

Settingan **upload\_max\_filesize** = 2M dan **post\_max\_size** = 8M artinya, untuk setiap tag `<input type="file">` ukuran file tidak boleh melebihi 2MB, dan total seluruh file yang diupload tidak boleh melebihi 8MB. Dalam percobaan sebelumnya, saya telah memperlihatkan apa yang terjadi ketika kita men-upload file yang melebihi batas-batas ini.

<sup>1</sup><http://localhost/dashboard/phpinfo.php>

## max\_file\_uploads

mail.force_extra_parameters	no value	no value
mail.log	no value	no value
max_execution_time	30	30
max_file_uploads	20	20
max_input_nesting_level	64	64
...	...	...

Gambar: Pengaturan max\_file\_uploads dari phpinfo

**max\_file\_uploads** berfungi untuk membatasi berapa banyak file yang bisa diupload dalam 1 kali pengiriman form. Nilai defaultnya adalah 20, artinya kita hanya bisa mengupload maksimal 20 file dalam 1 form. Tentunya total ukuran file seluruh file ini juga tidak boleh melebihi nilai **post\_max\_size**.

## max\_execution\_time

mail.force_extra_parameters	no value	no value
mail.log	no value	no value
max_execution_time	30	30
max_file_uploads	20	20
max_input_nesting_level	64	64
...	...	...

Gambar: pengaturan max\_execution\_time dari phpinfo

**max\_execution\_time** sebenarnya tidak hanya berkaitan dengan pemrosesan file upload, tapi seluruh pemrosesan di dalam PHP. Nilai **max\_execution\_time = 30** berarti setiap proses yang dilakukan oleh PHP tidak boleh melebihi 30 detik. Jika lebih, proses akan dihentikan paksa dan tampil pesan error.

Ketika kita mengupload file berukuran besar di jaringan yang lambat, sangat mungkin proses upload membutuhkan waktu lebih dari 30 detik.

## memory\_limit

max_input_time	60	60
max_input_vars	1000	1000
memory_limit	128M	128M
open_basedir	no value	no value
output_buffering	no value	no value

Gambar: Pengaturan memory\_limit dari phpinfo

Sama seperti **max\_execution\_time**, **memory\_limit** juga tidak hanya berkaitan dengan file upload. Ini adalah total maksimum memori yang dialokasikan kepada PHP. Jika PHP mengerjakan kode program yang cukup rumit, atau memproses file berukuran besar, akan membutuhkan alokasi memori yang besar pula. Nilai default dari **memory\_limit** adalah 128M. Artinya PHP diberikan jatah memori sebesar 128 MB.

## upload\_tmp\_dir

upload_max_filesize	2M	2M
upload_tmp_dir	C:\xampp\tmp	C:\xampp\tmp
user_dir	no value	no value
user_ini.cache_ttl	300	300

Gambar: Pengaturan upload\_tmp\_dir dari phpinfo

**upload\_tmp\_dir** adalah lokasi folder sementara (*temporary directory*) dari file upload. Nilai default dari XAMPP yang saya gunakan berada di C:\xampp\tmp. Artinya, ketika sebuah file diupload, ia akan ditempatkan ke dalam folder C:\xampp\tmp untuk sementara sebelum diproses atau pindahkan ke tempat lain.

Kita juga bisa memberikan nilai **upload\_tmp\_dir = NULL**. Jika di set dengan nilai **NULL**, PHP akan menggunakan *folder temporary* yang diatur oleh sistem operasi.

Dalam sistem operasi linux (yang umum digunakan pada web server), alamat folder ini juga harus bisa diakses oleh PHP (diberikan hak aksesnya).

## file\_uploads

expose_php	On	On
extension_dir	C:\xampp\php\ext	C:\xampp\php\ext
file_uploads	On	On
highlight.comment	#FF8000	#FF8000
highlight.default	#0000BB	#0000BB

Gambar: Pengaturan file\_uploads dari phpinfo

**file\_uploads** bisa jadi merupakan pengaturan paling penting untuk pemrosesan file upload. Karena jika settingan **file\_uploads** diisi **“off”**, PHP tidak akan bisa memproses form upload. Agar bisa berjalan, **file\_uploads** harus bernilai **“on”**, **“1”** atau **“true”**.

Seluruh pengaturan ini bisa diubah melalui file **php.ini**. Jika anda ingin melakukan percobaan, silahkan ubah beberapa settingan yang ada, dan jangan lupa untuk *restart* apache agar perubahan bisa berjalan.

```
907 ;;;;;;;;;;;;
908 ; File Uploads ;
909 ;;;;;;;;;;;;
910
911 ; Whether to allow HTTP file uploads.
912 ; http://php.net/file-uploads
913 file_uploads=On
914
915 ; Temporary directory for HTTP uploaded files (will use system default if not
916 ; specified).
917 ; http://php.net/upload-tmp-dir
918 upload_tmp_dir="C:\xampp\tmp"
919
920 ; Maximum allowed size for uploaded files.
921 ; http://php.net/upload-max-filesize
922 upload_max_filesize=2M
923
924 ; Maximum number of files that can be uploaded via a single request
925 max_file_uploads=20
```

Gambar: Mengubah pengaturan file upload dari `php.ini`

Dalam lanjutan bab ini, saya akan tetap menggunakan settingan default, yakni `upload_max_filesize = 2M` dan `post_max_size = 8M`.

Yang perlu diingat, jika anda sedang mengembangkan aplikasi atau form dengan file upload, pertimbangkan mengatur settingan ini agar sesuai dengan keinginan.

Misalkan jika saya membuat sebuah online gallery dimana user bisa mengupload gambar, ukuran file gambar sebesar 2MB dirasa sudah cukup besar. Tetapi jika saya membuat situs video sharing, 2MB tidak akan mencukupi. Saya bisa mengubah batasan file upload menjadi 50MB atau 100MB.

## 21.3 Memahami Kode Error File Upload

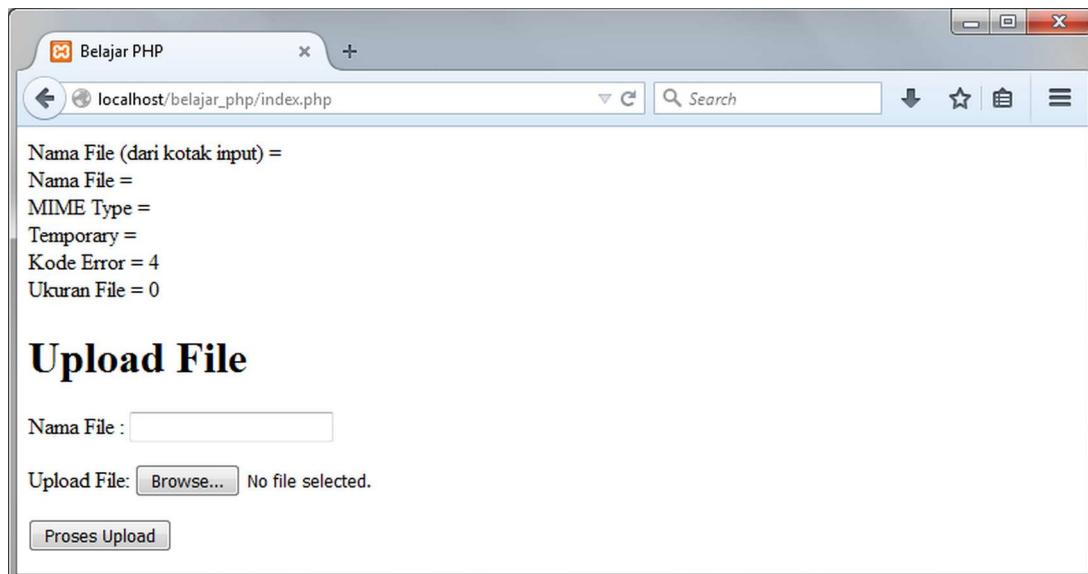
Dari contoh program yang telah kita buat, anda dapat melihat terdapat informasi mengenai kode error, yakni dari variabel `$_FILES["file_upload"]["error"]`. Variabel ini mengindikasikan apakah terdapat error atau tidak pada saat proses file upload. Nilainya disajikan dalam bentuk angka 0-8. Berikut penjelasannya:

- 0: Tidak ada error, file sukses sampai ke web server.
- 1: Ukuran file melewati batas `upload_max_filesize`. Pengaturan ini diset dari `php.ini`.
- 2: Ukuran file melewati batas `MAX_FILE_SIZE`. Pengaturan ini bisa di set dari form, dan akan saya bahas sesaat lagi.
- 3: File hanya diupload sebagian. Kemungkinan besar karena masalah jaringan atau koneksi yang tidak lancar.
- 4: File upload tidak ditemukan. Ini terjadi ketika form di submit, namun user tidak memilih file apapun (form kosong).
- 6: Folder sementara (*temporary directory*) tidak ditemukan. Error ini bisa terjadi karena salah menulis alamat folder `upload_tmp_dir`. Atau folder tersebut sudah terhapus.
- 7: PHP tidak bisa menulis ke harddisk server. Ini bisa terjadi ketika PHP tidak mendapat hak akses ke dalam *temporary directory*. Dalam server linux, setiap file maupun folder bisa dibatasi hak aksesnya.

- 8: Proses upload dihentikan oleh extension lain dari PHP. Artinya, terdapat kode program di server yang mencegah file di upload.

Dari seluruh kode error ini, tidak terdapat kode error nomor 5. Error nomor 5 sebelumnya ada, namun sudah dihilangkan dari PHP.

Silahkan anda coba berbagai kondisi untuk mendapatkan kode error diatas. Kode error 1 terjadi karena ukuran file melebihi **upload\_max\_filesize** (sudah saya praktekkan sebelumnya). Kode error 4 bisa didapat dengan men-submit form kosong.



Gambar: Kode error 4 terjadi karena saya tidak memilih file apapun untuk diupload

Berdasarkan nomor error, kita bisa membuat pesan kesalahan kepada pengunjung web. Caranya dengan memeriksa variabel `$_FILES["file_upload"]["error"]`, lalu memberikan keterangan kepada variabel `$pesan_error`. Seperti contoh berikut:

```

1 <?php
2 // cek apakah form telah di submit
3 if (isset($_POST["submit"])) {
4     // form telah disubmit, proses data
5     // siapkan array untuk pesan error
6     $arr_upload_error = array(
7         0 => 'File sukses di upload',
8         1 => 'Upload gagal, Ukuran file melewati batas maksimal 2MB',
9         2 => 'Upload gagal, Ukuran file melewati batas maksimal',
10        3 => 'Upload gagal, File hanya ter-upload sebagian',
11        4 => 'Upload gagal, Tidak ada file yang terupload',
12        6 => 'Upload gagal, Server Error',
13        7 => 'Upload gagal, Server Error',
14        8 => 'Upload gagal, Server Error',
15    );
16 $error = $_FILES["file_upload"]["error"];

```

```
17     $pesan_error = $arr_upload_error[$error];
18 }
19 ?>
20 <!DOCTYPE html>
21 <html>
22 <head>
23   <meta charset="UTF-8">
24   <title>Belajar PHP</title>
25 </head>
26 <body>
27 <h1>Upload File</h1>
28 <?php if (!empty($pesan_error)) {echo "<p>$pesan_error</p>";} ?>
29   <form action="index.php" method="post" enctype="multipart/form-data">
30     <p>Upload File: <input type="file" name="file_upload"></p>
31     <input type="submit" name="submit" value="Proses Upload">
32   </form>
33 </body>
34 </html>
```

Disini saya membuat sebuah array `$arr_upload_error` yang berisi semua kemungkinan kode error. Setiap elemant array berisi pesan mengenai apa yang terjadi.

Saya juga membuat penjelasan tentang error dengan “bahasa awam”. Maksudnya daripada membuat pesan error: ‘*Upload gagal, temporary directory tidak bisa diakses*’, akan lebih user friendly jika saya menggunakan pesan error ‘*Upload gagal, Server Error*’. Pengguna web kita tidak perlu tau apa itu *temporary directory*. Kita cukup informasikan bahwa kesalahan terjadi di sisi server.

Selanjutnya, saya tinggal mengisi variabel `$pesan_error` dengan nilai dari `$_FILES["file_upload"]["error"]`. Ini saya lakukan dengan 2 baris kode program berikut:

```
$error = $_FILES["file_upload"]["error"];
$pesan_error = $arr_upload_error[$error];
```

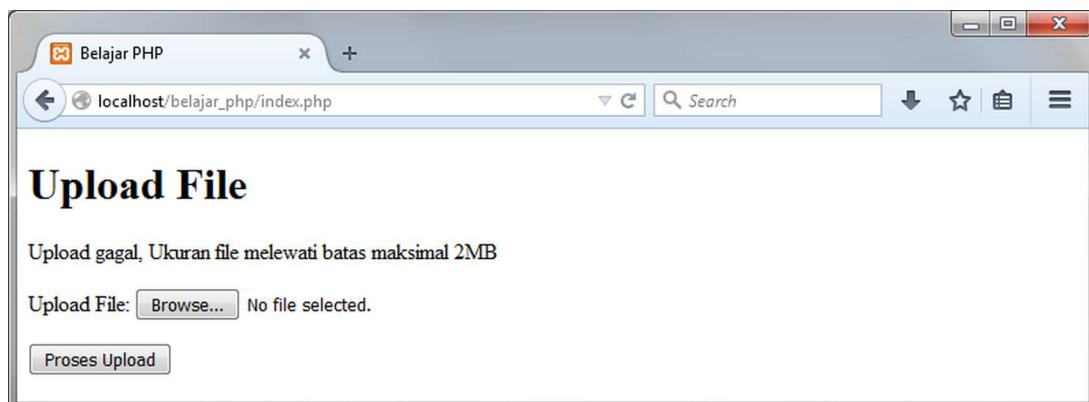
Ketika terjadi error yang disebabkan ukuran file melebihi 2MB, variabel `$_FILES["file_upload"]["error"]` akan berisi angka 1. Nilai 1 kemudian disimpan ke dalam `$error`. Baris program berikutnya akan mengakses isi array `$arr_upload_error[1]`, yang disimpan ke dalam `$pesan_error`.

Isi dari `$pesan_error` ini selanjutnya saya tampilkan sebelum form, dengan perintah:

```
<?php if (!empty($pesan_error)) {echo "<p>$pesan_error</p>";} ?>
```

Artinya, tampilkan isi dari `$pesan_error` hanya ketika variabel tersebut tidak kosong.

Untuk contoh ini dan selanjutnya, saya juga menghapus tag `<input type="text">` agar form kita lebih sederhana.



Gambar: Pesan error tampil karena ukuran file melebihi 2MB

## 21.4 Membatasi Ukuran File Upload dengan MAX\_FILE\_SIZE

Selain dari `php.ini`, kita juga bisa membatasi ukuran file upload dari dalam HTML. Caranya dengan menambahkan sebuah tag `<input type="hidden">` ke dalam form. Berikut contoh penulisannya:

```
<input type="hidden" name="MAX_FILE_SIZE" value="1000000">
```

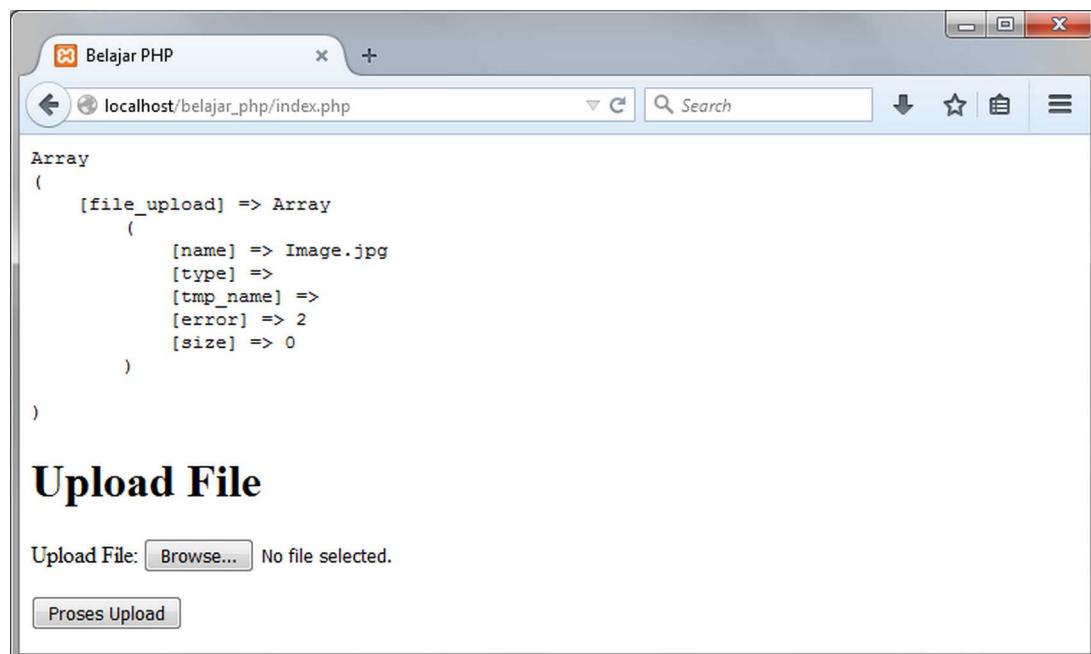
Sebagaimana yang kita ketahui, tag `<input type="hidden">` tidak akan terlihat di web browser. Tag ini biasanya digunakan untuk membantu coding PHP seperti dalam contoh ini. Agar dapat bekerja, tag diatas harus ditempatkan sebelum tag `<input type="file">`, seperti berikut ini:

```
1 <form action="index.php" method="post" enctype="multipart/form-data">
2   <input type="hidden" name="MAX_FILE_SIZE" value="1000000">
3   <p>Upload File: <input type="file" name="file_upload"></p>
4   <input type="submit" name="submit" value="Proses Data">
5 </form>
```

Perhatikan atribut `name="MAX_FILE_SIZE"` dan `value="1000000"`. Ini artinya, saya ingin membatasi file upload sebesar 1000000 byte, atau sekitar 1MB (sebenarnya 1MB terdiri dari 1.048.576 byte).

Ketika saya mengupload file dengan ukuran > 1MB. PHP akan menghasilkan kode error 2. Mari kita coba dengan kode berikut ini:

```
1 <?php
2 // cek apakah form telah di submit
3 if (isset($_POST["submit"])) {
4     // form telah disubmit, proses data
5     echo "<pre>";
6     print_r($_FILES);
7     echo "</pre>";
8 }
9 ?>
10 <!DOCTYPE html>
11 <html>
12 <head>
13     <meta charset="UTF-8">
14     <title>Belajar PHP</title>
15 </head>
16 <body>
17 <h1>Upload File</h1>
18 <form action="index.php" method="post" enctype="multipart/form-data">
19     <input type="hidden" name="MAX_FILE_SIZE" value="1000000">
20     <p>Upload File: <input type="file" name="file_upload"></p>
21     <input type="submit" name="submit" value="Proses Upload">
22 </form>
23 </body>
24 </html>
```



Gambar: PHP mengeluarkan error 2 karena ukuran file Image.jpg sebesar 1,3MB

File *Image.jpg* yang saya coba upload berukuran 1,3MB, ini sebenarnya masih dibawah 2MB yang menjadi ukuran maksimal file (yang di set dari **upload\_max\_filesize** *php.ini*). Namun

ternyata PHP sudah mengeluarkan error. Inilah efek dari penambahan tag `<input type="hidden">`.

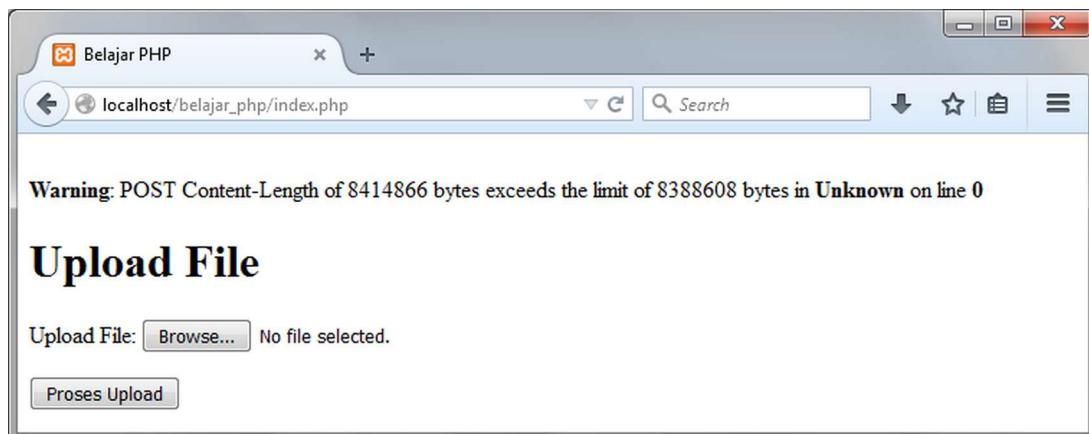
Sekarang, ukuran file maksimal yang bisa diupload dibatasi sebesar 1MB. Walaupun saya coba upload file yang berukuran 3MB, pesan error yang tampil tetap 2, bukan lagi 1. Dengan kata lain, settingan dari `MAX_FILE_SIZE` seolah-olah menimpa settingan `upload_max_filesize`.

Akan tetapi, metode ini memiliki kelemahan. Sebagaimana cara kerja HTML yang di proses di web browser, pembatasan dari `MAX_FILE_SIZE` dengan mudah dapat dibobol. Seseorang bisa menghapus tag ini di web browser mereka dan mengirim file untuk diupload. Akibatnya, validasi 1MB tidak lagi berlaku. Oleh karena itu, `MAX_FILE_SIZE` tidak bisa diandalkan. Kita sebaiknya tetap berpatokan kepada `upload_max_filesize`.

## 21.5 Mengatasi Error Warning: POST Content-Length

Jika anda mencoba mengupload file yang ukurannya lebih besar dari `post_max_size`, tidak akan tampil angka error dari `$_FILES["file_upload"]["error"]`. Jika anda masih ingat, `post_max_size` adalah ukuran maksimum total file upload dari suatu form.

Dalam `php.ini` yang saya gunakan, nilai `post_max_size = 8MB`. Artinya, total seluruh file upload dalam satu form tidak boleh melebihi 8MB. Apa dampaknya jika saya tetap upload file > 8MB? Berikut hasilnya:



Gambar: Error Warning: POST Content-Length

Hasil diatas saya dapat ketika mencoba mengupload file berukuran 8,2 MB, tepatnya 8414866 byte.

Ada hal menarik dari kode error diatas. Yang pertama, tidak ada tampilan hasil `print_r($_FILES)`. Yang kedua, perhatikan kata terakhir dari pesan error: "on line 0". Dimanakah baris nol? Selama ini dalam setiap pesan error, PHP selalu menampilkan baris mana yang terjadi kesalahan. Baris paling awal adalah baris ke-1 atau "on line 1". Tapi ini adalah *line 0*!

Error ini masuk ke dalam kategori **startup errors**. Yakni error yang terjadi sebelum kode PHP dijalankan. Tipe error seperti ini cukup sulit diatasi.

Ketika ukuran file upload melebihi nilai maksimum `post_max_size`, PHP melakukan hal yang diluar prediksi. Yakni menghapus seluruh isi variabel `$_POST` dan `$_FILES`. Inilah yang menjelaskan kenapa perintah `print_r()` tidak mengeluarkan tampilan apa pun.

Salah satu solusi untuk mengatasi error ini adalah dengan menaikkan nilai `post_max_size`, katakan menjadi 100MB. Dengan demikian, peluang terjadi error bisa sedikit diatasi.

Namun ini bukan solusi yang ideal. Bagaimana jika seseorang mencoba mengupload file dengan ukuran 1GB? Error yang sama akan tampil kembali. Yang juga menyulitkan, kita tidak bisa mendapatkan informasi kode error karena variabel `$_FILES` memang sudah tidak ada.

Solusinya saya temukan di [stackoverflow.com](http://stackoverflow.com)<sup>2</sup>, dan mungkin sedikit rumit. Caranya adalah dengan membuat validasi seperti berikut ini:

```
1 if ($_SERVER['REQUEST_METHOD'] == 'POST' && empty($_FILES) && empty($_POST))
2 {
3     $post_max = ini_get('post_max_size');
4     $pesan_error = "Ukuran file melewati batas maksimal ({$post_max}B)";
5 }
```

Untuk memeriksa error ini, kita harus mengecek 3 kondisi: apakah form dikirim dengan method "POST", apakah variabel `$_FILES` kosong dan apakah `$_POST` juga kosong. Jika iya, maka file yang diupload telah melebihi nilai `post_max_size`.

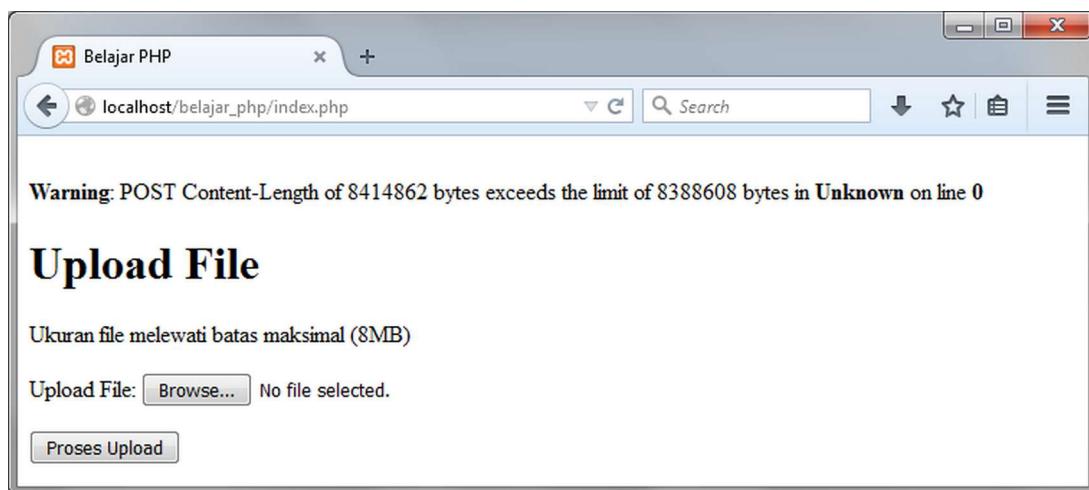
Fungsi `ini_get()` digunakan untuk mendapatkan informasi konfigurasi `php.ini`. Variabel `$post_max` akan berisi nilai hasil settingan `post_max_size` dari `php.ini`. Nilai ini bisa kita ambil untuk menjadi pesan error kepada pengguna.

Berikut adalah kode program lengkap dari validasi ini:

```
1 <?php
2 // cek apakah ukuran file melewati batas maksimal
3 if ($_SERVER['REQUEST_METHOD'] == 'POST' && empty($_FILES) && empty($_POST) \
4 ) {
5     // batas maksimum terlewati, buat pesan error
6     $postMax = ini_get('post_max_size');
7     $pesan_error = "Ukuran file melewati batas maksimal ({$postMax}B)";
8 }
9
10 // cek apakah form telah di submit
11 if (isset($_POST['submit'])) {
12     // form telah disubmit, cek apakah ada error
13     $error = $_FILES['file_upload']['error'];
14     if ($error == 0){
15         $pesan_error = "File sukses di upload";
16     }
17     else {
18         $pesan_error = "File gagal di upload";
19     }
20 }
```

<sup>2</sup><http://stackoverflow.com/questions/2133652/how-to-gracefully-handle-files-that-exceed-phps-post-max-size>

```
21  ?>
22  <!DOCTYPE html>
23  <html>
24  <head>
25      <meta charset="UTF-8">
26      <title>Belajar PHP</title>
27  </head>
28  <body>
29  <h1>Upload File</h1>
30  <?php if (!empty($pesan_error)) {echo "<p>$pesan_error</p>";} ?>
31      <form action="index.php" method="post" enctype="multipart/form-data">
32          <input type="hidden" name="MAX_FILE_SIZE" value="1000000">
33          <p>Upload File: <input type="file" name="file_upload"></p>
34          <input type="submit" name="submit" value="Proses Upload">
35      </form>
36  </body>
37  </html>
```



Gambar: pesan error tampil ketika file upload melebihi post\_max\_size

Yang juga perlu diketahui, validasi ini tidak bisa dilakukan di dalam kondisi `if (isset($_POST["submit"]))`. Karena, ketika ukuran file terlalu besar, variabel `$_POST["submit"]` sudah tidak berisi nilai apa-apa lagi (karena dihapus oleh PHP).

Namun bagaimana cara menghilangkan pesan error: `Warning: POST Content-Length of 8414862 bytes exceeds the limit of 8388608 bytes in Unknown on line 0?`

Kita bisa memaksa PHP untuk tidak menampilkan pesan **startup errors** dengan cara mengubah settingan `php.ini`: `display_startup_errors = Off`. Akan tetapi selama proses development (pembuatan program), sebaiknya tetap mengaktifkan konfigurasi ini dan baru dimatikan ketika website sudah "live".

```

554 ; The display of errors which occur during PHP's startup sequence are handled
555 ; separately from display_errors. PHP's default behavior is to suppress those
556 ; errors from clients. Turning the display of startup errors on can be useful in
557 ; debugging configuration problems. But, it's strongly recommended that you
558 ; leave this setting off on production servers.
559 ; Default Value: Off
560 ; Development Value: On
561 ; Production Value: Off
562 ; http://php.net/display-startup-errors
563 ; XAMPP: Turn display_startup_errors = Off here for a full Joomla support
564 display_startup_errors=On

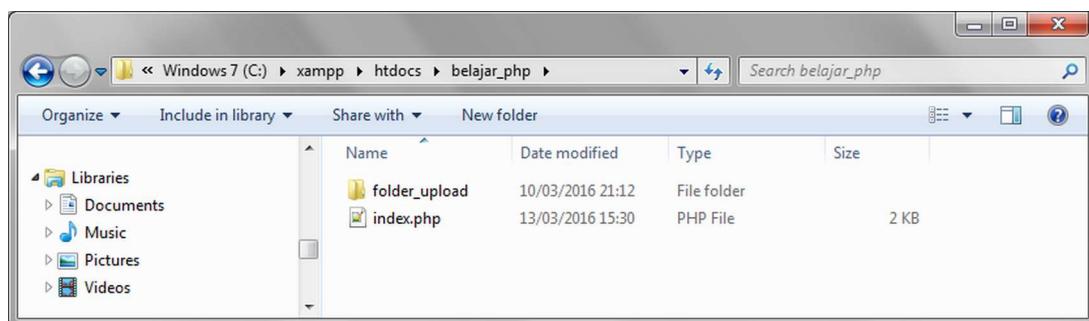
```

Gambar: Pengaturan settingan display\_startup\_errors di dalam php.ini

## 21.6 Memindahkan file Upload

Sampai saat ini, saya masih berlutut dengan validasi dan kesalahan yang terjadi saat file upload di submit. Sekarang kita akan bahas cara mengambil file ini. PHP menyediakan fungsi **move\_uploaded\_file()** untuk memindahkan file upload dari *temporary folder* ke folder yang kita inginkan.

Sebelum bisa menggunakan fungsi ini, kita harus mempersiapkan sebuah lokasi kemana file upload akan disimpan. Sebagai contoh, saya akan membuat folder “**folder\_upload**” yang berada di lokasi yang sama dengan file form PHP.



Gambar: Folder folder\_upload dalam 1 tempat dengan file index.php yang berisi kode untuk pemrosesan form

Lokasi folder ini tidak harus sama dengan file PHP yang memproses form. Anda juga bisa menempatkannya di lokasi atau folder lain selama masih bisa diakses. Berikut kode program lengkap cara memproses dan memindahkan file upload:

```

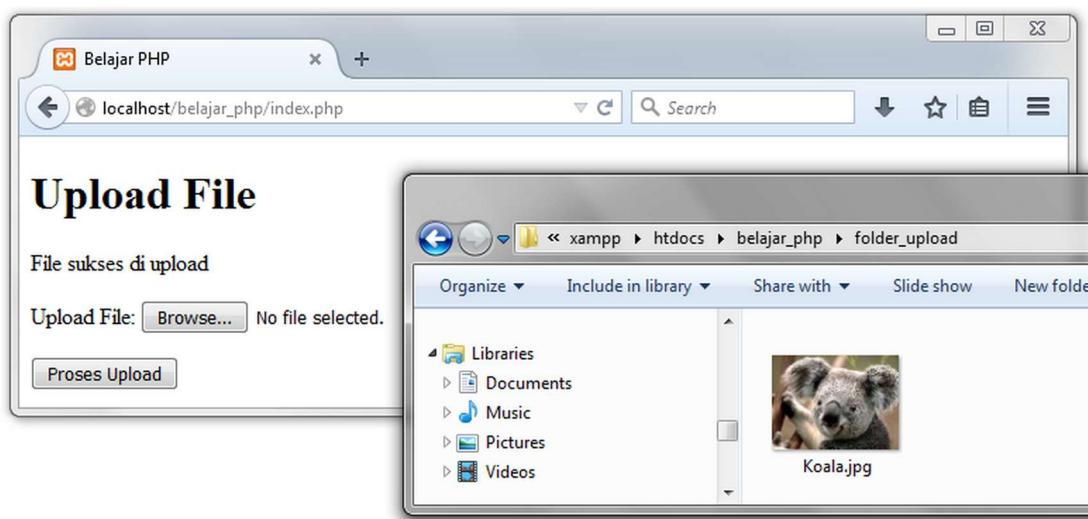
1 <?php
2 // cek apakah form telah di submit
3 if (isset($_POST["submit"])) {
4     // form telah disubmit, cek apakah ada error
5     $error = $_FILES["file_upload"]["error"];
6
7     if ($error === 0){
8         // tidak ada error
9         $pesan_error = "File sukses di upload";
10

```

```

11     // pindahkan file ke folder_upload
12     $nama_folder="folder_upload";
13     $tmp = $_FILES["file_upload"]["tmp_name"];
14     $nama_file = $_FILES["file_upload"]["name"];
15     move_uploaded_file($tmp, "$nama_folder/$nama_file");
16 }
17 else {
18     $pesan_error = "File gagal di upload";
19 }
20 }
21 ?>
22 <!DOCTYPE html>
23 <html>
24 <head>
25     <meta charset="UTF-8">
26     <title>Belajar PHP</title>
27 </head>
28 <body>
29 <h1>Upload File</h1>
30 <?php if (!empty($pesan_error)) {echo "<p>$pesan_error</p>";} ?>
31     <form action="index.php" method="post" enctype="multipart/form-data">
32         <input type="hidden" name="MAX_FILE_SIZE" value="1000000">
33         <p>Upload File: <input type="file" name="file_upload"></p>
34         <input type="submit" name="submit" value="Proses Upload">
35     </form>
36 </body>
37 </html>

```



Gambar: File upload Panda.jpg sukses dipindahkan ke folder\_upload

Silahkan anda jalankan kode program diatas dan upload sebuah file yang berukuran kurang dari 1MB. Kenapa? Karena di dalam form masih terdapat **MAX\_FILE\_SIZE** dengan nilai 1000000.

Artinya maksimal file adalah 1MB.

Jika tampil pesan “*File sukses di upload*”, langsung cek **folder\_upload**. Seharusnya folder tersebut akan berisi 1 file yang kita diupload.

Jika keluar pesan “*File gagal di upload*” kemungkinan besar file yang diupload melebihi 1MB.

Fungsi **move\_uploaded\_file()** membutuhkan 2 argumen. Argumen pertama adalah alamat *temporary directory*. Nilai ini bisa diakses dari variabel `$_FILES["file_upload"]["tmp_name"]`. Argumen kedua adalah lokasi dan nama file kemana file tersebut akan disimpan.

Perhatikan cara penulisan argumen kedua ini. Fungsi **move\_uploaded\_file()** tidak tahu tentang nama file asli dari yang kita diupload. Oleh karena itulah kita harus mengambil nama asli file dari variabel `$_FILES["file_upload"]["name"]`.

Fungsi *move\_uploaded\_file()* juga akan mengembalikan nilai **FALSE** jika tidak bisa memindahkan file, misalkan folder tujuan belum ada, atau masalah lain. Ini bisa kita manfaatkan agar kode program menjadi lebih rapi. Berikut perubahannya:

```
1 if ($error === 0){  
2     // pindahkan file ke folder_upload  
3     $nama_folder="folder_upload";  
4     $tmp = $_FILES["file_upload"]["tmp_name"];  
5     $nama_file = $_FILES["file_upload"]["name"];  
6     if (move_uploaded_file($tmp, "$nama_folder/$nama_file")){  
7         $pesan_error = "File sukses di upload";  
8     }  
9     else {  
10        $pesan_error = "File gagal di upload";  
11    }  
12 }
```

Saya menambahkan sebuah kondisi **if** untuk “melindungi” fungsi *move\_uploaded\_file()*. Jika karena suatu hal fungsi ini tidak berjalan, sudah kita antisipasi dengan menampilkan pesan “*File gagal di upload*”.

Silahkan anda test upload berbagai file dan periksa **folder\_upload**.

## 21.7 Validasi untuk File Upload yang Memiliki Nama Sama

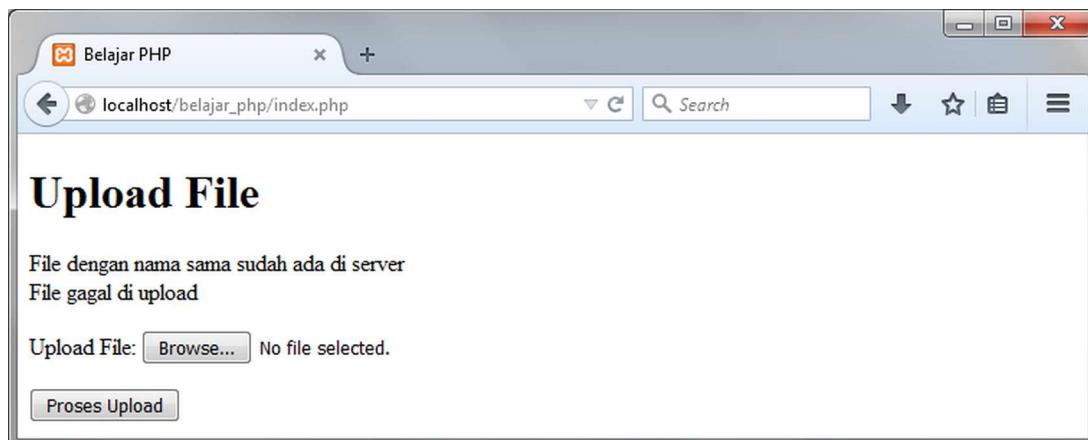
Jika anda mencoba upload file yang memiliki nama sama, fungsi **move\_uploaded\_file()** tetap mengizinkan hal ini dan tidak menampilkan error apapun. File terakhir akan menampa file lama yang memiliki nama sama (*overwrite*). Bagaimana jika kita tidak ingin hal ini terjadi?

Sebelum menjalankan fungsi **move\_uploaded\_file()**, kita bisa memeriksa apakah sebuah file sudah ada atau tidak. Ini dilakukan melalui fungsi **file\_exists()**. Fungsi **file\_exists()** membutuhkan 1 buah argumen, yaitu alamat file yang ingin diperiksa. Fungsi ini akan mengembalikan nilai **TRUE** jika file tersebut ada, atau **FALSE** jika file yang dicari tidak ada.

Dengan memanfaatkan fungsi `file_exists()`, saya bisa membuat sebuah kode program yang menampilkan pesan error jika file yang di upload ternyata memiliki nama yang sama dengan file yang sudah ada di server. Berikut kode programnya:

```
1 <?php
2 // cek apakah form telah di submit
3 if (isset($_POST["submit"])) {
4     // form telah disubmit, cek apakah ada error
5     $error = $_FILES["file_upload"]["error"];
6
7     if ($error === 0){
8
9         // siapkan variabel untuk pesan error
10        $pesan_error = "";
11
12        // siapkan variabel untuk pemindahan file
13        $nama_folder="folder_upload";
14        $tmp = $_FILES["file_upload"]["tmp_name"];
15        $nama_file = $_FILES["file_upload"]["name"];
16        $path_file = "$nama_folder/$nama_file";
17        $upload_gagal = false;
18
19        // cek apakah terdapat file dengan nama yang sama
20        if (file_exists($path_file)) {
21            $pesan_error = "File dengan nama sama sudah ada di server <br>";
22            $upload_gagal = true;
23        }
24        // pindahkan file upload jika semuanya OK
25        if (!$upload_gagal AND move_uploaded_file($tmp,$path_file)){
26            $pesan_error = "File sukses di upload";
27        }
28        else {
29            $pesan_error .= "File gagal di upload";
30        }
31    }
32 }
33 ?>
34 <!DOCTYPE html>
35 <html>
36 <head>
37     <meta charset="UTF-8">
38     <title>Belajar PHP</title>
39 </head>
40 <body>
41 <h1>Upload File</h1>
42 <?php if (!empty($pesan_error)) {echo "<p>$pesan_error</p>";} ?>
```

```
43  <form action="index.php" method="post" enctype="multipart/form-data">
44    <input type="hidden" name="MAX_FILE_SIZE" value="1000000">
45    <p>Upload File: <input type="file" name="file_upload"></p>
46    <input type="submit" name="submit" value="Proses Upload">
47  </form>
48 </body>
49 </html>
```



Gambar: Pesan error karena file sudah ada di server

Saya memodifikasi kode program kita sebelumnya. Kali ini dengan menambahkan sebuah penanda (atau dikenal juga dengan istilah “*flag*”).

Penanda yang dimaksud adalah variabel `$upload_gagal`. Saat file upload akan diproses, nilai `$upload_gagal` akan diset dengan nilai `false`. Ketika fungsi `file_exists($path_file)` menghasilkan nilai `TRUE`, yang artinya file sudah ada, variabel `$upload_gagal` saya set menjadi `true`.

Dibaris berikutnya, fungsi `move_uploaded_file()` hanya akan dijalankan jika `$upload_gagal = false`.

Silahkan anda pelajari sebentar alur logika program yang saya buat. Berbekal latihan dan materi PHP sepanjang buku ini, saya yakin anda sudah bisa memahaminya.

Keputusan untuk tidak membolehkan seseorang mengupload file dengan nama yang sama sebenarnya kurang *user friendly*. Karena, bukan salah si user hal ini terjadi.

Solusi alternatif, kita bisa me-*rename* setiap file upload, misalkan menjadi 0001.jpg, 0002.jpg, 0003.jpg, dst. Atau bisa juga dengan menggenerate nama unik berdasarkan username, seperti andi-001.jpg, andi-002.jpg, dst. Tentunya kita harus membuat kode program tambahan untuk bisa menghasilkan nama file seperti ini.

## 21.8 Membatasi Ukuran File Upload

Cara pembatasan ukuran file upload telah kita bahas beberapa kali, mulai dari settingan `php.ini` hingga menambahkan tag `input type="hidden"`.

Dalam kebanyakan kasus, kita tidak bisa mengakses file `php.ini`. Contohnya jika anda berencana menggunakan web hosting dengan tipe “*shared hosting*”. Di dalam *shared hosting*, kita tidak diizinkan mengutak-atik settingan `php.ini` (untuk alasan keamanan). Oleh karena itu, pengaturan ukuran file upload tidak bisa dilakukan dari `upload_max_filesize` dan `post_max_size`.

Alasan lain, sangat mungkin kita memiliki berbagai form upload untuk keperluan yang berbeda-beda. Mengatur settingan `upload_max_filesize` dan `post_max_size` dari `php.ini` akan berdampak ke seluruh form di server, tidak peduli itu form untuk upload video maupun form untuk upload gambar profil.

Disisi lain, menggunakan konstanta `MAX_FILE_SIZE` dari tag `<input>` juga tidak bisa dian-  
dalkan, karena kode HTML dapat diubah dari dari web browser.

Solusi yang bisa kita gunakan adalah dengan memeriksa variabel `$_FILES["file_upload"]["size"]`. Jika nilainya melebihi batas yang sudah ditentukan, hentikan proses upload dan tampilkan pesan kesalahan.

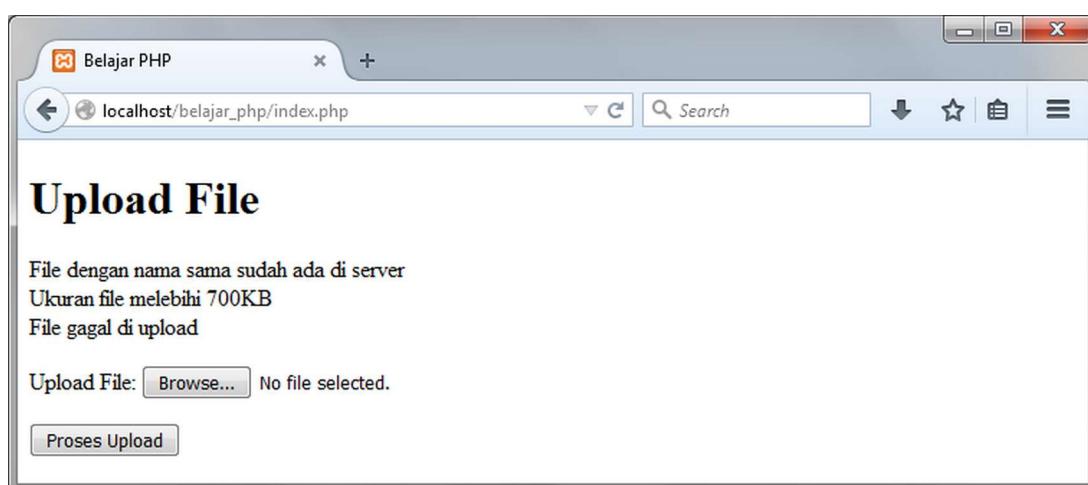
Dari kode program sebelumnya, saya bisa modifikasi untuk menambahkan validasi ini, seperti contoh berikut:

```
1 <?php
2 // cek apakah form telah di submit
3 if (isset($_POST["submit"])) {
4     // form telah disubmit, cek apakah ada error
5     $error = $_FILES["file_upload"]["error"];
6
7     if ($error === 0){
8
9         // siapkan variabel untuk pesan error
10        $pesan_error = "";
11
12        // siapkan variabel untuk pemindahan file
13        $nama_folder="folder_upload";
14        $tmp = $_FILES["file_upload"]["tmp_name"];
15        $nama_file = $_FILES["file_upload"]["name"];
16        $path_file = "$nama_folder/$nama_file";
17        $upload_gagal = false;
18
19        // cek apakah terdapat file dengan nama yang sama
20        if (file_exists($path_file)) {
21            $pesan_error = "File dengan nama sama sudah ada di server <br>";
22            $upload_gagal = true;
23        }
24
25        // cek apakah ukuran file tidak melebihi 700KB (716800 byte)
26        $ukuran_file = $_FILES["file_upload"]["size"];
27        if ($ukuran_file > 716800) {
28            $pesan_error .= "Ukuran file melebihi 700KB <br>";
```

```

29         $upload_gagal = true;
30     }
31     // pindahkan file upload jika semuanya OK
32     if (!upload_gagal AND move_uploaded_file($tmp,$path_file)){
33         $pesan_error = "File sukses di upload";
34     }
35     else {
36         $pesan_error .= "File gagal di upload";
37     }
38 }
39 }
40 ?>
41 <!DOCTYPE html>
42 <html>
43 <head>
44     <meta charset="UTF-8">
45     <title>Belajar PHP</title>
46 </head>
47 <body>
48 <h1>Upload File</h1>
49 <?php if (!empty($pesan_error)) {echo "<p>$pesan_error</p>";} ?>
50     <form action="index.php" method="post" enctype="multipart/form-data">
51         <input type="hidden" name="MAX_FILE_SIZE" value="1000000">
52         <p>Upload File: <input type="file" name="file_upload"></p>
53         <input type="submit" name="submit" value="Proses Upload">
54     </form>
55 </body>
56 </html>

```



Gambar: Ukuran file upload melebihi 700KB

Disini saya menambahkan 1 lagi validasi, yakni jika `$ukuran_file > 716800`, tampilkan pesan error lalu set variabel `$upload_gagal = true`.

## 21.9 Membatasi Jenis File Upload

Validasi berikutnya adalah membatasi jenis tipe file yang boleh diupload. Sebagai contoh, form biodata akan lebih menarik jika user bisa mengupload gambar profil nya sendiri. Tapi bagaimana jika ada yang iseng dan mengupload file mp3?

Terdapat beberapa alternatif solusi, mulai dari yang sederhana namun mudah di bobol, hingga yang cukup rumit tapi lebih aman.

Solusi paling sederhana dan yang paling mudah adalah menggunakan salah satu fitur terbaru di HTML5, yakni atribut `accept`. Atribut `accept` bisa diisi dengan format apa yang dibolehkan, apakah file jpg, png, pdf, mp3, dll. Atribut ini ditempatkan ke dalam tag `<input type="upload">`.

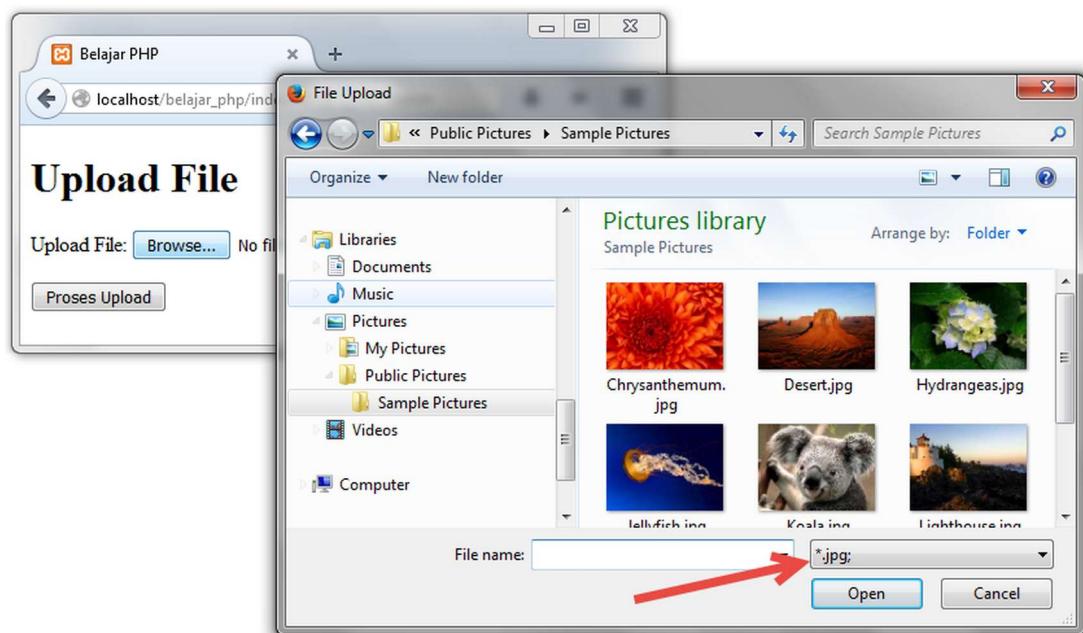


Jika anda sudah membeli eBook **HTML Uncover**, saya telah membahas sekilas tentang atribut ini di dalam bab tentang Form HTML5.

Sebagai contoh, untuk membatasi pilihan file upload hanya untuk gambar dengan format jpg, saya bisa menggunakan tag berikut:

```
<input type="file" name="file_upload" id="file_upload" accept=".jpg">
```

Ketika kode HTML tersebut dijalankan, jendela explorer file hanya akan menampilkan file-file yang berakhiran .jpg, seperti gambar berikut:



Gambar: File yang ditampilkan sudah ter-filter hanya untuk file .jpg

Bagaimana untuk 2 atau lebih file format? Berikut perubahannya:

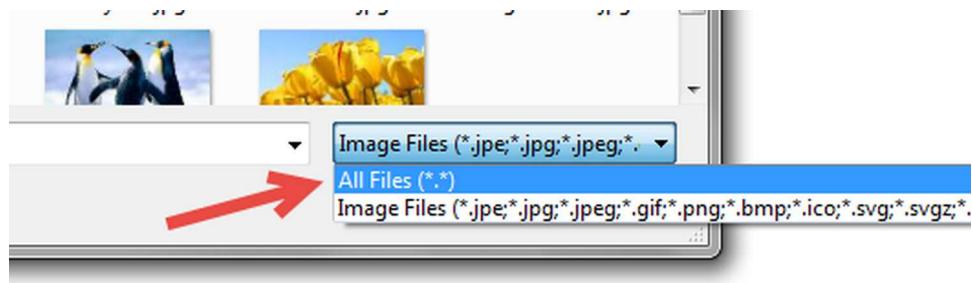
```
<input type="file" name="file_upload" id="file_upload"  
accept = ".jpg, .png, .gif">
```

Untuk file musik, bisa terdiri dari .mp3, .aac, .ogg, .m4a, atau .wav, sehingga bisa ditulis menjadi:

```
<input type="file" name="file_upload" id="file_upload" accept=".mp3, .aac, .ogg, .m4a, atau .wav">
```

Khusus untuk 3 tipe format yang sering digunakan, kita juga bisa menulis `accept="audio/*"`, `accept="video/*"` atau `accept="image/*"`. Web browser otomatis akan menyesuaikan format file untuk audio, video dan gambar.

Akan tetapi seperti layaknya validasi dari HTML, ini semua bisa diubah dengan mudah dari sisi pengguna. Ketika memilih file, terdapat pilihan untuk mengubah jenis file:



Gambar: Kita bisa menukar format pada saat memilih file upload

Penambahan atribut accept lebih kepada fitur, daripada sebuah sarana validasi. Akan jauh lebih aman jika kita melakukan validasi jenis file di server menggunakan PHP.

Jadi, bagaimana cara membatasi jenis file dari sisi PHP?

Kita bisa menggunakan informasi yang ada di dalam variabel `$_FILES["file_upload"]["type"]`. Variabel ini menampung jenis MIME type dari file yang diupload.

## Mengenal MIME Type

**MIME type** adalah sebuah standar internet untuk membedakan tipe dan jenis dokumen. MIME type terdiri dari 2 bagian: **type** dan **sub type**. Sebagai contoh, MIME type untuk gambar dengan format **jpg** adalah **image/jpeg**. **Image** merupakan **type**, sedangkan **jpeg** adalah **subtype**. Contoh lain, MIME type untuk audio dengan format **mp3** adalah **audio/mpeg**.

Perhatikan bahwa format file tidak selalu “pas” dengan MIME type, contohnya, file mp3 yang memiliki MIME type **audio/mpeg**. File terbaru micosoft word yang berakhiran **docx** memiliki MIMEtype: *application/vnd.openxmlformats-officedocument.wordprocessingml.document*.

Untuk penjelasan lebih lanjut mengenai MIME type bisa dibaca di [wikipedia](#)<sup>a</sup>. Daftar lengkap MIME type bisa dilihat dari [hul.harvard.edu](#)<sup>b</sup>.

<sup>a</sup>[http://en.wikipedia.org/wiki/Internet\\_media\\_type](http://en.wikipedia.org/wiki/Internet_media_type)

[4http://hul.harvard.edu/ois/systems/wax/wax-public-help/mimetypes.htm](http://hul.harvard.edu/ois/systems/wax/wax-public-help/mimetypes.htm)

Dengan menggunakan informasi dari `$_FILES["file_upload"]["type"]`, saya bisa membuat validasi seperti berikut ini:

```
1 <?php
2   $mime_boleh= array("image/jpeg", "image/gif", "image/png");
3   if (!in_array($_FILES["file_upload"]["type"], $mime_boleh))
4   {
5     $pesan_error = "Mohon upload file gambar (.gif, .png, .jpg) <br>";
6     $upload_gagal = true;
7   }
8 ?>
```

Artinya jika MIME type dari file yang diupload tidak cocok dengan salah satu isi variabel `$mime_boleh`, batalkan proses pemindahan file.

Selain dari MIME type, pembatasan file ini juga bisa dengan melihat nama file. Variabel `$_FILES["file_upload"]["name"]` berisi nama asli dari file yang diupload, lengkap dengan extension-nya.

Untuk mengambil bagian extension ini, kita bisa menggunakan fungsi `pathinfo()`. Fungsi `pathinfo()` berguna untuk memecah sebuah alamat file (*path*) menjadi beberapa bagian, berikut contohnya:

```
1 <?php
2   $alamat_file="belajar_php/folder_upload/laguku.mp3";
3
4   echo "<pre>";
5   print_r(pathinfo($alamat_file));
6   echo "</pre>";
7 ?>
```

```
Array
(
    [dirname] => belajar_php/folder_upload
    [basename] => laguku.mp3
    [extension] => mp3
    [filename] => laguku
)
```

Fungsi `pathinfo()` akan mengembalikan sebuah array. Jika saya memiliki file dengan nama `laguku.mp3`, saya bisa “memotong” nama file ini untuk mencari extensionnya dengan kode berikut:

```
1 <?php
2 $nama_file="laguku.mp3";
3 $file_info=pathinfo($nama_file);
4 echo $file_info["extension"]; // mp3
5 ?>
```

Kembali kepada form upload, saya bisa membuat validasi seperti berikut:

```
1 <?php
2 $file_boleh= array("jpg", "gif", "png");
3 $filename = $_FILES['file_upload']['name'];
4 $file_info = pathinfo($filename);
5
6 if (!in_array($file_info["extension"],$file_boleh))
7 {
8     $pesan_error = "Mohon upload file gambar (.gif, .png, .jpg )<br>";
9     $upload_gagal = true;
10 }
11 ?>
```

Validasi ini mirip seperti pengecekan MIME type. Silahkan anda tambahkan kode diatas kedalam kode program form upload, dan coba upload selain file gambar. PHP akan mengeluarkan pesan error: "*Mohon upload file gambar (.gif, .png, .jpg )*".

Akan tetapi, validasi seperti ini juga masih memiliki kelemahan. Bagaimana jika seseorang berbuat iseng dengan me-rename file *virus.exe* menjadi *virus.jpg*? File tersebut sebenarnya tidak berubah, yang berubah hanya nama dan extension filenya saja. Kedua validasi yang kita buat menggunakan *MIME Type* dan *file extension* akan bobol. Keduanya melihat bahwa file tersebut hanyalah sebuah file gambar (jpg). Silahkan anda coba sendiri.

Ini hanya sebagian kecil dari masalah tentang "PHP Security", yakni bagaimana cara mengamankan setiap kemungkinan yang bisa terjadi. Seseorang bisa mengupload *virus.jpg* tadi, lalu merubah kembali nama file tersebut dari dalam sistem. Dan.... situs kita bobol!

Saya tidak akan membahas cara pengamanan dari serangan seperti ini, karena memang cukup rumit dan butuh pengetahuan PHP lanjutan. Tapi khusus untuk file gambar, kita bisa menggunakan 1 lagi senjata validasi, yakni menggunakan fungsi *getimagesize()*.

Fungsi *getimagesize()* termasuk salah satu fungsi PHP untuk memanipulasi gambar. Yup betul, dengan PHP kita bisa memotong, menyatukan, bahkan membuat gambar sendiri secara dinamis. Pernah melihat gambar CHAPTA? Gambar angka yang susah dibaca ini bisa dibuat dengan PHP.



Cara mengolah gambar dengan PHP saya rencanakan akan dibahas di eBook PHP Lanjutan.

Sama seperti fungsi *pathinfo()*, fungsi *getimagesize()* juga menghasilkan informasi tentang sebuah file gambar dalam bentuk array. Mari kita coba. Silahkan jalankan kode berikut dan upload sebuah gambar.

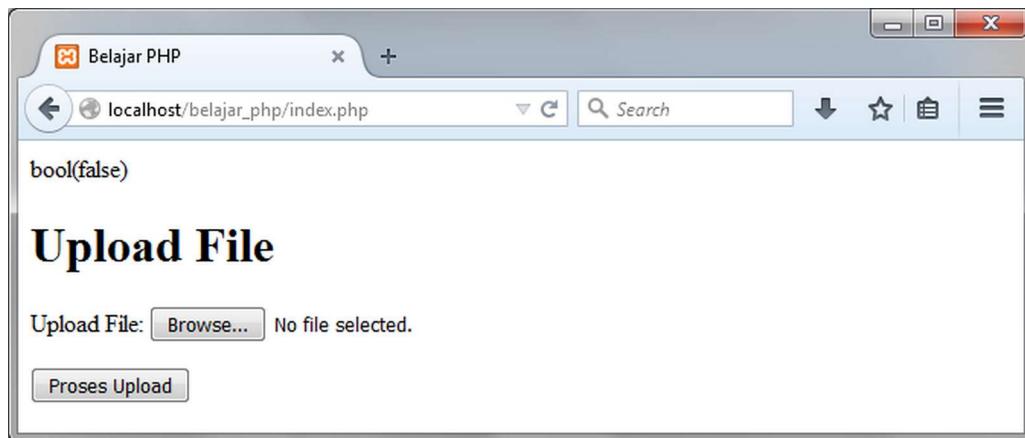
```
1 <?php
2 // cek apakah form telah di submit
3 if (isset($_POST["submit"])) {
4     $check = getimagesize($_FILES["file_upload"]["tmp_name"]);
5     var_dump($check);
6
7     echo "<pre>";
8     print_r($check);
9     echo "</pre>";
10 }
11 ?>
12 <!DOCTYPE html>
13 <html>
14 <head>
15     <meta charset="UTF-8">
16     <title>Belajar PHP</title>
17 </head>
18 <body>
19 <h1>Upload File</h1>
20     <form action="index.php" method="post" enctype="multipart/form-data">
21         <p>Upload File:
22             <input type="file" name="file_upload" id="file_upload">
23         </p>
24         <input type="submit" name="submit" value="Proses Upload">
25     </form>
26 </body>
27 </html>
```



Gambar: Hasil dari fungsi getimagesize() pada file upload

Kode program diatas akan menampilkan informasi dari pemanggilan fungsi `getimagesize()` dari variabel `$_FILES["file_upload"]["tmp_name"]`. Informasi ini mencakup ukuran gambar, channel yang digunakan, hingga MIME Type.

Walaupun tampak ‘mubazir’, saya menggunakan fungsi `var_dump()` dan fungsi `print_r()` secara bersamaan untuk percobaan berikutnya. Sekarang, silahkan anda upload file selain gambar, misalkan `.doc` atau `.mp3`. Bagaimana hasilnya?



Gambar: Hasil: boolean false, karena yang diupload bukan file gambar

Seperti yang terlihat, fungsi `getimagesize()` mengembalikan nilai `FALSE` jika yang diperiksa bukan file gambar.

Bagaimana dengan file yang “seolah-olah” gambar? Anda bisa coba sendiri dengan mengubah extension sebuah file menjadi jpg, misalnya `test.zip` menjadi `test.jpg`. File seperti ini akan lolos jika menggunakan validasi *MIME type* dan *pathinfo()*. Bagaimana dengan `getimagesize()`?

Hasilnya akan tetap `FALSE`!

Dengan demikian kode program yang kita rancang sudah cukup baik dalam menvalidasi file gambar. Berikut contoh kode program gabungan validasi file yang memiliki nama sama, validasi ukuran file, dan validasi format gambar:

```

1  <?php
2  // cek apakah form telah di submit
3  if (isset($_POST["submit"])) {
4      // form telah disubmit, cek apakah ada error
5      $error = $_FILES["file_upload"]["error"];
6
7      if ($error === 0){
8
9          // siapkan variabel untuk pesan error
10         $pesan_error = "";
11
12         // siapkan variabel untuk pemindahan file
13         $nama_folder="folder_upload";
14         $tmp = $_FILES["file_upload"]["tmp_name"];

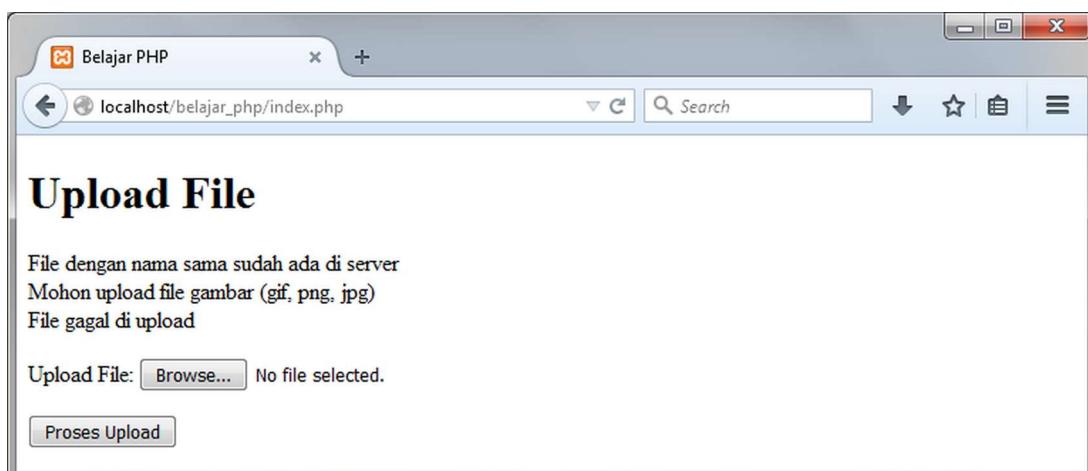
```

```
15     $nama_file = $_FILES["file_upload"]["name"];
16     $path_file = "$nama_folder/$nama_file";
17     $upload_gagal = false;
18
19     // cek apakah terdapat file dengan nama yang sama
20     if (file_exists($path_file)) {
21         $pesan_error = "File dengan nama sama sudah ada di server <br>";
22         $upload_gagal = true;
23     }
24
25     // cek apakah ukuran file tidak melebihi 700KB (716800 byte)
26     $ukuran_file = $_FILES["file_upload"]["size"];
27     if ($ukuran_file > 716800) {
28         $pesan_error .= "Ukuran file melebihi 700KB <br>";
29         $upload_gagal = true;
30     }
31
32     // cek apakah yang diupload adalah file gambar
33     $check = getimagesize($_FILES["file_upload"]["tmp_name"]);
34     if ($check === false) {
35         $pesan_error .= "Mohon upload file gambar (gif, png, jpg)<br>";
36         $upload_gagal = true;
37     }
38
39     // pindahkan file upload jika semuanya OK
40     if (!$upload_gagal AND move_uploaded_file($tmp,$path_file)) {
41         $pesan_error = "File sukses di upload";
42     }
43     else {
44         $pesan_error .= "File gagal di upload";
45     }
46 }
47 }
48 ?>
49 <!DOCTYPE html>
50 <html>
51 <head>
52     <meta charset="UTF-8">
53     <title>Belajar PHP</title>
54 </head>
55 <body>
56 <h1>Upload File</h1>
57 <?php if (!empty($pesan_error)) {echo "<p>$pesan_error</p>";} ?>
58 <form action="index.php" method="post" enctype="multipart/form-data">
59     <input type="hidden" name="MAX_FILE_SIZE" value="1000000">
60     <p>Upload File:</p>
```

```

61      <input type="file" name="file_upload" accept="image/*">
62  </p>
63  <input type="submit" name="submit" value="Proses Upload">
64 </form>
65 </body>
66 </html>

```



Gambar: Beragam validasi untuk file upload

Dalam kode program diatas, kita telah melakukan validasi untuk 3 masalah: nama file yang sama, ukuran file melewati batas yang ditentukan, serta format file. Walaupun kode ini tidak 100% aman, tapi sudah cukup kuat mencegah seseorang yang “jahil”.

## 21.10 Multiple Upload File

Untuk form yang lebih kompleks, kadang 1 file upload belum cukup. Untuk mengupload 2 atau lebih file, terdapat 2 alternatif: membuat beberapa tag `<input type="file">`, atau menambahkan atribut **multiple**. Kita akan membahas keduanya.

Cara pertama adalah dengan membuat beberapa tag `<input type="file">`, seperti contoh berikut:

```

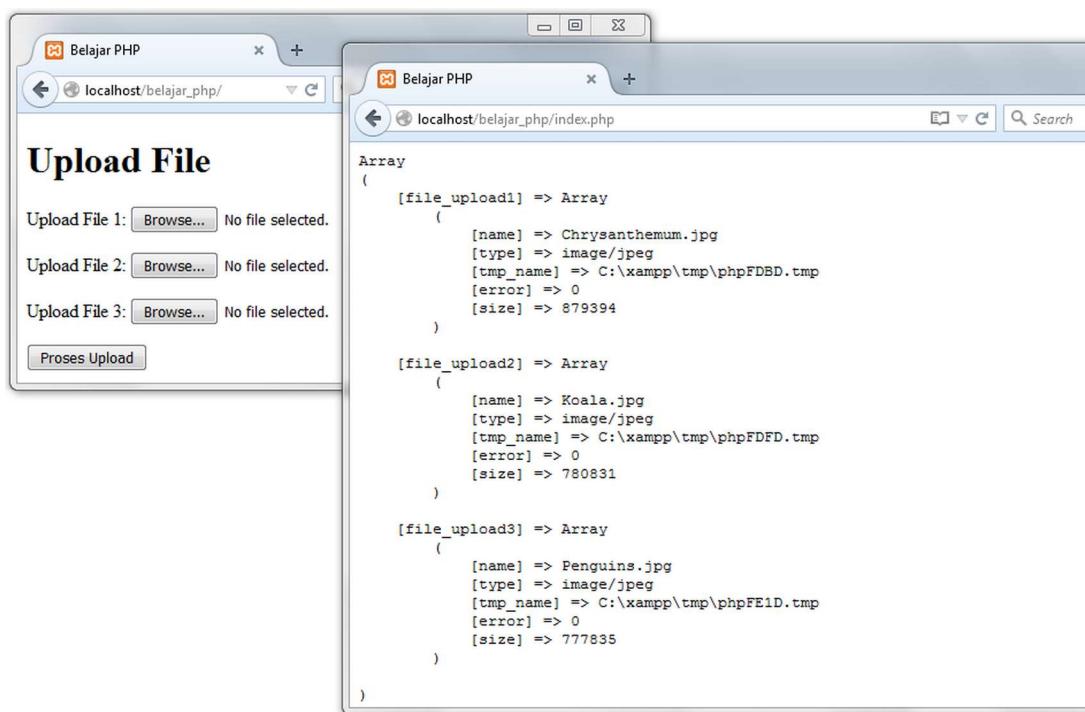
1 <?php
2 // cek apakah form telah di submit
3 if (isset($_POST["submit"])) {
4
5     // tampilkan isi form
6     echo "<pre>";
7     print_r($_FILES);
8     echo "</pre>";
9
10    // ... pemrosesan form disini..
11    // ... pemrosesan form disini..

```

```

12     // ... pemrosesan form disini..
13 }
14 ?>
15 <!DOCTYPE html>
16 <html>
17 <head>
18     <meta charset="UTF-8">
19     <title>Belajar PHP</title>
20 </head>
21 <body>
22 <h1>Upload File</h1>
23 <form action="index.php" method="post" enctype="multipart/form-data">
24     <p>Upload File 1: <input type="file" name="file_upload1"></p>
25     <p>Upload File 2: <input type="file" name="file_upload2"></p>
26     <p>Upload File 3: <input type="file" name="file_upload3"></p>
27     <input type="submit" name="submit" value="Proses Upload">
28 </form>
29 </body>
30 </html>

```



Gambar: Pembuatan form dengan 3 tag input type upload

Disini saya membuat 3 buat tag `<input type="file">`. Masing-masing dengan nama `file_upload1`, `file_upload2` dan `file_upload3`. Ketika form di-submit saya menampilkan isi variabel `$_FILES` dengan fungsi `print_r()`.

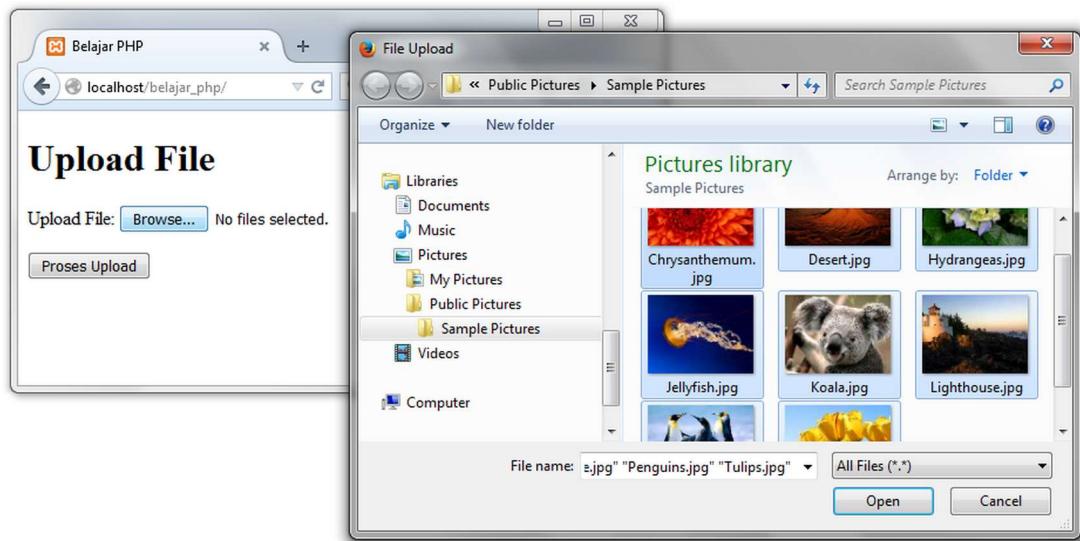
Tidak ada yang baru disini. Setiap isian form upload memiliki nama masing-masing. Hanya saja, untuk pemrosesan form kita harus membuat 3 kali validasi serta 3 kali pemanggilan fungsi

`move_uploaded_file()`. Saya tidak akan membahas kode programnya, tapi saya yakin anda sudah bisa merancangnya sendiri.

Cara kedua untuk mengupload banyak file adalah dengan menambahkan atribut **multiple** ke dalam tag `<input type="file">`, seperti berikut ini:

```
<input type="file" name="file_upload" multiple>
```

Hasilnya, sekarang kita bisa memilih lebih dari 1 file dari jendela *browse file* dengan cara memblok atau menekan tombol **CRTL + klik**, seperti gambar berikut:



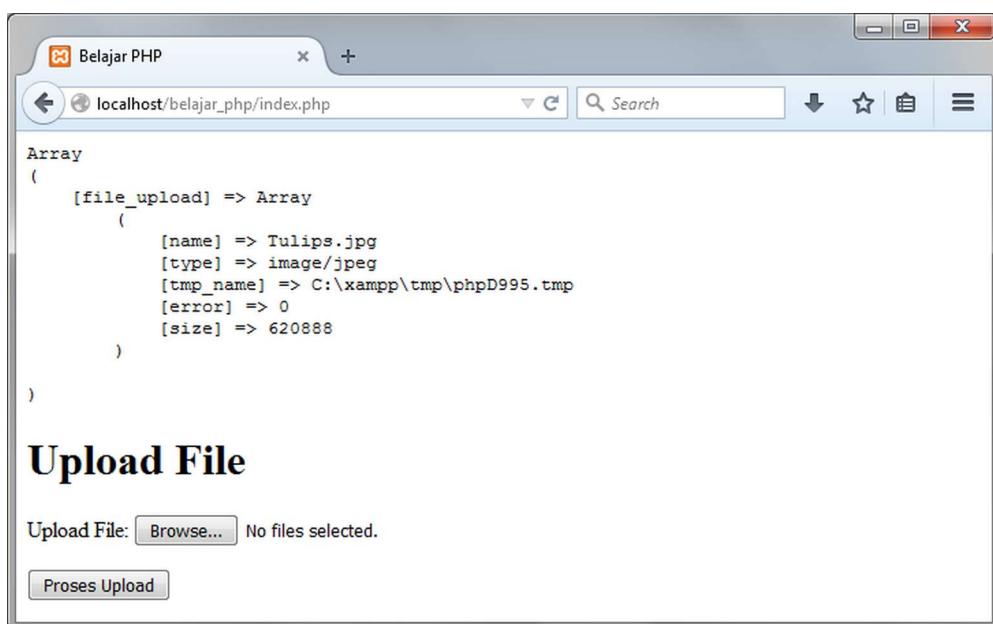
Gambar: Upload multiple file dengan cara memblok file yang akan diupload

Bagaimana cara pemrosesan file upload seperti ini? Mari kita coba:

```

1  <?php
2  // cek apakah form telah di submit
3  if (isset($_POST["submit"])) {
4
5      // tampilkan isi form
6      echo "<pre>";
7      print_r($_FILES);
8      echo "</pre>";
9
10     // ... pemrosesan form disini..
11     // ... pemrosesan form disini..
12     // ... pemrosesan form disini..
13 }
14 ?>
15 <!DOCTYPE html>
16 <html>
17 <head>
```

```
18  <meta charset="UTF-8">
19  <title>Belajar PHP</title>
20 </head>
21 <body>
22 <h1>Upload File</h1>
23 <form action="index.php" method="post" enctype="multipart/form-data">
24   <p>Upload File: <input type="file" name="file_upload" multiple></p>
25   <input type="submit" name="submit" value="Proses Upload">
26 </form>
27 </body>
28 </html>
```



Gambar: Berapa banyak file yang terupload?

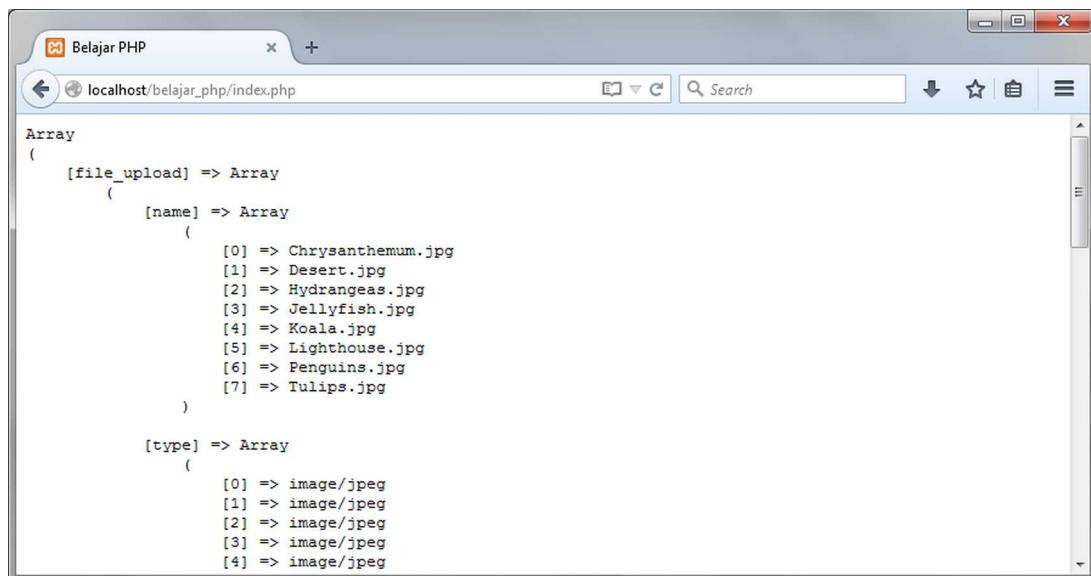
Hasil diatas saya dapat ketika memilih 8 file sakaligus untuk diupload. Tapi, seperti yang terlihat, hanya 1 file yang tampil dari perintah `print_r($_FILES)`. Kemana sisanya?

Ini terjadi karena setiap file yang diupload akan menimpa file upload sebelumnya. Dengan kata lain, variabel `$_FILES["file_upload"]` akan terus tertimpa dengan file-file yang diupload. Jadi bagaimana solusinya?

Caranya adalah dengan mengubah `$_FILES["file_upload"]` menjadi sebuah array. Sehingga masing-masing file upload akan berada di dalam variabel `$_FILES["file_upload"][]` untuk file pertama, `$_FILES["file_upload"][]` untuk file kedua, dan seterusnya.

Untuk membuatnya, saya akan menukar atribut `name` dari `file_upload` menjadi `file_upload[]`. Perhatikan bahwa nama `file_upload[]` berarti saya ingin membuat `$_FILES["file_upload"]` menjadi sebuah array. Silahkan ubah tag `<input type="file">` menjadi seperti berikut, dan kembali jalankan kode program:

```
<input type="file" name="file_upload[]" multiple>
```



Gambar: Hasil upload multiple file sebagai array 3 Dimensi

Dapat terlihat bahwa variabel `$_FILES["file_upload"]` telah menjadi array 3 dimensi. Untuk mengakses nama file dari masing-masing file upload, caranya dari variabel `$_FILES["file_upload"] ["name"] [0]`, `$_FILES["file_upload"] ["name"] [1]`, dst hingga `$_FILES["file_upload"] ["name"] [8]`.

Selanjutnya, bagaimana cara memproses file-file ini? Sekilas terlihat rumit, tapi sebenarnya cukup sederhana. Kita tinggal mengecek ada berapa file yang diupload, lalu membuat sebuah perulangan untuk memindahkan semua file. Berikut kode programnya:

```

1  <?php
2  // cek apakah form telah di submit
3  if (isset($_POST["submit"])) {
4
5      // hitung jumlah file upload
6      $jumlah_file = count($_FILES["file_upload"] ["name"]);
7
8      // pindahkan seluruh file ke folder_upload
9      $nama_folder = "folder_upload";
10     for ($i=0; $i < $jumlah_file; $i++)
11     {
12         $tmp = $_FILES["file_upload"] ["tmp_name"] [$i];
13         $nama_file = $_FILES["file_upload"] ["name"] [$i];
14         move_uploaded_file($tmp, "$nama_folder/$nama_file");
15     }
16 }
17 ?>
18 <!DOCTYPE html>
```

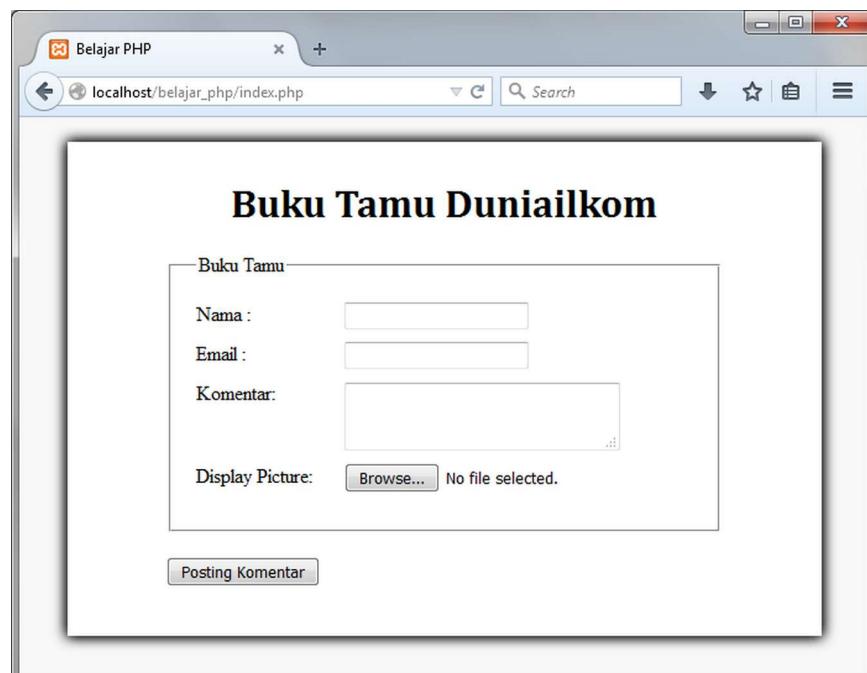
```
19 <html>
20 <head>
21   <meta charset="UTF-8">
22   <title>Belajar PHP</title>
23 </head>
24 <body>
25 <h1>Upload File</h1>
26   <form action="index.php" method="post" enctype="multipart/form-data">
27     <p>Upload File: <input type="file" name="file_upload[]" multiple></p>
28     <input type="submit" name="submit" value="Proses Upload">
29   </form>
30 </body>
31 </html>
```

Silahkan anda pelajari sebentar kode diatas. Saya mencari jumlah file upload dengan menggunakan fungsi `count($_FILES["file_upload"]["name"])`, lalu membuat perulangan FOR untuk memindahkan file satu persatu.

## 21.11 Case study: Buku Tamu DuniaIlkom

Menutup bab ini, saya akan membuat sebuah studi kasus. Kali ini berupa sebuah buku tamu sederhana. Disini saya menampilkan form yang terdiri dari beberapa isian. Salah satunya berupa file upload untuk gambar tamu (*display picture*).

Berikut tampilan awal dari buku tamu duniaIlkom:



Gambar: Tampilan form "buku tamu duniaIlkom"

Terlihat saya memiliki 4 isian form: **nama**, **email**, **komentar**, dan **display picture** (upload gambar). Dari tampilan design, ini sama persis dengan studi kasus bab sebelumnya: *Pembelian Buku DuniaIlkom*. Berikut kode HTML dan CSS untuk membuat tampilan diatas:

**index.php**

---

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar PHP</title>
6   <style>
7     body {
8       background-color: #F8F8F8;
9     }
10    div.container {
11      width: 450px;
12      padding: 10px 80px 30px;
13      background-color: white;
14      margin: 20px auto;
15      box-shadow: 1px 0px 10px, -1px 0px 10px ;
16    }
17    h1 {
18      text-align: center;
19      font-family: Cambria, "Times New Roman", serif;
20    }
21    p {
22      margin:0;
23    }
24    fieldset {
25      padding:20px;
26    }
27    input, textarea {
28      margin-bottom:10px;
29    }
30    label {
31      width:110px;
32      float:left;
33      margin-right:10px;
34    }
35    .error {
36      background-color: #FFCEC;
37      padding: 10px 15px;
38      margin: 0 0 20px 0;
39      border: 1px solid red;
40      box-shadow: 1px 0px 3px red ;
41    }
```

```
42  </style>
43  </head>
44  <body>
45  <div class="container">
46  <h1>Buku Tamu DuniaIlkom</h1>
47  <form action="index.php" method="post" enctype="multipart/form-data" >
48  <fieldset>
49  <legend>Buku Tamu</legend>
50  <p>
51  <label for="nama">Nama : </label>
52  <input type="text" name="nama" id="nama" >
53  </p>
54  <p>
55  <label for="email">Email : </label>
56  <input type="text" name="email" id="email" >
57  </p>
58  <p>
59  <label for="komentar">Komentar: </label>
60  <textarea name="komentar" cols="25" name="komentar"></textarea>
61  </p>
62  <p>
63  <label for="file_upload">Display Picture: </label>
64  <input type="hidden" name="MAX_FILE_SIZE" value="1000000" >
65  <input type="file" name="file_upload" id="file_upload" accept="image/*" >
66  </p>
67  </fieldset>
68  <br>
69  <p>
70  <input type="submit" name="submit" value="Posting Komentar" >
71  </p>
72  </form>
73
74  </div>
75  </body>
76  </html>
```

---

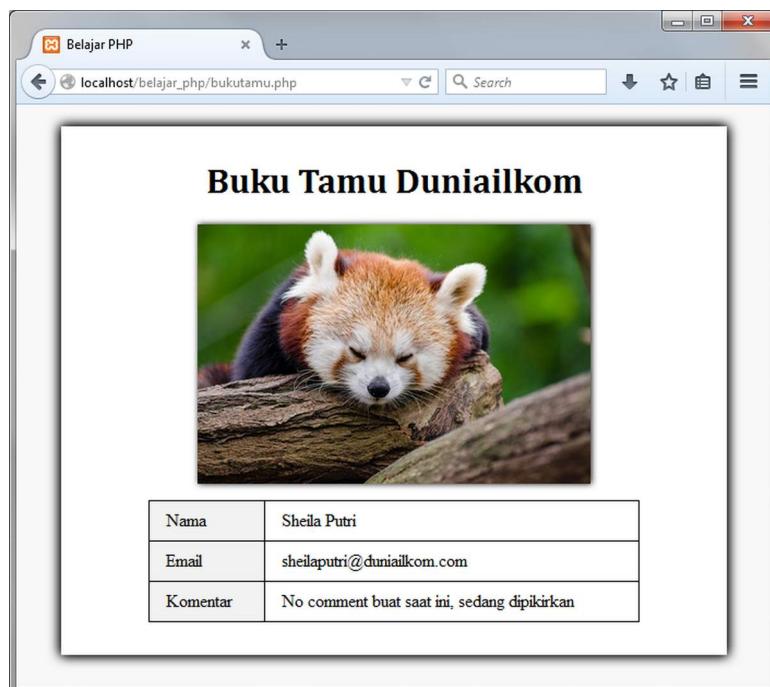
Halaman ini hanya terdiri dari kode HTML dan CSS saja, belum menggunakan PHP sama sekali. Berikutnya, saya akan mempersiapkan halaman akhir dari form ini, yakni dimana komentar ditampilkan lengkap dengan display picture yang diupload. Saya akan menggunakan metode `include()` yang sama seperti case study bab sebelumnya. Berikut kode untuk halaman `buku-tamu.php`:

**bukutamu.php**

---

```
1 <?php
2     $nama = "Sheila Putri";
3     $email = "sheilaputri@duniaikom.com";
4     $komentar = "No comment buat saat ini, sedang dipikirkan";
5     $nama_file = "redpanda.jpg";
6 ?>
7 <!DOCTYPE html>
8 <html>
9 <head>
10    <meta charset="UTF-8">
11    <title>Belajar PHP</title>
12    <style>
13        body {
14            background-color: #F8F8F8;
15        }
16        div.container {
17            width: 450px;
18            padding: 10px 80px 30px;
19            background-color: white;
20            margin: 20px auto;
21            box-shadow: 1px 0px 10px, -1px 0px 10px ;
22        }
23        h1 {
24            text-align: center;
25            font-family: Cambria, "Times New Roman", serif;
26        }
27        table {
28            border-collapse: collapse;
29            border-spacing: 0;
30            border: 1px black solid;
31            width: 100%
32        }
33        th, td {
34            padding: 8px 15px;
35            border: 1px black solid;
36        }
37        td:first-child {
38            background-color: #F2F2F2;
39        }
40        img {
41            width: 80%;
42            display: block;
43            margin: 0px auto;
44            margin-bottom: 15px;
```

```
45      box-shadow: 1px 0px 5px, -1px 0px 5px #FFF;
46  }
47 </style>
48 </head>
49 <body>
50 <div class="container">
51 <h1>Buku Tamu DuniaIlkom</h1>
52 <table>
53 ">
54 <tr>
55   <td>Nama</td> <td><?php echo $nama; ?></td>
56 </tr>
57 <tr>
58   <td>Email</td> <td><?php echo $email; ?></td>
59 </tr>
60 <tr>
61   <td>Komentar</td> <td><?php echo $komentar; ?></td>
62 </tr>
63 </table>
64 </body>
65 </html>
```



Gambar: Tampilan halaman hasil buku tamu duniaIlkom

Dibaris paling awal saya membuat beberapa variabel dummy. Variabel asli nantinya akan berasal dari form. Khusus untuk *display picture*, saya juga sudah menempatkan sebuah gambar **redpanda.jpg** di **folder\_upload**.

Karena tampilan diatas sudah sesuai dengan keinginan saya, kode PHP dummy sudah bisa dihapus. File bukutamu.php ini selanjutnya akan dipanggil dari dalam file index.php (menggunakan metode *include*).

Sekarang, kita masuk ke pekerjaan inti, yakni membuat validasi serta pemrosesan form. Bisakah anda merancangnya sendiri? Jika study case bab sebelumnya anda ‘nyontek’ menggunakan kode program yang saya buat, inilah saatnya untuk test kemampuan anda :)

Disini kita hanya perlu membuat validasi untuk `nama`, `email` dan `file upload`. Untuk isian `komentar` tidak akan saya tambahkan validasi apapun. Dengan demikian, komentar boleh dikosongkan.

Khusus untuk file upload, saya ingin membuat validasi lengkap, mulai dari cek kode error variabel `$_FILES["file_upload"]["error"]`, memastikan tidak ada nama file yang sama, ukuran file harus < 1MB, serta file yang diupload haruslah file gambar. Semua ini sudah kita pelajari sepanjang bab ini.

Baiklah, berikut kode lengkap case study **Buku Tamu DuniaIlkom**:

index.php

```
1 <?php
2 // cek apakah form telah di submit
3 if (isset($_POST["submit"])) {
4     // form telah disubmit, proses data
5
6     // ambil nilai form kecuali file upload
7     $nama = htmlentities(strip_tags(trim($_POST["nama"])));
8     $email = htmlentities(strip_tags(trim($_POST["email"])));
9     $komentar = htmlentities(strip_tags(trim($_POST["komentar"])));
10
11    // siapkan variabel untuk menampung pesan error
12    $pesan_error="";
13
14    // cek apakah "nama" sudah diisi atau tidak
15    if (empty($nama)) {
16        $pesan_error .= "Nama belum diisi <br>";
17    }
18
19    // cek apakah "email" sudah diisi atau tidak
20    if (empty($email)) {
21        $pesan_error .= "Email belum diisi <br>";
22    }
23    // email harus sesuai dengan format
24    elseif (!preg_match("/^.+@.+\\..+/", $email) ) {
25        $pesan_error .= "Format email tidak sesuai <br>";
26    }
27
28    // cek apakah gambar berhasil di upload
```

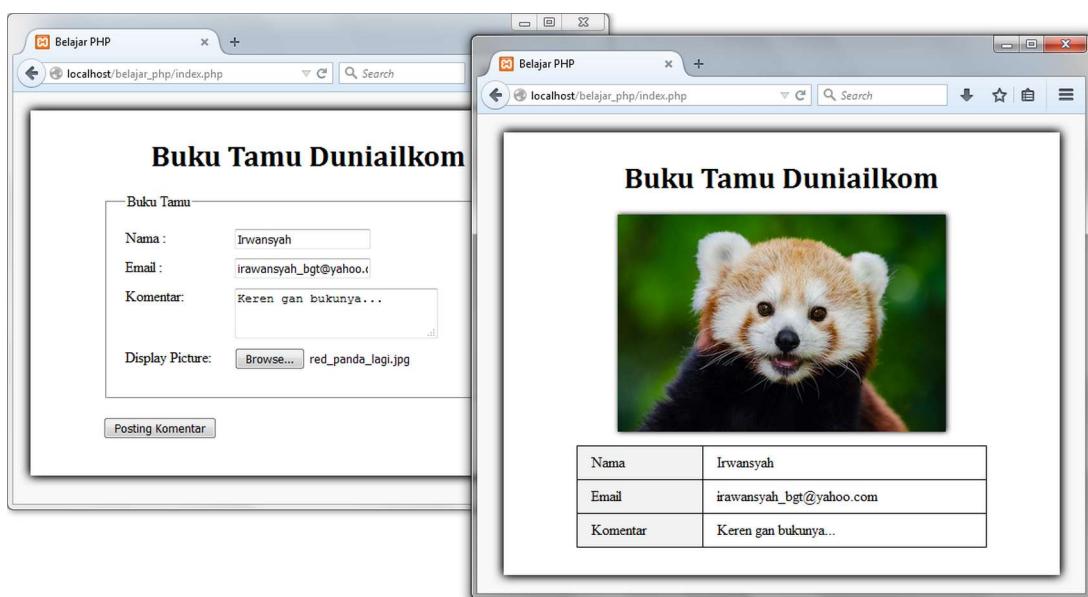
```
29     $upload_error = $_FILES["file_upload"]["error"];
30     if ($upload_error != 0){
31         // gambar gagal di upload siapkan pesan error
32         $arr_upload_error = array(
33             1 => 'Ukuran file melewati batas maksimal',
34             2 => 'Ukuran file melewati batas maksimal 1MB',
35             3 => 'File hanya ter-upload sebagian',
36             4 => 'Tidak ada file yang terupload',
37             6 => 'Server Error',
38             7 => 'Server Error',
39             8 => 'Server Error',
40         );
41         $pesan_error .= $arr_upload_error[$upload_error];
42     }
43     else {
44         // tidak ada error, masuk ke validasi file upload berikutnya
45         // periksa apakah ada file dengan nama yg sama
46         $nama_folder = "folder_upload";
47         $nama_file = $_FILES["file_upload"]["name"];
48         $path_file = "$nama_folder/$nama_file";
49
50         if (file_exists($path_file)) {
51             $pesan_error .= "File dengan nama sama sudah ada di server <br>";
52         }
53
54         // cek apakah ukuran file tidak melebihi 1MB(1048576 byte)
55         $ukuran_file = $_FILES["file_upload"]["size"];
56         if ($ukuran_file > 1048576) {
57             $pesan_error .= "Ukuran file melebihi 700KB <br>";
58         }
59
60         // cek apakah memang file gambar
61         $check = getimagesize($_FILES["file_upload"]["tmp_name"]);
62         if ($check === false) {
63             $pesan_error .= "Mohon upload file gambar (gif, png, atau jpg )";
64         }
65     }
66
67     // jika lolos validasi, proses form dan tampilkan
68     if ($pesan_error === "") {
69         // pindahkan file ke folder_upload
70         $nama_folder="folder_upload";
71         $tmp = $_FILES["file_upload"]["tmp_name"];
72         $nama_file = $_FILES["file_upload"]["name"];
73         move_uploaded_file($tmp, "$nama_folder/$nama_file");
74     }
```

```
75      // semua OK, tampilkan hasil form dan bye-bye
76      include("bukutamu.php");
77      die();
78  }
79 }
80 else {
81     // form belum disubmit atau halaman ini tampil untuk pertama kali
82     // berikan nilai awal untuk semua isian form
83     $pesan_error = "";
84     $nama = "";
85     $email = "";
86     $komentar="";
87 }
88
89 // cek apakah ukuran file melewati batas post_max_size
90 // ditempatkan disini agar variabel $pesan_error tidak ter-reset
91 if ($_SERVER['REQUEST_METHOD'] == 'POST' && empty($_FILES) && empty($_POST)\
92 ) {
93     $postMax = ini_get('post_max_size');
94     $pesan_error = "Ukuran file melewati batas maksimal ({$postMax}B)";
95 }
96 ?>
97 <!DOCTYPE html>
98 <html>
99 <head>
100    <meta charset="UTF-8">
101    <title>Belajar PHP</title>
102    <style>
103        body {
104            background-color: #F8F8F8;
105        }
106        div.container {
107            width: 450px;
108            padding: 10px 80px 30px;
109            background-color: white;
110            margin: 20px auto;
111            box-shadow: 1px 0px 10px, -1px 0px 10px ;
112        }
113        h1 {
114            text-align: center;
115            font-family: Cambria, "Times New Roman", serif;
116        }
117        p {
118            margin:0;
119        }
120        fieldset {
```

```
121     padding:20px;
122 }
123 input, textare a {
124     margin-bottom:10px;
125 }
126 label {
127     width:110px;
128     float:left;
129     margin-right:10px;
130 }
131 .error {
132     background-color: #FFECEC;
133     padding: 10px 15px;
134     margin: 0 0 20px 0;
135     border: 1px solid red;
136     box-shadow: 1px 0px 3px red ;
137 }
138 </style>
139 </head>
140 <body>
141
142 <div class="container">
143 <h1>Buku Tamu DuniaIlkom</h1>
144 <?php
145     // tampilkan error jika ada
146     if ($pesan_error !== "") {
147         echo "<div class=\"error\">$pesan_error</div>";
148     }
149 ?>
150 <form action="index.php" method="post" enctype="multipart/form-data" >
151 <fieldset>
152 <legend>Buku Tamu</legend>
153 <p>
154     <label for="nama">Nama : </label>
155     <input type="text" name="nama" id="nama" value="<?php echo $nama ?>">
156 </p>
157 <p>
158     <label for="email">Email : </label>
159     <input type="text" name="email" id="email" value="<?php echo $email ?>">
160 </p>
161 <p>
162     <label for="komentar">Komentar: </label>
163     <textarea name="komentar" cols="25" name="komentar">
164         <?php echo $komentar; ?>
165     </textarea>
166 </p>
```

```
167 <p>
168     <label for="file_upload">Display Picture: </label>
169     <input type="hidden" name="MAX_FILE_SIZE" value="1048576">
170     <input type="file" name="file_upload" id="file_upload" accept="image/*">
171 </p>
172 </fieldset>
173 <br>
174 <p>
175     <input type="submit" name="submit" value="Posting Komentar">
176 </p>
177 </form>
178
179 </div>
180
181 </body>
182 </html>
```

---



Gambar: Buku Tamu DuniaIlkom

Kode program diatas lumayan panjang (hampir 200 baris kode program). Sekali lagi, semua teknik yang ada disini sudah kita pelajari. Cara menampilkan pesan kesalahan form juga saya pinjam dari case study “pembelian buku duniaIlkom”.

Silahkan anda pelajari secara perlahan apa saja fungsi dari setiap kode tersebut. Juga test langsung dengan mengupload berbagai file, mulai dari file dengan ukuran melebihi batas maksimum, upload file yang sama berulang kali, hingga test upload file yang bukan gambar.

---

Selesai dengan form, berikutnya kita akan masuk ke pembahasan tentang cookie dan session.

# 22. Cookie dan Session

**Cookie** dan **Session** merupakan “anggota” berikutnya dari superglobals variabel di dalam PHP. Keduanya juga digunakan sebagai sarana input ke dalam kode program. Mari kita bahas dengan lebih detail.

## 22.1 Apa itu Cookie?

Saya yakin anda sudah sering mendengar istilah *cookie*, bukan sejenis kue, tapi **cookie** web browser.

**Cookie** (dan juga **session**) adalah sebuah mekanisme yang digunakan untuk “mengingat” siapa yang sedang mengunjungi website saat ini, apakah berasal dari orang yang sama? apakah user tersebut sudah mengunjungi website ini minggu lalu? atau memang pengunjung baru?

Sebagai contoh, kita ambil situs **facebook**. Pada saat pertama kali mengunjungi [facebook.com](https://www.facebook.com), akan tampil halaman yang berisi form pendaftaran dan form login. Jika sudah mendaftar sebelumnya, tentu saya bisa masuk dari form login.



Gambar: Halaman awal [facebook.com](https://www.facebook.com)

Misalkan kali ini saya baru pertama kali mendaftar. Setelah mengisi form registrasi dan login, saya putuskan untuk menutup web browser (tanpa logout dari facebook), kemudian mematikan komputer.

Keesokan harinya, ketika membuka halaman [facebook.com](https://www.facebook.com), tidak ada lagi form pendaftaran, tapi tampilan halaman profile facebook saya. Hari-hari berikutnya juga demikian. Bagaimana facebook bisa tau bahwa yang mengakses saat ini adalah “saya”?

Mungkin dari alamat komputer (*IP address*). Tapi sepertinya tidak, karena alamat IP bisa saja berubah-ubah. Jika anda mengakses internet dari operator GSM seperti Telkomsel, XL, Tri, dll,

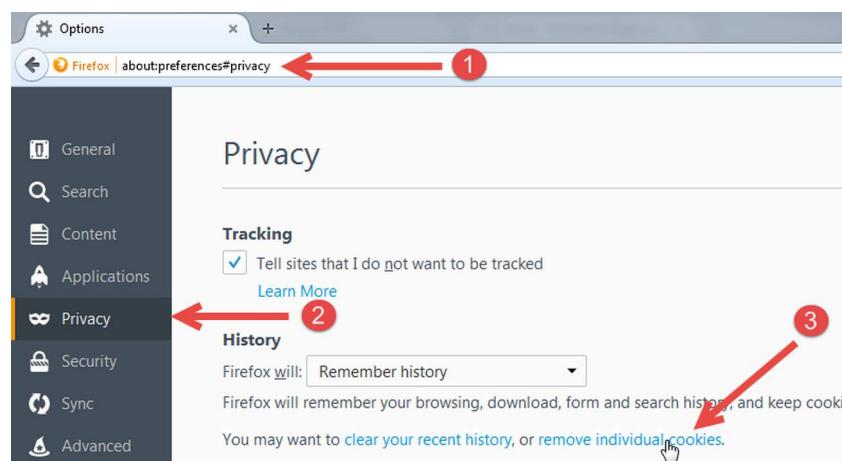
IP address yang digunakan akan selalu bertukar setiap kali modem aktif. Selain itu, alamat IP tersebut juga digunakan bersama-sama.

Dalam kasus diatas, facebook menggunakan **cookie**. **Cookie** berfungsi untuk mengingat dan mengetahui siapa yang sedang mengunjungi website saat ini. Cookie ini berbentuk **data yang “dititipkan” ke dalam web browser**. Ketika saya mengakses situs facebook dari web browser yang sama di kemudian hari, facebook bisa mengetahui bahwa ‘orang ini’ sudah pernah login dengan username “andre”.

Bagaimana ketika saya mengunjungi facebook dari web browser yang berbeda? Karena cookie melekat di web browser, facebook akan berkesimpulan bahwa saya adalah pengunjung baru. Ini juga yang menjadi alasan kita bisa membuka 2 akun facebook yang berbeda pada saat bersamaan di komputer yang sama, asal menggunakan web browser yang berbeda.

Agar lebih jelas apa itu cookies, mari kita lihat ‘penampakannya’ di dalam web browser.

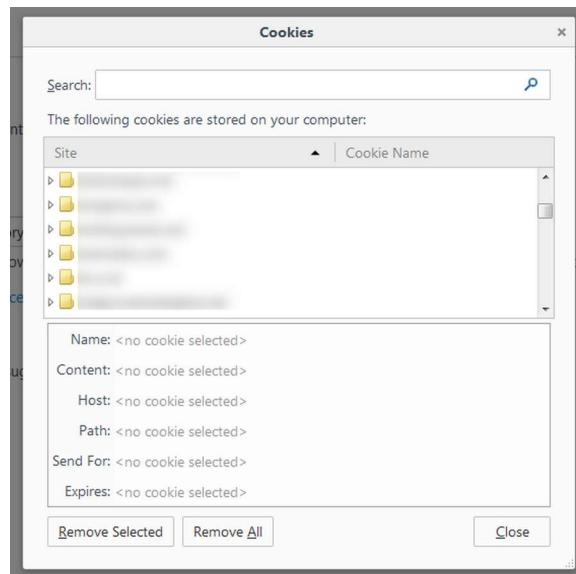
Sebagai contoh, saya akan menggunakan web browser **Mozilla Firefox**. Silahkan buka web browser, lalu ketik alamat: **about:preferences#privacy** di web address (nomor 1).



Gambar: Jendela untuk mengatur setting terkait privacy Mozilla Firefox

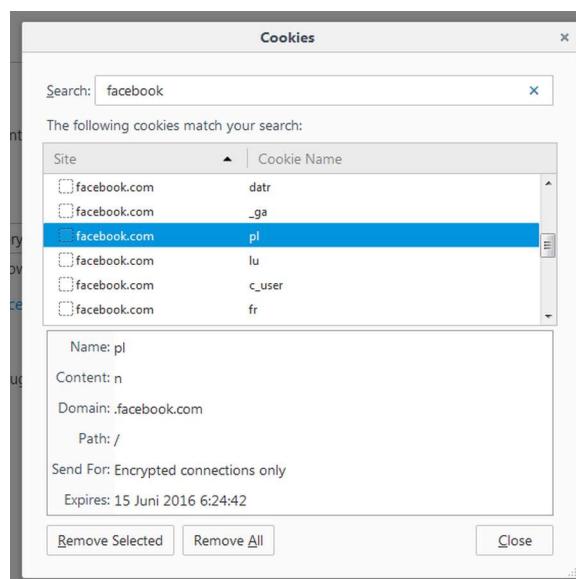
Halaman ini juga bisa diakses dari icon “hamburger” di sudut kanan atas, pilih “**Option**”, atau dari **Menu Tools -> Options**. Kemudian lalu pilih tab **Privacy** (nomor 2).

Dari halaman **Privacy**, klik link **remove individual cookies** (nomor 3). Disini kita bisa mencari dan menghapus cookie yang tersimpan di dalam web browser.



Gambar: Seluruh cookie yang terdapat di web browser

Jika anda sudah menggunakan web browser untuk jangka waktu yang cukup lama, pada jendela ini akan terlihat seluruh website yang menitipkan cookienya. Kita bisa mengetik nama situs di kolom paling atas untuk mencari cookie dari facebook:



Gambar: Cookie dari facebook

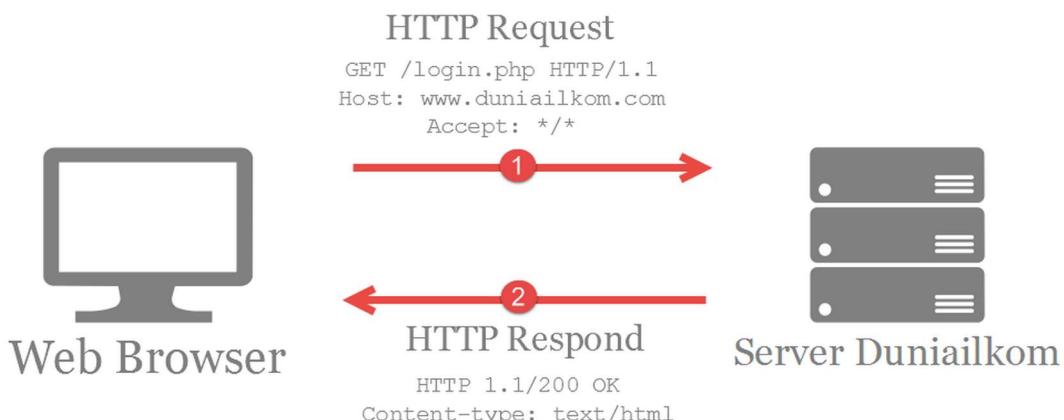
Seperti yang terlihat, terdapat beberapa cookie yang berasal dari situs facebook.com. Kita akan mempelajari semua hal tentang cookie dalam bab ini.

## 22.2 Cara Kerja Cookie

Sebelum masuk ke dalam kode program PHP untuk membuat cookie, saya ingin menjelaskan sedikit **prinsip kerja cookie**. Ini mungkin agak teknis, tapi sangat penting untuk dipahami.

Dari penjelasan sebelumnya dapat terlihat bahwa cookie disimpan di dalam web browser. Pada prakteknya, cookie ini akan selalu “dibawa” di dalam **HTTP Header** (semoga anda masih ingat dengan materi ini).

Berikut adalah alur **request – respond** yang pernah kita pelajari dalam bab 17 tentang **HTTP Header**:



Gambar: Alur request – respond dari web browser ke web server

Pada saat user mengetikkan alamat website ke dalam web browser dan menekan tombol **Enter**, proses permintaan (*request*) akan dikirim oleh web browser kepada web server. Permintaan ini sampaikan dalam bentuk **HTTP header**.

Di dalam HTTP header terdapat informasi seperti nama website, halaman yang diminta, dan lainnya. Dalam gambar diatas, panah no 1 meminta (*request*) halaman **login.php** ke server duniaIlkom yang berada di alamat [www.duniaIlkom.com](http://www.duniaIlkom.com).<sup>1</sup>

Ketika *request* ini sampai ke web server, web server akan memproses kode PHP yang ada di halaman **login.php** lalu membalasnya (*respond*) kepada web browser. Respond yang dikirim juga terdiri dari **HTTP header**, beserta kode HTML dari halaman yang diminta (panah no 2). Dari gambar diatas, saya menampilkan sedikit isi dari **HTTP header** ini.

Siklus diatas akan terus dijalankan setiap kali website duniaIlkom ditampilkan. Sekarang, bagaimana jika terdapat **cookie**?

Agar cookie sampai di web browser, tentunya data ini dikirim oleh web server. Di dalam PHP, cookie tidak ubahnya sebuah variabel yang berisi pasangan **nama** dan **nilai**. Sebagai contoh, untuk membuat cookie “**user\_id**” dengan nilai “**admin**”, saya bisa menggunakan fungsi **setcookie()** sebagai berikut:

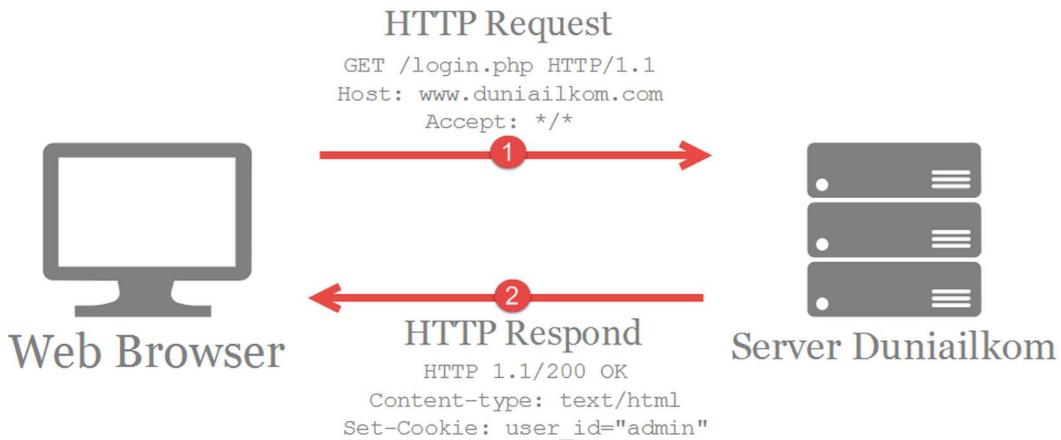
```
setcookie("user_id", "admin");
```

Di dalam PHP, ini tidak lain sama seperti kode berikut:

```
$user_id = "admin";
```

<sup>1</sup><http://www.duniaIlkom.com>.

Saya akan membahas fungsi `setcookie()` sesaat lagi. Fungsi `setcookie()` ini selanjutnya saya tempatkan di halaman `login.php`. Dengan demikian, alur *request – respond* sebelumnya akan menjadi seperti berikut:



Gambar: Data cookie dikirim dari web server ke web browser

Tampak tidak ada yang berubah. Tetapi perhatikan penambahan baris terakhir di **HTTP header respond** (garis nomor 2). Kali ini terdapat baris: `Set-Cookie: user_id="admin"`. Ini artinya, web server meminta kepada web browser agar menyimpan sebuah cookie dengan data: `user_id="admin"`.

Ketika menerima instruksi ini, web browser akan membuat data cookie `user_id="admin"` dan menyimpannya. Selain itu, web browser juga menambahkan informasi alamat situs yang memerintahkan hal ini, yakni situs `duniaIlkom.com`.

Ketika web browser meminta halaman lain ke situs `duniaIlkom.com` (melakukan *request*), seluruh data cookie akan ditambahkan dan dibawa di dalam **HTTP header**, seperti gambar berikut:



Gambar: Data cookie akan dibawa di dalam HTTP header

Perhatikan penambahan baris baru di **HTTP header request** (garis nomor 1), yakni `Cookie: user_id="admin"`. Inilah data cookie yang tadinya di set dari halaman `login.php`. Beginilah cara data cookie dikirim dari web browser.

Pada setiap *request* ke situs duniaIlkom, data cookie akan selalu dibawa di dalam **HTTP header**, terlepas dari halaman apa yang akan diakses, apakah itu home.php, login.php, artikel.php, dll. Selama situs itu adalah duniaIlkom.com, data cookie **user\_id="admin"** akan terus ‘dibawa’ di dalam HTTP header.

Jika anda masih bingung dengan penjelasan diatas, jangan khawatir. Prinsip kerja cookie ini akan semakin dipahami ketika kita masuk ke prakteknya. Inilah yang akan segera kita lakukan.

## 22.3 Cara Membuat Cookie

Untuk membuat cookie, PHP menyediakan fungsi **setcookie()**. Fungsi ini bisa diisi dengan beberapa *argumen*. Untuk membuat cookie, setidaknya kita butuh 2 argumen, yakni **nama** dari cookie, dan **nilainya**.

Sebagai contoh, untuk membuat cookie “**nama**” dengan nilai “**Sheila**”, bisa ditulis dengan perintah berikut:

```
1 <?php
2   setcookie("nama", "Sheila");
3 ?>
```

Mari langsung praktek. Silahkan buat sebuah halaman PHP (misalkan **index.php**), lalu ketikkan kode berikut:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar PHP</title>
6 </head>
7 <body>
8 <h1>Cookie</h1>
9 <?php
10  setcookie("nama", "Sheila");
11 ?>
12 </body>
13 </html>
```



Gambar: Error pada saat membuat cookie

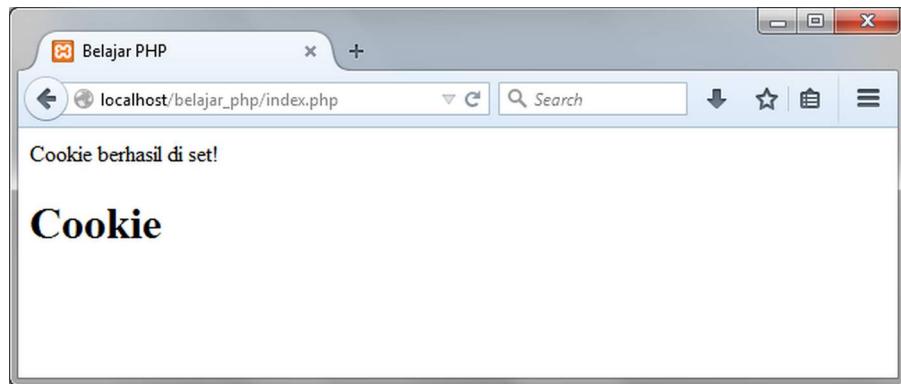
### Apa yang terjadi? Apakah anda masih ingat dengan error ini?

Pesan error “*Warning: Cannot modify header information – headers already sent by*” adalah pesan error yang juga kita dapat saat mempelajari fungsi **header()**. Ini terjadi karena fungsi **setcookie()** juga akan memodifikasi HTTP header.

Anda masih ingat syarat menggunakan fungsi yang memanipulasi HTTP header? Yup, fungsi **setcookie()** harus diletakkan di posisi paling awal, sebelum kode HTML (kecuali anda sudah men-set fitur **output buffering**).

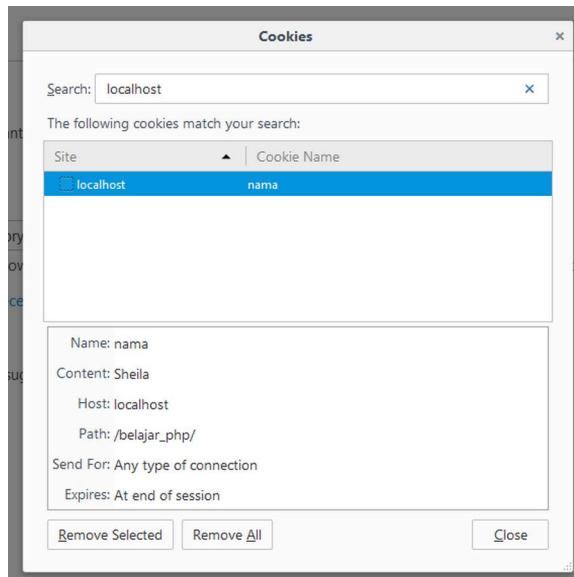
Baik, mari kita pindahkan fungsi tersebut ke baris pertama:

```
1 <?php
2     setcookie("nama", "Sheila");
3     echo "Cookie berhasil di set!";
4 ?>
5 <!DOCTYPE html>
6 <html>
7 <head>
8     <meta charset="UTF-8">
9     <title>Belajar PHP</title>
10 </head>
11 <body>
12     <h1>Cookie</h1>
13 </body>
14 </html>
```



Gambar: Cookie “nama” berhasil di-set

Jika tidak ada error, seharusnya cookie **nama** sudah berhasil di buat. Bagaimana cara memastikannya? Mari kita cek di dalam web browser dari menu **Privacy** seperti yang kita lakukan saat melihat cookie facebook. Bedanya, kali ini ketik “*localhost*” di kolom pencarian cookie.



Gambar: Cookie “nama” di dalam web browser Mozilla Firefox

Jika sebelumnya anda pernah menjalankan script lain dari localhost (seperti *wordpress*, *joomla*, dll). Kemungkinan besar akan terdapat beberapa cookie di localhost. Namun jika ini adalah pertama kali anda menjalankan kode PHP yang berisi *cookie*, hanya akan terdapat 1 hasil seperti gambar diatas.

Di sini anda dapat melihat isi dari cookie yang saya buat sebelumnya. Pada baris **Name** berisi “**nama**” dan di baris **Content** berisi “**Sheila**”. Keduanya sesuai dengan fungsi **setcookie()** yang saya jalankan.

Selain kedua informasi ini, terdapat baris **Host**, **Path**, **Send For**, dan **Expires**:

**Host** adalah nama website yang membuat cookie ini, yakni “*localhost*”. Jika cookie di set dari situs [www.duniaikom.com](http://www.duniaikom.com), maka baris host ini juga akan berisi [www.duniaikom.com](http://www.duniaikom.com).

**Path** adalah nama bagian website dimana cookie ini bisa diakses. Pada gambar diatas, baris ini berisi “*/belajar\_php/*”. Ini karena saya menjalankan fungsi **setcookie()** dari file yang berada di

```
htdocs\belajar_php\index.php.
```

Nilai Path = “/belajar\_php/” artinya, cookie ini hanya bisa diakses oleh kode PHP yang berada di dalam folder **belajar\_php**. Jika saya mencoba mengakses cookie ini dari file `htdocs\index.php`, cookie **nama** tidak lagi ditemukan. Saya akan membahas hal ini dengan lebih detail sesaat lagi.

**Send For** berisi informasi apakah cookie ini dikirim pada protocol **http**, **https** atau keduanya. Nilai “*Any type of connection*” berarti cookie **nama** akan dikirim untuk kedua protocol ini (baik **http** maupun **https**).

Pengertian sederhananya, **http** adalah salah satu *protocol* atau cara komunikasi antara web browser dan web server. Data yang dikirim menggunakan protocol **http** akan dikirim ‘apa adanya’. Sebagai contoh, situs duniaIlkom menggunakan protocol *http* dengan alamat URL lengkap: `http://www.duniaIlkom.com`.

Sedangkan **https** adalah *protocol* atau cara komunikasi antara web yang dikirim di-enkripsi. Di enkripsi ini artinya seluruh komunikasi dari dan ke webserver akan diproteksi agar tidak bisa ‘dicuri’ oleh pihak lain. Biasanya protocol **https** dipakai ketika ingin melalukan transaksi finansial yang melibatkan *password*. Contohnya, situs login klik BCA diakses dari alamat `https://ibank.klikbca.com`.

Cookie bisa di set apakah akan dikirim pada protocol **http** saja, **https** saja, atau keduanya.

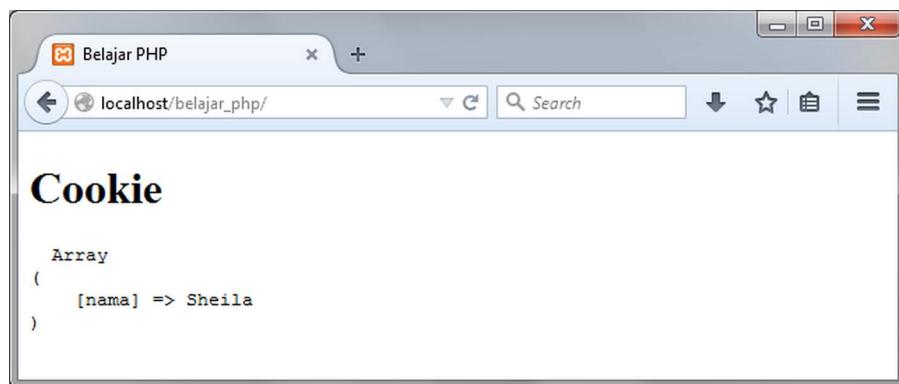
**Expires** adalah waktu yang menunjukkan berapa lama cookie aktif sebelum terhapus secara otomatis oleh web browser. Nilai **At end of session**, artinya cookie **nama** akan dihapus ketika web browser ditutup. Kita bisa mengatur waktu expired ini dari fungsi **setcookie()**.

Saya akan membahas cara mengubah semua pengaturan cookie ini sesaat lagi.

## 22.4 Superglobals Variable `$_COOKIE`

Sama seperti superglobal variabel lainnya, PHP menyediakan variabel `$_COOKIE` yang berisi seluruh data tentang cookie. Langsung saja kita cek dengan perintah `print_r($_COOKIE)`:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6  </head>
7  <body>
8      <h1>Cookie</h1>
9      <pre>
10     <?php
11         print_r($_COOKIE);
12     ?>
13     </pre>
14 </body>
15 </html>
```



Gambar: Menampilkan cookie dari variabel \$\_COOKIE

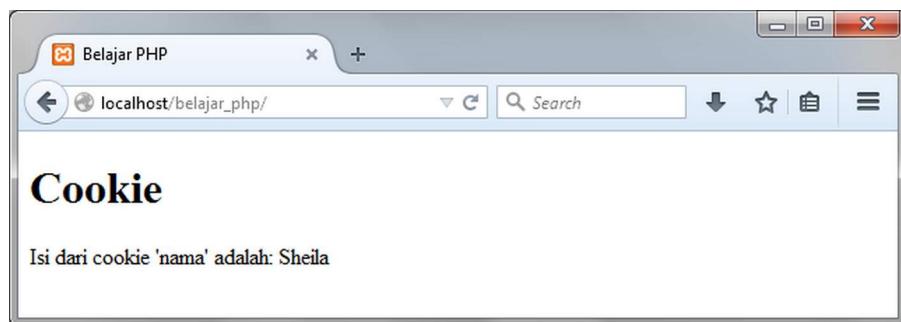
Jika anda menjalankan kode diatas dan menemukan beberapa baris cookie (selain cookie **nama**) bisa jadi sebelumnya sudah pernah menjalankan kode PHP lain di localhost yang juga menggunakan cookie, seperti *WordPress*, *Joomla*, dll. Semua cookie yang ada di localhost akan ditampilkan oleh kode program diatas.

Anda mungkin akan bertanya, bukankah di dalam web browser terdapat berbagai cookie lain? Salah satunya cookie dari facebook yang saya tampilkan di awal bab ini. Kenapa cookie tersebut tidak terlihat?

Sebuah cookie hanya bisa diakses oleh situs yang membuatnya. Jika berasal dari facebook.com, maka hanya situs facebook.com sajalah yang bisa mengakses cookie tersebut.

Untuk menampilkan cookie secara individu, kita tinggal mengakses element array dari **\$\_COOKIE**, seperti contoh berikut:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar PHP</title>
6  </head>
7  <body>
8      <h1>Cookie</h1>
9      <?php
10         echo "Isi dari cookie 'nama' adalah: ".$_COOKIE["nama"];
11     ?>
12  </body>
13  </html>
```



Gambar: Menampilkan cookie "nama" secara langsung

Karena sebuah cookie bisa saja tidak ada (sudah *expired* atau dihapus), biasanya kita ingin mengecek terlebih dahulu apakah cookie itu ada atau tidak sebelum menampilkannya. Berikut contoh kode program yang saya maksud:

```

1 <?php
2   if (isset($_COOKIE["nama"])) {
3     echo "Isi dari cookie 'nama' adalah: ".$_COOKIE["nama"];
4   }
5   else {
6     echo "Cookie 'nama' belum ada";
7   }
8 ?>

```

Dengan kode program ini, kita bisa memastikan hanya menampilkan cookie jika cookie tersebut memang "ada".

## 22.5 Jeda Antara Set Cookie dan Menampilkan Cookie

Karena kita sudah mempelajari cara membuat dan menampilkan cookie, saya ingin menjelaskan kembali tentang prinsip kerja cookie yang dikirim di dalam **HTTP header**.

Silahkan jalankan kode program berikut hanya 1 kali saja (jangan klik refresh):

```

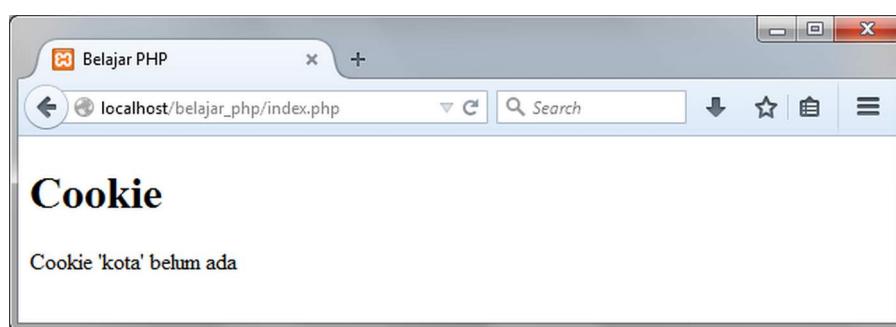
1 <?php
2   setcookie("kota", "Jakarta");
3 ?>
4 <!DOCTYPE html>
5 <html>
6 <head>
7   <meta charset="UTF-8">
8   <title>Belajar PHP</title>
9 </head>
10 <body>

```

```

11  <h1>Cookie</h1>
12  <?php
13  if (isset($_COOKIE["kota"])) {
14      echo "Isi dari cookie 'kota' adalah: ".$_COOKIE["kota"];
15  }
16  else {
17      echo "Cookie 'kota' belum ada";
18  }
19  ?>
20  </body>
21  </html>

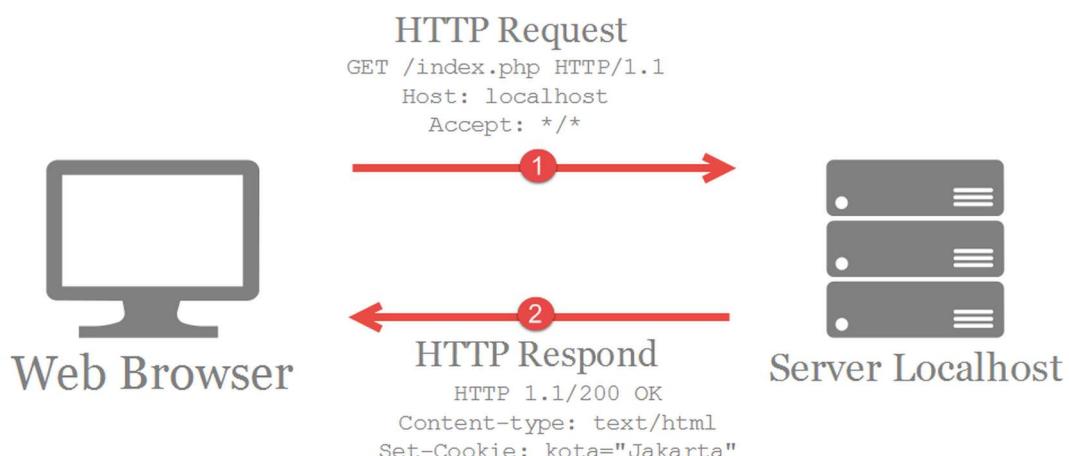
```



Gambar: Cookie “kota” tidak terdeteksi

Apa yang terjadi? Kenapa cookie “kota” dianggap tidak ada? Padahal jelas-jelas saya men-set cookie **kota** dengan nilai “**Jakarta**”.

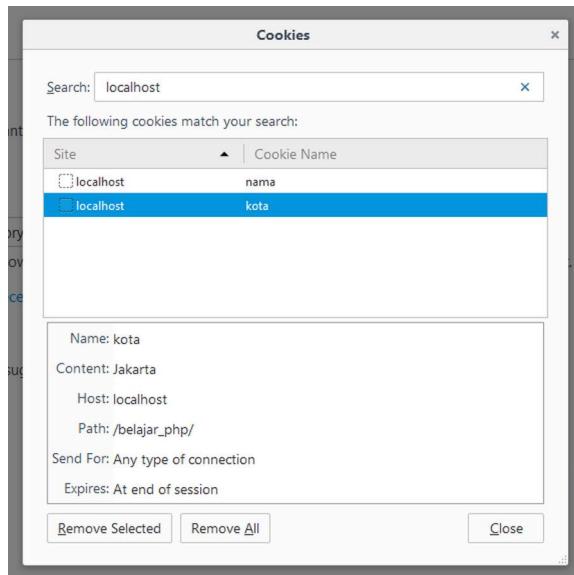
Saya pastikan kode program diatas tidak ada yang salah. Inilah penjelasan dari prinsip kerja cookie. Ketika web server apache menjalankan kode program diatas, ia akan melihat terdapat instruksi untuk membuat cookie **kota**. Web server kemudian memasukkan instruksi pembuatan cookie ini ke dalam **HTTP header**, lalu mengirimnya kepada web browser seperti gambar berikut:



Gambar: Instruksi untuk membuat cookie “kota” dikirim dari web server ke web browser

Yang perlu di ketahui, saat ini cookie "kota" belum ada di dalam web server. Web server baru saja memberikan instruksi pembuatan cookie kota ke web browser. Sebuah cookie hanya bisa diakses ketika web browser mengirimkannya pada saat *request*. Inilah yang menyebabkan tampilnya teks "Cookie 'kota' belum ada". Karena `$_COOKIE['kota']` memang masih kosong.

Sesampainya di web browser, hasil dari kode diatas akan ditampilkan dan web browser membuat cookie kota (sesuai dengan instruksi yang terdapat di **HTTP header** tadi). Anda bisa memeriksanya sendiri:



Gambar: Cookie kota sudah ada di web browser

Sekarang, silahkan **reload** atau **refresh** halaman tersebut. Kali ini akan tampil "*Isi dari cookie 'kota' adalah: Jakarta*". Ini terjadi karena data tentang cookie kota telah didapat dari **HTTP header** kiriman web browser.

Untuk memastikan pemahaman, silahkan anda ubah fungsi `setcookie("kota", "Jakarta")` menjadi `setcookie("kota", "Bandung")`. Ketika kita memberikan nilai yang berbeda pada cookie yang sama, nilai baru akan menimpa nilai lama. Dengan kata lain, disini saya ingin merubah nilai cookie kota dari "Jakarta" menjadi "Bandung".

Berikut kode programnya:

```
1 <?php
2     setcookie("kota", "Bandung");
3 ?>
4 <!DOCTYPE html>
5 <html>
6 <head>
7     <meta charset="UTF-8">
8     <title>Belajar PHP</title>
9 </head>
10 <body>
11     <h1>Cookie</h1>
```

```

12 <?php
13 if (isset($_COOKIE["kota"])) {
14     echo "Isi dari cookie 'kota' adalah: ".$_COOKIE["kota"];
15 }
16 else {
17     echo "Cookie 'kota' belum ada";
18 }
19 ?>
20 </body>
21 </html>

```

Sebelum menjalankan kode diatas, bisakah anda menebak hasil yang ditampilkan? Hasilnya adalah: *“Isi dari cookie ‘kota’ adalah: Jakarta”*.

Kenapa bukan Bandung? Sekali lagi, fungsi `setcookie("kota", "Bandung")` akan dikirim terlebih dahulu ke dalam web browser agar nilai cookie `kota` diubah di web browser (perubahan ini dilakukan bukan di web server, tetapi web browser!). Sedangkan variabel `$_COOKIE["kota"]` saat ini masih berisi **“Jakarta”** yang berasal dari HTTP header web browser sebelumnya (dikirim pada saat menampilkan halaman ini).

Silahkan anda reload lagi dan sekarang akan tampil *“Isi dari cookie ‘kota’ adalah: Bandung”*. Prinsip kerja cookie seperti ini sangat penting untuk dipahami.

## 22.6 Function `setcookie()`

Format penulisan:

```

bool setcookie ( string $name [, string $value = "" [, int $expire = 0
                [, string $path = "" [, string $domain = ""
                [, bool $secure = false [, bool $httponly = false ]]]]] ] )

```

Fungsi `setcookie()` telah saya bahas sekilas sebelum ini. Namun seperti yang terlihat, fungsi `setcookie()` bisa ditulis dengan 7 argumen, 6 diantaranya bersifat opsional (boleh tidak ditulis).

### Argumen Pertama: Nama Cookie

Satu-satunya argumen yang harus ditulis dari fungsi `setcookie()` adalah **nama** dari cookie tersebut. Berikut contohnya:

```

1 <?php
2     setcookie("nama");
3 ?>

```

Jika sebuah cookie ditulis tanpa nilai seperti ini (hanya nama cookienya saja), artinya kita ingin menimpa cookie tersebut dengan **nilai kosong**. Atau dengan kata lain, akan **menghapus** cookie tersebut. Silahkan anda jalankan kode program diatas, kemudian cek ulang cookie `name`. Cookie ini tidak bisa ditemukan lagi.

## Argumen Kedua: Nilai Cookie

Argumen kedua dari fungsi `setcookie()` digunakan untuk menampung nilai cookie. Kita sudah mencoba perintah ini sebelumnya:

```
1 <?php
2     setcookie("nama", "Sheila");
3 ?>
```

Ini artinya saya membuat sebuah cookie **nama** dengan nilai "Sheila".

Contoh lain, untuk membuat cookie **total** dengan nilai **150000**, saya bisa menggunakan kode berikut:

```
1 <?php
2     setcookie("total", 150000);
3 ?>
```

Jika argumen kedua diisi dengan **NULL**, efeknya akan menghapus cookie yang ada:

```
1 <?php
2     setcookie("total", NULL);
3 ?>
```

Sekarang cookie **total** sudah tidak ada lagi.

## Argumen Ketiga: Masa Aktif Cookie

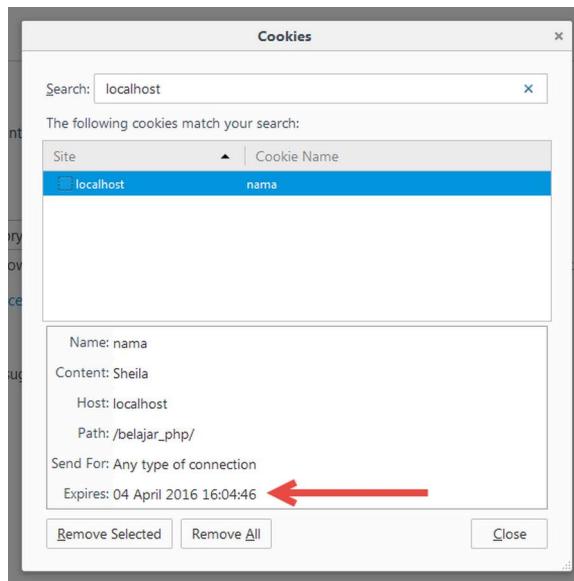
Argumen ketiga dari fungsi `setcookie()` digunakan untuk mengatur durasi masa aktif cookie, yakni berapa lama cookie akan disimpan di dalam web browser. Apakah 5 menit, 1 jam, 1 hari, 1 bulan, atau bahkan 10 tahun.

Jika argumen ketiga ini tidak ditulis, masa aktif cookie adalah sampai web browser di tutup.

Durasi cookie harus diisi dengan nilai **Unix timestamp** (masih ingat pengertiannya?). Sebagai contoh, untuk membuat cookie **nama** dengan nilai "Sheila" aktif selama 1 jam, bisa ditulis sebagai berikut:

```
1 <?php
2     setcookie("nama", "Sheila", time() + 3600);
3 ?>
```

Fungsi `time()` digunakan untuk mencari waktu saat ini. Dengan menulis `time() + 3600` ini artinya waktu *expired* cookie **nama** adalah 3600 detik berikutnya, atau sama dengan 1 jam. Berikut hasil cookie **nama** yang diperiksa melalui web browser:



Gambar: Masa aktif cookie nama selama 1 jam

Perhatikan di baris **Expires: 04 April 2016 16:04:46**. Pada saat saya menjalankan fungsi diatas, saat ini adalah pukul 15:04:46. Artinya, cookie akan aktif hingga 1 jam kedepan.

Bagaimana dengan 1 minggu? Tinggal menghitung jumlah detiknya, kemudian tambahkan ke dalam fungsi **time()**, seperti berikut:

```

1 <?php
2   setcookie("nama", "Sheila", time() + 60*60*24*7);
3 ?>

```

Bagaimana dengan 1 tahun? Saya yakin anda sudah bisa membuatnya sendiri. Tapi, bagimana dengan yang ini?

```

1 <?php
2   setcookie("nama", "Sheila", time() - 60);
3 ?>

```

Disini saya membuat durasi cookie 1 menit sebelumnya, bukan 1 menit ke depan. Ini diterjemahkan oleh web browser dengan cara menghapus cookie **nama**. Dengan kata lain, jika kita memberikan nilai untuk waktu yang telah berlalu, **cookie akan dihapus**.

Digabungkan dengan cara menghapus cookie sebelumnya, saya bisa menulis perintah berikut untuk memastikan cookie benar-benar sudah dihapus:

```

1 <?php
2   setcookie("nama", null, time() - 60);
3 ?>

```

Umumnya, kita hanya perlu mengatur fungsi **setcookie()** hingga argumen ketiga ini. Untuk 4 argumen selanjutnya tidak sering dipakai, tapi tetap akan saya jelaskan.

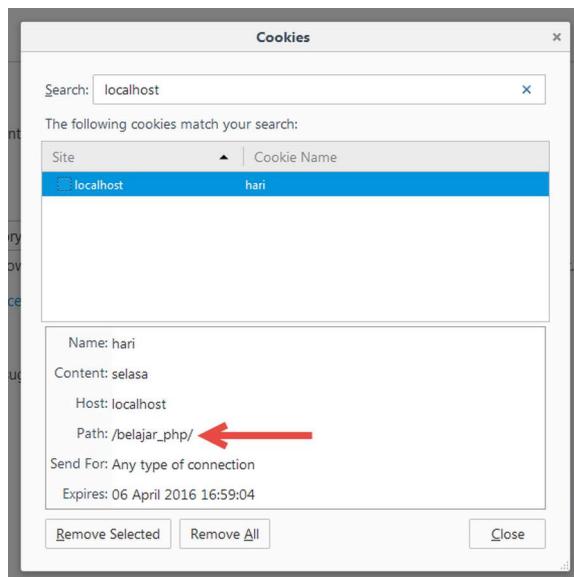
## Argumen Keempat: Mengatur Path

**Path** adalah istilah yang mengacu pada alamat sebuah file, seperti: `htdocs\belajar_php\index.php`. Secara default, cookie yang kita set hanya berlaku di dalam **path** tempat file itu berada.

Sebagai contoh, saya akan membuat sebuah cookie dari file yang berada di `htdocs\belajar_php\buat_cookie.php`, perintahnya adalah sebagai berikut:

```
1 <?php
2     setcookie("hari", "selasa", time() + 3600);
3 ?>
```

Setelah dijalankan, mari kita cek isi dari cookie "hari":



Gambar: Path cookie "hari" adalah /belajar\_php/

Perhatikan di baris **path**, nilainya adalah: `/belajar_php/`. Ini artinya, cookie **hari** hanya bisa diakses oleh seluruh kode PHP yang ada di dalam folder **belajar\_php** dan dibawahnya, seperti `/belajar_php/satu/`.

Jika saya membuat file lain, misalnya `cek_cookie.php` dengan path lengkap: `htdocs\belajar_php\cek_cookie.php`, maka halaman tersebut masih bisa mendeteksi cookie ini. Begitu juga jika file tersebut saya letakkan di `htdocs\belajar_php\folder_baru\cek_cookie.php`, cookie **hari** ini masih bisa di akses karena berada di dalam folder `htdocs\belajar_php\`.

Tapi bagaimana jika saya akses dari `htdocs\cek_cookie.php`? Kali ini file `cek_cookie.php` tidak lagi berada di dalam folder `belajar_php`.

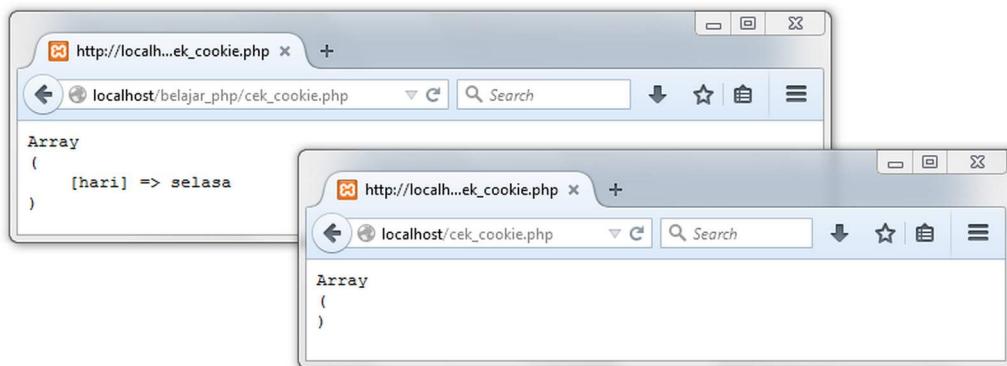
Mari kita cek. Berikut adalah fungsi `print_r()` yang saya buat di dalam file `cek_cookie.php`:

```

1 <?php
2 echo "<pre>";
3 print_r($_COOKIE);
4 echo "<pre>";
5 ?>

```

Saya mengujinya dengan menempatkan file `cek_cookie.php` di lokasi yang berbeda. Pertama di alamat `http://localhost/belajar_php/cek_cookie.php`, dan yang kedua di `http://localhost/cek_cookie.php`.



Gambar: Cookie hari tidak terdeteksi dari file `http://localhost/cek_cookie.php`

Terlihat bahwa cookie `hari` hanya terdeteksi dari file `belajar_php/cek_cookie.php`. Inilah prilaku *default* dari `path` di dalam fungsi `setcookie()` jika tidak dituliskan.

Untuk mengatur `path`, kita bisa menambahkan argumen ke-4 untuk fungsi `setcookie()`. Sebagai contoh, agar cookie `hari` juga bisa terdeteksi dari halaman `localhost/cek_cookie.php`, saya bisa menulisnya sebagai berikut:

```

1 <?php
2 setcookie("hari", "selasa", time() + 3600, "/");
3 ?>

```

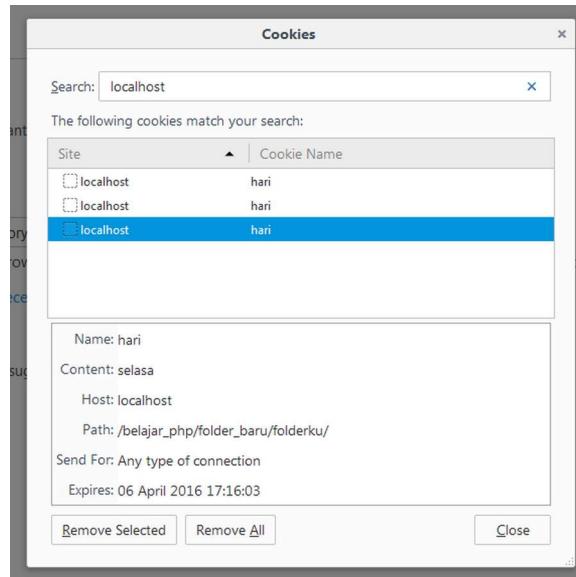


Gambar: Cookie hari sudah terlihat di halaman `localhost/cek_cookie.php`

Tanda `/` berarti saya ingin membuat cookie `hari` untuk seluruh domain, yakni `localhost`. Jika saya ingin membuat cookie yang hanya bisa diakses oleh file yang berada di dalam folder `belajar_php/folder_baru/folderku/cek_cookie.php`, saya bisa menulisnya sebagai berikut:

```
1 <?php
2     setcookie("hari","selasa",time() + 3600,
3                 "/belajar_php/folder_baru/folderku/");
4 ?>
```

Yang menarik, walaupun saya telah membuat 3 buah cookie dengan nama yang sama, yakni **hari**, tapi ketiganya berbeda:



Gambar: 3 buah cookie “hari”

Setiap cookie ini “hidup” di dalam path-nya masing-masing. Silahkan anda praktikkan sendiri, dan ubah salah satu nilai cookie **hari**, misalnya menjadi “selasa”, dan cek cookie mana yang akan berubah. Ini juga berlaku ketika ingin menghapus cookie, kita harus menulis alamat pathnya dengan lengkap.

## Argumen Kelima: Mengatur Domain

Argumen kelima dari fungsi `setcookie()` tidak bisa kita coba karena butuh pengaturan tentang **domain** dan **subdomain**. **Domain** adalah nama sebuah website. Sebagai contoh: *duniaikom.com*, *google.com*, dan *php.net* adalah nama-nama domain.

**Sub domain** adalah domain seperti *blog.duniaikom.com* atau *forum.duniaikom.com*. Perhatikan penambahan kata sebelum domain utama *duniaikom.com*.

Contoh situs aktif yang menggunakan sub domain adalah situs **kompas.com**. Untuk berita teknologi, berada di *tekno.kompas.com*, sedangkan untuk berita bisnis berada di *bisniskeuangan.kompas.com*.

Secara default, cookie bisa diakses dari seluruh sub domain, atau sama dengan:

```
1 <?php
2   setcookie("hari","rabu",time() + 3600, "/", "duniailkom.com");
3 ?>
```

Tapi jika saya menulis sebagai berikut:

```
1 <?php
2   setcookie("hari","rabu",time() + 3600, "/", "blog.duniailkom.com");
3 ?>
```

Artinya, cookie **hari** hanya bisa diakses oleh sub domain *blog.duniailkom.com* dan subdomain dibawahnya seperti *tekno.blog.duniailkom.com* atau *berita.tekno.blog.duniailkom.com*. Namun cookie ini tidak bisa diakses dari domain utama maupun sub domain sekelas, seperti *forum.duniailkom.com*, karena tidak berada dibawah subdomain *blog.duniailkom.com*.

## Argumen Keenam: Mengatur Protocol http atau https

Secara default, cookie akan dikirim dalam 2 jenis protocol: **http** dan **https**. Jika kita menambahkan argumen keenam dari fungsi **setcookie()** dengan nilai **TRUE**, cookie hanya dikirim jika diakses dari protocol **https**.

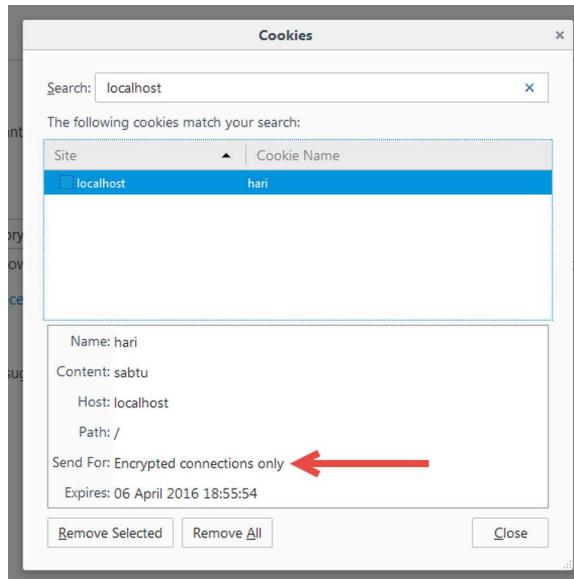
Berikut contoh penulisannya:

```
1 <?php
2   setcookie("hari","rabu",time() + 3600, "/", "duniailkom.com", TRUE);
3 ?>
```

Artinya, cookie **hari** hanya bisa diakses dari alamat: <https://duniailkom.com>. Jika diakses dari <http://duniailkom.com>, cookie nama tidak akan ditemukan.

Mari kita coba di localhost:

```
1 <?php
2   setcookie("hari","senin",time() + 3600, "/", "localhost", TRUE);
3 ?>
```



Gambar: Cookie nama hanya dikirim pada Encrypted connections only

Terlihat bahwa baris **Send for** untuk cookie **hari** berisi nilai “**Encrypted connections only**”. Artinya, cookie hanya akan dikirim jika diakses melalui <https://localhost>.

Jika argumen keenam ini tidak ditulis, nilai defaultnya adalah **FALSE**, dimananya cookie dapat diakses baik pada protocol **http** maupun **https**.

## Argumen Ketujuh: Hanya bisa diakses dari web server

Cookie tidak hanya bisa diset dan diakses dari bahasa pemrograman server seperti PHP. Tapi juga bisa dari bahasa pemrograman *client side* seperti JavaScript.

Ini membuka peluang seseorang membuat kode JavaScript jahat, lalu menggiring pengunjung menjalankan kode tersebut untuk mengakses informasi yang ada di cookie. Ini adalah salah satu teknik “hacking” dengan XSS – (*Cross-site Scripting*).

Dengan memberikan nilai **TRUE** pada argumen ketujuh fungsi **setcookie()**, cookie hanya bisa diakses dari server dan tidak bisa dari JavaScript. Berikut contoh penulisannya:

```

1 <?php
2   setcookie("hari","kamis",time() + 3600, "/", "localhost", FALSE, TRUE);
3 ?>

```

Tapi tidak semua web browser mendukung fitur ini, hanya web browser yang relatif baru saja.

## 22.7 Jeda Ketika Menghapus Cookie

Pembahasan ini sebenarnya kebalikan dari materi tentang jeda antara set cookie dan menampilkan cookie. Disana kita sudah melihat bahwa fungsi **setcookie()** butuh ‘perjalanan’ ke web browser, baru bisa dibuat. Hal ini juga berlaku untuk proses menghapus cookie. Mari kita coba.

Kali ini saya akan membuat cookie **bulan**, dengan nilai “**februari**” dengan menggunakan perintah berikut:

```

1 <?php
2   setcookie("bulan", "februari", time() + 3600);
3 ?>

```

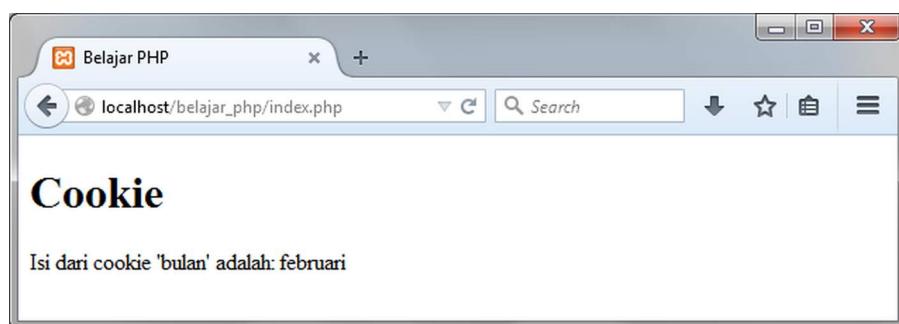
Pastikan cookie ini telah ada (bisa cek dari web browser), lalu jalankan kode program berikut:

```

1 <?php
2   setcookie("bulan", null, time()-60);
3 ?>
4 <!DOCTYPE html>
5 <html>
6 <head>
7   <meta charset="UTF-8">
8   <title>Belajar PHP</title>
9 </head>
10 <body>
11   <h1>Cookie</h1>
12   <?php
13     if (isset($_COOKIE["bulan"])) {
14       echo "Isi dari cookie 'bulan' adalah: ".$_COOKIE["bulan"];
15     }
16     else {
17       echo "Cookie 'bulan' sudah dihapus";
18     }
19   ?>
20 </body>
21 </html>

```

Di baris kedua, saya menjalankan fungsi `setcookie("bulan", null, time()-60)`. Ini adalah perintah untuk menghapus cookie **bulan**. Setelah menjalankan fungsi ini, saya memeriksa apakah cookie bulan masih ada atau sudah dihapus menggunakan fungsi `isset($_COOKIE["bulan"])`. Hasilnya?



Gambar: Cookie bulan masih ada!

Saya yakin anda sudah paham kenapa kode program tersebut tetap menampilkan *“Isi dari cookie ‘bulan’ adalah: februari”*. Ini karena untuk menghapus cookie, perlu *“jeda”* 1 kali *reload*, disebabkan instruksi penghapusan cookie harus dikirim terlebih dahulu ke web browser.

Sebagai alternatif solusi, saya bisa menambahkan fungsi **unset()**:

```
1 <?php
2     // hapus cookie, perintah akan dikirim ke web browser
3     setcookie("bulan",null,time()-60);
4
5     // hapus isi variabel cookie hanya untuk halaman ini
6     unset($_COOKIE["bulan"]);
7 ?>
8 <!DOCTYPE html>
9 <html>
10 <head>
11     <meta charset="UTF-8">
12     <title>Belajar PHP</title>
13 </head>
14 <body>
15     <h1>Cookie</h1>
16     <?php
17     if (isset($_COOKIE["bulan"])) {
18         echo "Isi dari cookie 'bulan' adalah: ".$_COOKIE["bulan"];
19     }
20     else {
21         echo "Cookie 'bulan' sudah dihapus";
22     }
23 ?>
24 </body>
25 </html>
```

Untuk memastikan kode diatas berjalan, silahkan buat kembali cookie bulan. Sekarang, hasil dari kode diatas langsung: *“Cookie ‘bulan’ sudah dihapus”*.

Fungsi `unset($_COOKIE["bulan"])` saya gunakan untuk menghapus cookie **“bulan”** hanya untuk halaman ini saja. Fungsi **unset()** tidak akan menghapus cookie. Untuk menghapus cookie asli, kita tetap menggunakan fungsi **setcookie()**. Trik seperti ini diperlukan agar halaman tidak perlu di *reload* untuk menunggu cookie asli dihapus dari web browser.

## 22.8 Membuat Multiple Cookie

Untuk membuat lebih dari 1 cookie, bisa dengan cara memanggil fungsi **setcookie()** beberapa kali. Sebagai contoh, untuk membuat cookie **id**, **nama** dan **hak\_akses**, saya bisa menggunakan kode program berikut:

```
1 <?php
2     setcookie("id", "AAC01", time() + 60*60*24);
3     setcookie("nama", "Nurul", time() + 60*60*24);
4     setcookie("hak_akses", "admin", time() + 60*60*24);
5 ?>
```

Untuk mengakses ketiga cookie ini, bisa dengan kode program berikut:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="UTF-8">
5     <title>Belajar PHP</title>
6 </head>
7 <body>
8     <h1>Cookie</h1>
9     <?php
10     echo $_COOKIE["id"]; echo "<br>";
11     echo $_COOKIE["nama"]; echo "<br>";
12     echo $_COOKIE["hak_akses"];
13 ?>
14 </body>
15 </html>
```

Tidak ada hal baru disini selain memanggil beberapa kali fungsi `setcookie()` dan mengakses variabel `$_COOKIE`. Berikutnya, kita akan masuk ke `session`.

## 22.9 Mengenal Session

Session sebenarnya adalah bentuk lain dari cookie. Session juga berfungsi untuk mengingat ‘siapa’ user yang saat ini sedang mengakses situs kita. Tapi ada beberapa perbedaan mendasar antara session dengan cookie.

### Data session disimpan di dalam web server

Telah kita pelajari bahwa data cookie disimpan di dalam **web browser**. Sedangkan untuk session, datanya disimpan di dalam **web server**.

Session tetap menggunakan cookie untuk dapat bekerja, tapi isi dari cookie tersebut hanya berupa referensi atau alamat yang merujuk kepada sebuah file di dalam web server. Seluruh data session, disimpan di dalam file ini (yang berada di dalam web server).

Karena data disimpan di dalam web server, session relatif lebih aman dari pada cookie, karena isi session tidak bisa ‘diintip’ dari web browser seperti halnya cookie.

### **Session akan hilang ketika web browser di tutup**

Berbeda dengan cookie yang bisa diset umurnya hingga 10 tahun atau lebih, data yang disimpan di dalam session hanya bertahan sampai web browser tersebut ditutup, atau hanya dalam 1 sesi.

Namun perlu juga diketahui, beberapa web browser menawarkan fitur untuk menyimpan session pada saat web browser akan ditutup. Jadi, data session bisa saja tidak terhapus apabila user menggunakan fitur ini.

### **Session disimpan di dalam variabel superglobal `$_SESSION`**

Walaupun prinsip kerjanya mirip dengan cookie (dan memang menggunakan cookie), data tentang session disimpan di dalam variabel superglobal `$_SESSION`. Cara pengaksesan variabel ini sama seperti variabel superglobal lain.

### **Session bisa menyimpan lebih banyak data**

Data yang disimpan di cookie memiliki batas maksimal. Karena tentu saja kita tidak mau harddisk penuh sesak oleh cookie yang dititipkan dari ratusan website. Maksimum data yang bisa disimpan ke dalam cookie web browser adalah 4kB, atau sekitar 4000 karakter. Ini berlaku untuk total seluruh cookie dari sebuah website.

Sedangkan pada session, karena datanya disimpan di dalam server, tidak ada batasan. Batas maksimalnya adalah ruang harddisk di web server yang bisa mencapai puluhan GB.

## **22.10 Prinsip Kerja dan Cara Pembuatan Session**

Untuk mulai menggunakan session, kita harus menggunakan fungsi `session_start()`. Fungsi `session_start()` adalah sebuah fungsi “ajaib” yang melakukan banyak kerja secara otomatis.

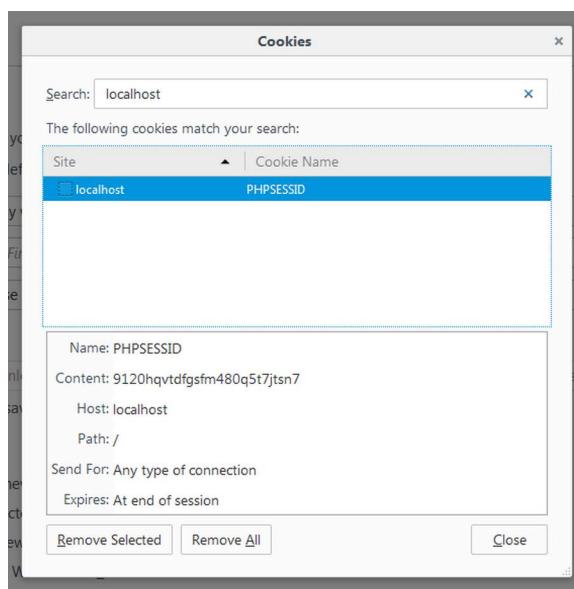
Pertama kali, fungsi ini akan membuat sebuah **cookie** yang berisi referensi kepada file session di web browser. Jika cookie referensi ini sudah ada, maka tinggal mengambil nilai referensinya saja (cookie tidak akan dibuat lagi). Selanjutnya, fungsi `session_start()` akan membuka file referensi di dalam web server dan mengambil isinya untuk dimasukkan kedalam variabel `$_SESSION` (jika ada).

Penjelasan ini akan lebih mudah dipahami ketika kita masuk ke praktiknya. Oleh karena itu, silahkan jalankan kode program berikut:

```
1 <?php
2     session_start();
3 ?>
4 <!DOCTYPE html>
5 <html>
6 <head>
7     <meta charset="UTF-8">
8     <title>Belajar PHP</title>
9 </head>
10 <body>
11     <h1>Session</h1>
12     <p>Cookie untuk session sudah dibuat!</p>
13 </body>
14 </html>
```

Saya menempatkan fungsi `session_start()` pada baris pertama kode PHP. Ini karena session menggunakan cookie dan cookie menggunakan **HTTP header**.

Silahkan jalankan kode program diatas, dan mari kita cek cookie yang dibuat oleh fungsi `session_start()` ini:

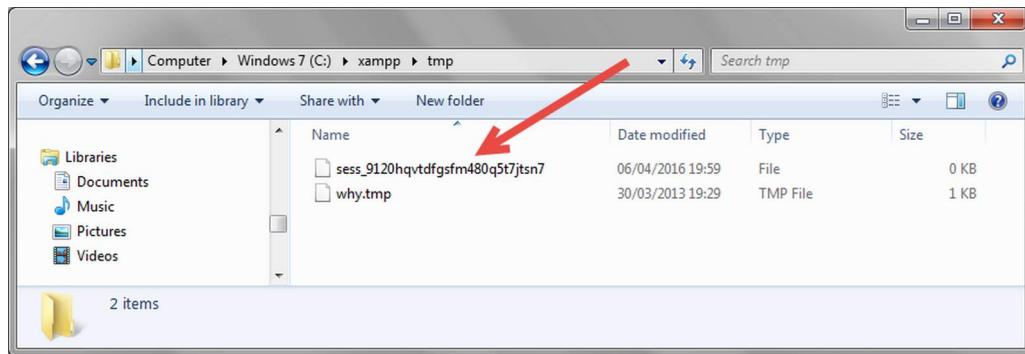


Gambar: Cookie yang digunakan oleh session

Disini kita bisa melihat sebuah cookie dengan nama: **PHPSESSID**. Inilah nama default yang digunakan oleh PHP. Jika anda berminat, nama cookie ini bisa disetting dari `php.ini`, yakni dari pengaturan `session.name`.

Perhatikan baris **Content**, isinya berupa karakter acak: `9120hqvtdfgsfm480q5t7jtsn7`. Inilah yang saya maksud dengan **referensi dari session**. Karakter ini akan berpasangan dengan sebuah file di dalam web server. Dimanakah file ini disimpan?

Secara default, file untuk session disimpan di dalam folder `C:\xampp\tmp`, mari kita periksa:



Gambar: Lokasi file session

Disinilah seluruh data dari session disimpan. Anda bisa membukanya dengan teks editor **Notepad++** untuk melihat apa isi dari file ini. Tapi karena kita belum menginput nilai session apapun, file ini juga masih kosong. Lokasi tempat file session juga bisa disetting dari **php.ini**, yakni di pengaturan **session.save\_path**.

Baik, selanjutnya mari kita input sebuah data ke dalam session:

```

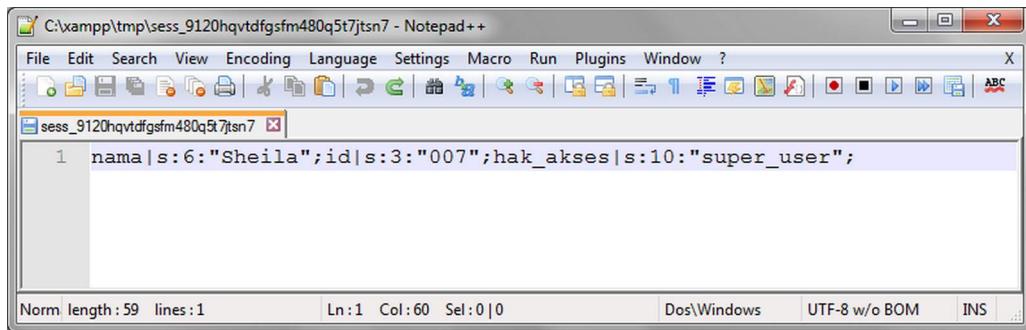
1 <?php
2     session_start();
3 ?>
4 <!DOCTYPE html>
5 <html>
6 <head>
7     <meta charset="UTF-8">
8     <title>Belajar PHP</title>
9 </head>
10 <body>
11     <h1>Session</h1>
12     <?php
13         $_SESSION["nama"] = "Sheila";
14         $_SESSION["id"] = "007";
15         $_SESSION["hak_akses"] = "super_user";
16     ?>
17 </body>
18 </html>

```

Disini saya tetap menggunakan fungsi **session\_start()**. Fungsi ini akan memeriksa apakah saat ini cookie **PHPSESSID** sudah tersedia di web browser. Jika sudah, ambil nilai referensnya. Jika belum ada, buat cookie baru.

Untuk membuat session, caranya cukup sederhana. Kita tinggal menginput data yang diinginkan ke dalam variabel **\$\_SESSION**. Pada contoh diatas, saya membuat 3 buah data session: **nama**, **id** dan **hak\_akses**.

Setelah menjalankan kode diatas, mari kita cek kembali isi dari file session tadi:



Gambar: Isi dari file “sess\_9120hqvtdfgsm480q5t7jtsn7”

Terlihat bagaimana isi dari session disimpan ke dalam file “sess\_9120hqvtdfgsm480q5t7jtsn7”. Kita tidak perlu mempelajari atau mengubah isi file ini, karena PHP otomatis mengambilnya lewat fungsi `session_start()`.

Lalu, bagaimana cara mengambil dan menampilkan data session? Caranya juga sama seperti variabel superglobal lain. Berikut contohnya:

```

1 <?php
2   session_start();
3 ?>
4 <!DOCTYPE html>
5 <html>
6 <head>
7   <meta charset="UTF-8">
8   <title>Belajar PHP</title>
9 </head>
10 <body>
11   <h1>Session</h1>
12   <?php
13     echo "Nama = ".$_SESSION["nama"]."<br>";
14     echo "ID = ".$_SESSION["id"]."<br>";
15     echo "Hak akses = ".$_SESSION["hak_akses"];
16   ?>
17 </body>
18 </html>

```

Untuk mengakses data session `nama`, saya tinggal memanggil variabel `$_SESSION["nama"]`. Begitu juga dengan variabel session lain.

## 22.11 Menghapus Session

Bagaimana cara menghapus session? Terdapat beberapa perintah, tergantung apa yang ingin kita hapus.

Pertama, jika ingin menghapus data cookie secara individual, bisa menggunakan fungsi `unset()`, seperti contoh berikut:

```
1 <?php
2     session_start();
3     //... kode PHP lain disini...
4     unset($_SESSION["nama"]);
5 ?>
```

Sekarang, variabel session nama sudah tidak ada lagi (sudah dihapus dari file session). Yang juga perlu diingat, fungsi **session\_start()** tetap harus dipanggil.

Yang kedua, jika kita ingin **menghapus seluruh variabel session**, bisa menggunakan fungsi **session\_unset()**, seperti berikut:

```
1 <?php
2     session_start();
3     //... kode PHP lain disini...
4     session_unset();
5 ?>
```

Sekarang, seluruh variabel yang ada di dalam **\$\_SESSION** akan terhapus.

Yang ketiga, apabila ingin **menghapus fisik file session**, bisa menggunakan fungsi **session\_destroy()**, seperti berikut:

```
1 <?php
2     session_start();
3     //... kode PHP lain disini...
4     session_destroy();
5 ?>
```

Pemanggilan fungsi **session\_destroy()** akan menghapus file session dari temporary folder. Sekarang, file **sess\_9120hqvtdfgsfm480q5t7jtsn7** sudah tidak ada lagi, termasuk seluruh data session didalamnya juga ikut terhapus.

Walaupun data session itu sudah tidak ada, cookie yang digunakan oleh session masih ditemukan di dalam web browser. Ketika saya membuat kembali data session (sebelum menutup web browser) referensi yang digunakan tetap **9120hqvtdfgsfm480q5t7jtsn7**.

Jadi bagaimana cara menghapus seluruh data session berserta cookienya? Untuk menghapus cookie ini, sedikit rumit karena kita perlu mencari tau seluruh argumen yang digunakan PHP untuk membuat cookie **PHPSESSID**. Berikut cara menghapus cookie session yang saya ambil dari PHP Manual:

```
1 <?php
2 // mulai session
3 session_start();
4
5 // ambil seluruh data terkait cookie yang digunakan untuk session
6 $params = session_get_cookie_params();
7
8 // set ulang cookie (proses hapus cookie)
9 setcookie(session_name(), null, time() - 42000,
10           $params["path"], $params["domain"],
11           $params["secure"], $params["httponly"]
12         );
13 // hapus file session di dalam server
14 session_destroy();
15 ?>
```

Disini saya menggunakan beberapa fungsi yang belum kita bahas. Pertama, fungsi `session_get_cookie_params()`. Fungsi ini digunakan untuk mengambil seluruh data cookie yang digunakan ketika membuat session. Data ini seperti lama expired cookie, nama path, nama domain (host), dll. Hasilnya berupa *associative array*, seperti berikut ini:

```
1 <?php
2 // mulai session
3 session_start();
4
5 // ambil seluruh data terkait cookie yang digunakan untuk session
6 $params = session_get_cookie_params();
7 echo "<pre>";
8 print_r($params);
9 echo "</pre>";
10 ?>
11
12 /* Output-----
13 Array
14 (
15     [lifetime] => 0
16     [path] => /
17     [domain] =>
18     [secure] =>
19     [httponly] =>
20 )
-----*/
```

Pengaturan ini didapat dari settingan session dari `php.ini`. Terlihat bahwa pengaturan untuk `domain`, `secure`, dan `httponly` kosong, karena memang tidak saya ubah dari `php.ini`.

Hasil dari fungsi `session_get_cookie_params()` kemudian menjadi input untuk fungsi `setcookie()`. Argumen pertama untuk fungsi ini adalah hasil pemanggilan fungsi `session_name()`. Fungsi `session_name()` digunakan untuk mengambil nama cookie, dimana dalam contoh kita sebenarnya ini adalah string: `PHPSESSID`. Anda bisa memeriksanya sendiri dengan kode berikut:

```
1 <?php
2 echo session_name(); // PHPSESSID
3 ?>
```

Dengan menulis manual fungsi `setcookie()`, kode program untuk menghapus cookie session sebelumnya bisa ditulis menjadi :

```
1 <?php
2 // mulai session
3 session_start();
4
5 // set ulang cookie (proses hapus cookie)
6 setcookie("PHPSESSID", null, time() - 42000, "/", "", "", "");
7
8 // hapus file session di dalam server
9 session_destroy();
10 ?>
```

Perhatikan penulisan fungsi `setcookie()`, argumen ke 5, 6 dan 7 saya isi dengan string kosong. Ini sesuai dengan hasil yang didapat dari fungsi `session_get_cookie_params()`.

Dalam prakteknya, kita tidak perlu sampai menghapus data cookie session seperti ini. Menghapus session melalui fungsi `unset()` atau `session_unset()` sudah mencukupi.

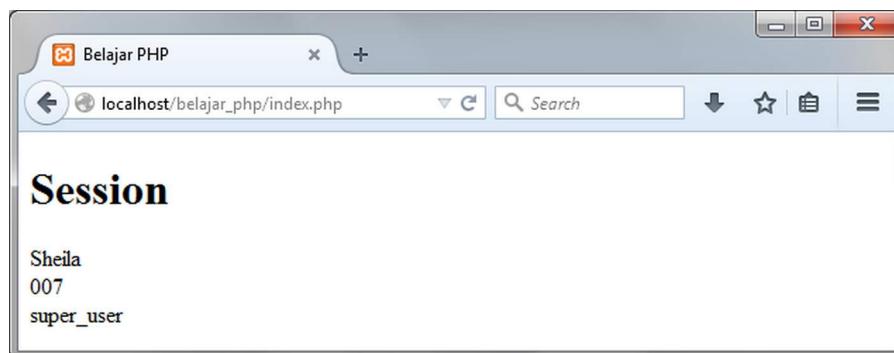
## 22.12 Jeda Antara Set Session dan Menampilkan Session

Berbeda dengan cookie yang harus memiliki “jeda” ketika cookie di buat dan diakses, di dalam session jeda ini tidak ada. Karena file session disimpan di dalam server, kita tidak perlu menunggu agar data ini bisa diakses. Berikut contohnya:

```

1  <?php
2      session_start();
3  ?>
4  <!DOCTYPE html>
5  <html>
6  <head>
7      <meta charset="UTF-8">
8      <title>Belajar PHP</title>
9  </head>
10 <body>
11     <h1>Session</h1>
12     <?php
13         $_SESSION["nama"] = "Sheila";
14         $_SESSION["id"] = "007";
15         $_SESSION["hak_akses"] = "super_user";
16
17         echo $_SESSION["nama"];      echo "<br>";
18         echo $_SESSION["id"];      echo "<br>";
19         echo $_SESSION["hak_akses"];
20     ?>
21 </body>
22 </html>

```



Gambar: Session bisa di set dan diakses dalam 1 halaman

Setelah menginput tiga data ke dalam session, saya langsung mengakses dan menampilkannya menggunakan perintah **echo**. Hasilnya, seluruh data session sukses ditampilkan. Jika kode program diatas diganti dengan cookie, maka saya harus menunggu hingga cookie tercipta di web browser (1 kali proses reload).

## 22.13 Case Study: Form Login dengan Cookie

Menutup bab tentang cookie dan session, saya akan merancang studi kasus sederhana membuat form login yang menggunakan **cookie**, yakni ‘mini situs’: **Sekolah SMA 1 Jambu Air** :)

Disini saya memiliki 3 ‘halaman rahasia’: `daftar_siswa.php`, `nilai_siswa.php`, dan `alamat_sekolah.php`. Ketiga halaman ini hanya bisa diakses oleh user yang menginput username dan

password yang benar.

Jika langsung mengakses salah satu halaman tanpa login, dia akan di *redirect* ke halaman `index.php`. Di halaman `index.php` terdapat sebuah **form login**. Di sinilah proses login dilakukan.

Pada saat user mengisi dan men-submit form login, saya sudah mempersiapkan proses validasi. Hanya jika username dan password benar (dengan data yang ada di sistem), barulah bisa mengakses halaman lain.

Dihalaman login inilah, saya membuat cookie “`username`”. Cookie ini yang akan menjadi patokan bahwa seseorang telah login dan mendapatkan hak akses.

Terkahir, saya menyediakan tombol **logout** untuk user keluar dari sistem. Caranya adalah dengan menghapus cookie “`username`”. Total saya akan membuat 5 halaman: `index.php`, `daftar_siswa.php`, `nilai_siswa.php`, `alamat_sekolah.php`, dan `logout.php`.

## Kode Program Untuk Cek Cookie

Pertama, saya butuh sebuah kode program untuk memeriksa apakah cookie `username` ada atau tidak. Kode program ini akan ditambahkan di bagian paling atas setiap halaman yang harus memiliki hak akses, yakni halaman `daftar_siswa.php`, `nilai_siswa.php`, dan `alamat_sekolah.php`. Jika cookie ditemukan, tampilkan halaman. Jika tidak ada, *redirect* ke halaman `index.php`.

Berikut kode program untuk memeriksa cookie tersebut:

```
1 <?php
2 // periksa apakah user sudah login, cek kehadiran cookie username
3 // jika tidak ada, redirect ke index.php
4 if (!isset($_COOKIE["username"])) {
5     header("Location: index.php");
6 }
7 ?>
```

Cukup dengan tiga baris program! (diluar komentar dan tag pembuka PHP). Saya yakin anda sudah bisa memahami maksud kode diatas. Jika variabel `$_COOKIE["username"]` tidak terdeteksi, jalankan fungsi `header("Location: index.php")`, hasilnya user akan di *redirect*.

## Halaman index.php

Selanjutnya adalah merancang halaman `index.php`. Berikut kode program lengkapnya:

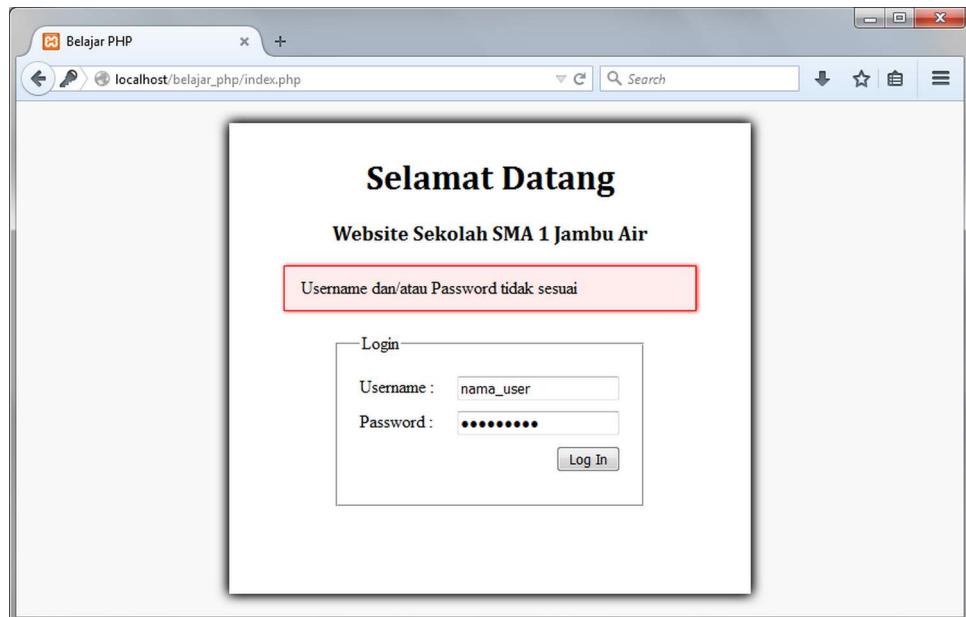
## index.php

```
1 <?php
2 // cek apakah form telah di submit
3 if (isset($_POST["submit"])) {
4     // form telah disubmit, proses data
5
6     // ambil nilai form
7     $username = htmlentities(strip_tags(trim($_POST["username"])));
8     $password = htmlentities(strip_tags(trim($_POST["password"])));
9
10    // siapkan variabel untuk menampung pesan error
11    $pesan_error="";
12
13    // cek apakah "username" sudah diisi atau tidak
14    if (empty($username)) {
15        $pesan_error .= "Username belum diisi <br>";
16    }
17
18    // cek apakah "password" sudah diisi atau tidak
19    if (empty($password)) {
20        $pesan_error .= "Password belum diisi <br>";
21    }
22    // username harus "admin" dan password adalah "rahasia"
23    if ($username!="admin" OR $password!="rahasia") {
24        $pesan_error .= "Username dan/atau Password tidak sesuai";
25    }
26
27    // jika lolos validasi, set cookie
28    if ($pesan_error === "") {
29        setcookie("username","admin");
30        setcookie("nama","Andika");
31        header("Location: data_siswa.php");
32    }
33 }
34 else {
35     // form belum disubmit atau halaman ini tampil untuk pertama kali
36     // berikan nilai awal untuk semua isian form
37     $pesan_error = "";
38     $username = "";
39     $password = "";
40 }
41
42 ?>
43 <!DOCTYPE html>
44 <html>
```

```
45 <head>
46     <meta charset="UTF-8">
47     <title>Belajar PHP</title>
48     <style>
49         body {
50             background-color: #F8F8F8;
51         }
52         div.container {
53             width: 380px;
54             padding: 10px 50px 80px;
55             background-color: white;
56             margin: 20px auto;
57             box-shadow: 1px 0px 10px, -1px 0px 10px ;
58         }
59         h1,h3 {
60             text-align: center;
61             font-family: Cambria, "Times New Roman", serif;
62         }
63         p {
64             margin:0;
65         }
66         fieldset {
67             padding:20px;
68             width: 240px;
69             margin: auto;
70         }
71         input {
72             margin-bottom:10px;
73         }
74         input[type=submit] {
75             float:right;
76         }
77         label {
78             width:80px;
79             float:left;
80             margin-right:10px;
81         }
82         .error {
83             background-color: #FFECEC;
84             padding: 10px 15px;
85             margin: 0 0 20px 0;
86             border: 1px solid red;
87             box-shadow: 1px 0px 3px red ;
88         }
89     </style>
90 </head>
```

```
91 <body>
92 <div class="container">
93 <h1>Selamat Datang</h1>
94 <h3>Website Sekolah SMA 1 Jambu Air</h3>
95 <?php
96 // tampilkan error jika ada
97 if ($pesan_error !== "") {
98     echo "<div class=\"error\">$pesan_error</div>";
99 }
100 ?>
101 <form action="index.php" method="post">
102 <fieldset>
103 <legend>Login</legend>
104     <p>
105         <label for="username">Username : </label>
106         <input type="text" name="username" id="username"
107             value="<?php echo $username ?>">
108     </p>
109     <p>
110         <label for="password">Password : </label>
111         <input type="password" name="password" id="password"
112             value="<?php echo $password?>">
113     </p>
114     <p>
115         <input type="submit" name="submit" value="Log In">
116     </p>
117 </fieldset>
118 </form>
119 </div>
120 </body>
121 </html>
```

---



Gambar: Halaman index.php berisi form login

Silahkan anda pelajari sebentar kode program diatas. Disini saya memiliki 2 form isian: **username** dan **password**. Untuk proses validasi form tidak akan saya bahas lagi karena sudah kita pelajari dalam bab sebelumnya.

Pengecekan isian username dan password dilakukan oleh kode program berikut:

```

1 // username harus "admin" dan password adalah "rahasia"
2 if ($username!="admin" OR $password!="rahasia") {
3     $pesan_error .= "Username dan/atau Password tidak sesuai";
4 }
```

Disini saya membuat manual nama username yang harus disini, yakni **“admin”** dengan password **“rahasia”**. Dalam prakteknya, isi data ini bisa diambil dari database.

Jika username dan password cocok (tidak ada error), perintah berikut akan dijalankan:

```

1 if ($pesan_error === "") {
2     setcookie("username", "admin");
3     setcookie("nama", "Andika");
4     header("Location: data_siswa.php");
5 }
```

Disini saya membuat 2 buah cookie: **username** dengan nilai **“admin”**, dan **nama** dengan nilai **“Andika”**. Cookie **username** saya buat untuk mendeteksi ini adalah user yang berhak, sedangkan cookie **nama** hanya untuk tambahan informasi. Setelah kedua cookie dibuat, user langsung saya *redirect* ke halaman **data\_siswa.php**.

Karena halaman ini langsung saya *redirect*, sebuah alur *request-respond* akan dijalankan. Sehingga di halaman **data\_siswa.php**, kedua cookie ini sudah langsung bisa diakses. Dengan metode ini, saya tidak mendapat masalah terkait jeda antara saat cookie di buat dan saat cookie baru bisa diakses.

## Halaman `data_siswa.php`, `nilai_siswa.php`, dan `alamat_sekolah.php`

Apabila di halaman `index.php` seseorang login dengan data yang benar, yakni username “`admin`” dan password “`rahasia`”, ia akan di *redirect* ke halaman `daftar_siswa.php`.

Di halaman `daftar_siswa.php` (dan juga halaman `nilai_siswa.php` dan `alamat_sekolah.php`) tidak banyak kode PHP yang diperlukan. Selain kode program untuk memeriksa ada tidaknya cookie, sisanya hanyalah kode HTML dan CSS.

Agar tampilan “mini website” **Sekolah SMA 1 Jambu Air** ini terlihat cukup indah, saya menambahkan sebuah menu horizontal dan beberapa kode CSS. Berikut tampilan dari halaman `daftar_siswa.php`:



Gambar: Halaman `data_siswa.php`

Berikut kode lengkap untuk halaman `data_siswa.php`:

### `data_siswa.php`

```

1 <?php
2     // periksa apakah user sudah login, cek kehadiran cookie username
3     // jika tidak ada, redirect ke index.php
4     if (!isset($_COOKIE["username"])) {
5         header("Location: index.php");
6     }
7 ?>
8 <!DOCTYPE html>
9 <html>
10 <head>
11     <meta charset="UTF-8">
12     <title>Belajar PHP</title>

```

```
13 <style>
14     /* =====GLOBAL STYLE===== */
15     body {
16         background-color: #F8F8F8;
17     }
18     div.container {
19         width: 530px;
20         padding: 10px 50px 60px;
21         background-color: white;
22         margin: 20px auto;
23         box-shadow: 1px 0px 10px, -1px 0px 10px ;
24     }
25     h1,h3 {
26         text-align: center;
27         font-family: Cambria, "Times New Roman", serif;
28     }
29
30     /* =====NAVIGATION===== */
31     ul{
32         padding: 0;
33         margin: 20px 0;
34         list-style: none;
35         overflow: hidden;
36     }
37     nav li a {
38         float: left;
39         background-color: #E3E3E3;
40         color: black;
41         text-decoration: none;
42         font-size: 20px;
43         height: 30px;
44         line-height: 30px;
45         padding: 5px 20px;
46     }
47     nav li a:hover {
48         background-color: #757575;
49         color: white;
50     }
51
52     /* =====TABLE===== */
53     table {
54         border-collapse:collapse;
55         border-spacing:0;
56         border:1px black solid;
57         width:100%
58     }
```

```
59     th, td {
60         padding:8px 15px;
61         border:1px black solid;
62     }
63     td:first-child {
64         background-color: #F2F2F2;
65     }
66     </style>
67 </head>
68 <body>
69 <div class="container">
70     <nav>
71         <ul>
72             <li><a href="data_siswa.php">Data Siswa</a></li>
73             <li><a href="nilai_siswa.php">Nilai Siswa</a>
74             <li><a href="alamat_sekolah.php">Alamat Sekolah</a></li>
75             <li><a href="logout.php">Logout</a>
76         </ul>
77     </nav>
78     <h1>Selamat Datang, <?php echo $_COOKIE["nama"] ?></h1>
79     <h3>Data Siswa SMA 1 Jambu Air</h3>
80     <table>
81         <tr>
82             <th>No.</th><th>Nama</th><th>Umur</th><th>Tempat Lahir</th>
83         </tr>
84         <tr>
85             <td>01</td><td>Andika Sitepu</td><td>16</td><td>Medan</td>
86         </tr>
87         <tr>
88             <td>02</td><td>Joko Susboyo</td><td>17</td><td>Bogor</td>
89         </tr>
90         <tr>
91             <td>03</td><td>Stephani Aleza</td><td>17</td><td>Jakarta</td>
92         </tr>
93     </table>
94 </div>
95 </body>
96 </html>
```

Pada baris paling awal, saya menempatkan kode program untuk memeriksa apakah cookie ditemukan atau tidak. Jika tidak, langsung *redirect* ke halaman **index.php**.

Seperti yang terlihat, selain kode program tersebut, sisanya adalah kode HTML dan CSS. Silahkan anda pelajari sejenak sebagai penyegaran skill HTML dan CSS anda :)

Di bagian judul saya menambahkan lagi sedikit kode PHP:

```
<h1>Selamat Datang, <?php echo $_COOKIE["nama"] ?></h1>
```

Ini digunakan untuk membuat salam pembuka yang nilainya diambil dari cookie. Cookie **nama** juga saya set di halaman `index.php`.

Untuk halaman `nilai_siswa.php` dan `alamat_sekolah.php`, isinya sangat mirip. Yang berbeda hanyalah kode HTML untuk membuat konten. Dimana saya menukarnya dengan tabel lain agar tampak berbeda. Berikut tampilan halaman `nilai_siswa.php` dan `alamat_sekolah.php`:



Gambar: Halaman `nilai_siswa.php`



Gambar: Halaman `alamat_sekolah.php`

Agar menghemat tempat, saya tidak menampilkan kode program untuk halaman ini. Silahkan anda buka file `belajar_php.zip` untuk kode lengkapnya.

## Halaman logout.php

Halaman terakhir adalah `logout.php`. Halaman ini bisa diakses dari menu `logout` yang sudah saya sediakan. Disini kode program yang dibutuhkan juga tidak banyak. Saya cukup menghapus cookie, lalu `redirect` kembali ke halaman `index.php`. Berikut kode program untuk halaman `logout.php`:

### logout.php

```
1 <?php
2 // hapus cookie
3 setcookie("username", null, time() - 60);
4 setcookie("nama", null, time() - 60);
5
6 // redirect ke halaman index.php
7 header("Location: index.php");
8 ?>
```



Gambar: Link ke halaman `logout.php`

Saya sangat menyarankan anda untuk mencoba dan menjalankan sendiri case study ini. Disini anda bisa melihat bagaimana penggunaan cookie untuk membuat fitur login di dalam sebuah website.

Untuk sekedar catatan, cara penggunaan cookie yang saya buat disini sangat sederhana dan belum bisa dibilang aman. Setiap orang bisa mengecek manual cookie yang dihasilkan. Selain itu, di dalam cookie juga tersimpan informasi tentang username "admin" dan nama "Andika". Untuk aplikasi yang tidak butuh keamanan tinggi, seperti tugas atau skripsi, teknik login seperti ini sudah mencukupi.

Sebagai latihan, bisakah anda memodifikasi kode case study ini agar menggunakan session? Perubahannya tidak terlalu banyak, tapi cukup untuk memastikan pemahaman anda tentang session. Selain itu, dengan menggunakan session, data kita akan lebih aman (jika dibandingkan dengan cookie).

Tantangan kedua, silahkan anda buka seluruh kode program case study ini (dari folder `belajar_php.zip`). Anda akan melihat banyak kode HTML dan CSS yang sama di halaman

`data_siswa.php`, `nilai_siswa.php`, dan `alamat_sekolah.php`. Bisakah anda memecah bagian ini menjadi halaman terpisah? Bagian header bisa disimpan ke dalam file `header.php`. Lalu input kembali menggunakan fungsi `include()`.

---

Dalam bab ini kita telah mempelajari tentang cara penggunaan **cookie** dan **session** di dalam PHP. Ini juga menutup pembahasan tentang variabel superglobals PHP.

Berikutnya, saya akan masuk ke materi yang paling ditunggu-tunggu: Berkomunikasi dengan database MySQL.

# 23. Koneksi PHP dengan MySQL

Dengan semua materi PHP yang telah kita bahas sejauh ini, sebenarnya sudah bisa digunakan untuk membuat berbagai aplikasi web dinamis. Kita sudah mempelajari cara membagi halaman HTML dengan fungsi `include()`, redirect dengan fungsi `header()`, memproses form, upload data, hingga membuat dan mengakses `cookie`.

Namun akan tiba saatnya anda butuh sebuah fitur untuk menyimpan seluruh informasi dari pengguna. Dan itulah fungsi dari database. Dengan database, data dari website bisa disimpan dan diakses pada kemudian hari.

Sebenarnya PHP juga memiliki fungsi khusus untuk menyimpan dan membaca data dari file seperti `.txt`, namun database menawarkan fitur tambahan lain seperti kecepatan akses, kemudahan menampilkan dan mengubah data, hingga menghubungkan satu data dengan data lain.

Salah satu aplikasi database yang paling populer dan paling banyak digunakan adalah **MySQL**. Dalam bab ini saya akan membahas cara komunikasi antara PHP dengan MySQL.

## 23.1 Kenapa Harus MySQL?

Dari sekian banyak aplikasi database, MySQL merupakan yang paling populer dan paling banyak digunakan terutama bersama PHP. Salah satu alasannya karena MySQL bersifat open source dan gratis (setidaknya hingga saat ini).

Tentu saja selain MySQL juga tersedia aplikasi database lain seperti **Oracle**, **PostgreSQL**, dan **Microsoft SQL Server**. PHP pun juga bisa digunakan untuk berkomunikasi dengan aplikasi database ini, tapi relatif jarang.

Walaupun gratis, fitur yang ada di dalam MySQL tidak kalah dengan aplikasi database berbayar seperti **Oracle**. Salah satu hal yang sering dilakukan oleh programmer pemula (termasuk saya dulu) adalah mencoba mempelajari Oracle dan mengesampingkan MySQL karena menganggapnya hanya cocok untuk proyek-proyek kecil.

Sebenarnya tidak salah, tapi lebih ke “belum waktunya”. Aplikasi database seperti **Oracle** dan **Microsoft SQL Server** biasanya hanya digunakan oleh perusahaan besar, dan di Indonesia jumlahnya tidak begitu banyak.

Sebagai pembuktian, silahkan anda cari situs web hosting yang menyediakan **Oracle** dan **Microsoft SQL Server**. Nyaris tidak ada. Kalaupun ada harganya selangit (karena harus membayar biaya lisensi). Mayoritas web hosting umumnya hanya menyediakan **MySQL** (dan **MariaDB**) dengan sistem operasi **Linux**.

Untuk itulah saya menyarankan untuk fokus dulu ke MySQL sebelum melirik database lain. Apalagi jika anda ingin fokus ke web programming, 90% akan menggunakan **MySQL** (atau **MariaDB**).

Contohnya, ensiklopedi terbesar di dunia: [Wikipedia juga menggunakan MySQL<sup>1</sup>](#). Bisa dibayangkan betapa besarnya ukuran database wikipedia, dan semua itu bisa dihandle oleh MySQL.

Setelah mempelajari MySQL, silahkan lanjut ke aplikasi database lainnya seperti **Oracle** atau **Microsoft SQL Server**.

## 23.2 MySQL atau MariaDB?

Seperti yang pernah saya bahas saat proses instalasi XAMPP (bab 3). Untuk XAMPP versi terbaru tidak lagi menggunakan MySQL, tetapi **MariaDB**.

Pada dasarnya, MariaDB adalah *cloningan* MySQL. Ini dilakukan kalangan programmer untuk jaga-jaga jika suatu saat MySQL tidak lagi dikembangkan/menjadi berbayar. Karena saat ini pengembangan MySQL berada di perusahaan induknya, **Oracle**. Yang sebenarnya juga saingan database MySQL.

Seluruh kode program yang akan kita pelajari bisa berjalan di MySQL maupun MariaDB tanpa perubahan apapun. Fungsi-fungsi PHP yang digunakan juga tetap diawali dengan perintah “`mysqli_`”.

## 23.3 Mengenal SQL (Structured Query Language)

Untuk bisa menggunakan MySQL, mau tidak mau anda juga harus paham cara penggunaan bahasa **SQL (Structured Query Language)**. Mirip seperti HTML yang digunakan untuk membuat halaman web, SQL digunakan untuk membuat database.

SQL bersifat generik dan hampir semua aplikasi database menggunakan (tidak hanya MySQL saja). Tentunya MySQL menambahkan beberapa perintah khusus yang hanya berlaku di MySQL dan tidak terdapat di Oracle, begitu juga sebaliknya.

Materi tentang MySQL dan bahasa SQL sangat luas. Karena fokus buku ini adalah tentang kode PHP, saya hanya membahas sekilas tentang SQL.



Untuk penjelasan yang lebih detail terkait MySQL (dan bahasa SQL), saya berencana membuat eBook **MySQL Uncover**. Saat ini eBook tersebut masih belum tersedia. Jika anda tertarik mempelajarinya, bisa mengikuti tutorial belajar [MySQL Dasar<sup>2</sup>](#) di situs duniaIlkom.

## 23.4 Menjalankan MySQL Server dan MySQL Client

Sama seperti web server Apache dan web browser, aplikasi MySQL juga menggunakan arsitektur serupa, yakni ada yang bertindak sebagai **MySQL Server**, dan ada yang berfungsi sebagai

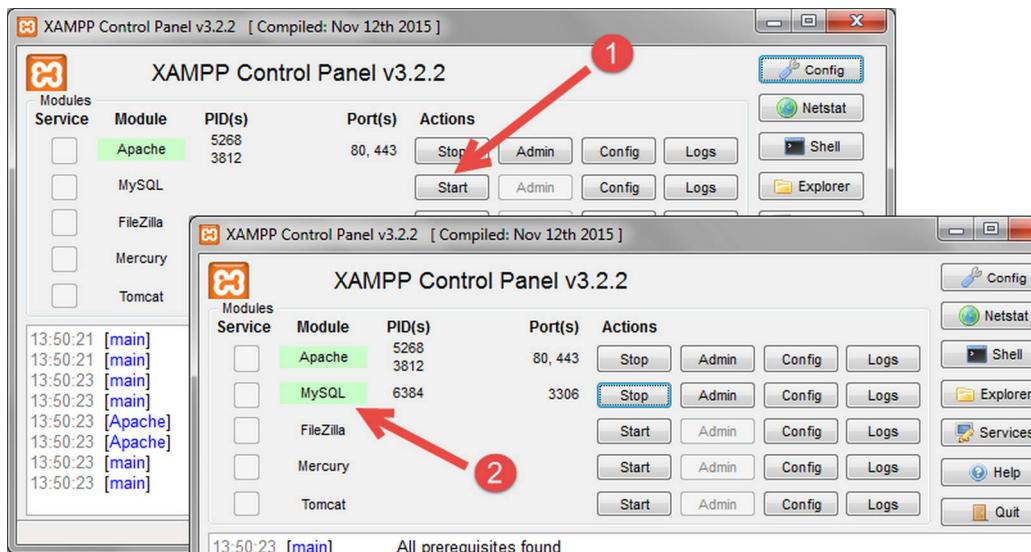
<sup>1</sup><https://en.wikipedia.org/wiki/Wikipedia:FAQ/Technical>

<sup>2</sup><http://www.duniaIlkom.com/tutorial-belajar-mysql-dan-index-artikel-mysql/>

**MySQL Client.** Untungnya XAMPP sudah membundle kedua aplikasi ini sehingga kita bisa langsung pakai.

**MySQL Server** adalah aplikasi yang akan memproses dan mengolah seluruh database. Sistem kerjanya sangat mirip seperti web server Apache yang mengolah permintaan dari web browser.

Pertama, mari kita jalankan aplikasi MySQL Server. Caranya sangat mudah, silahkan buka **XAMPP Control Panel**, lalu klik tombol Start pada baris MySQL, seperti gambar berikut:



Gambar: Menjalankan MySQL Server dari XAMPP Control Panel

Setelah mengklik tombol Start (1), pastikan ada warna hijau di belakang tulisan MySQL (2). Kalau berwarna merah, berarti MySQL belum berjalan atau bentrok dengan aplikasi MySQL Server lain.

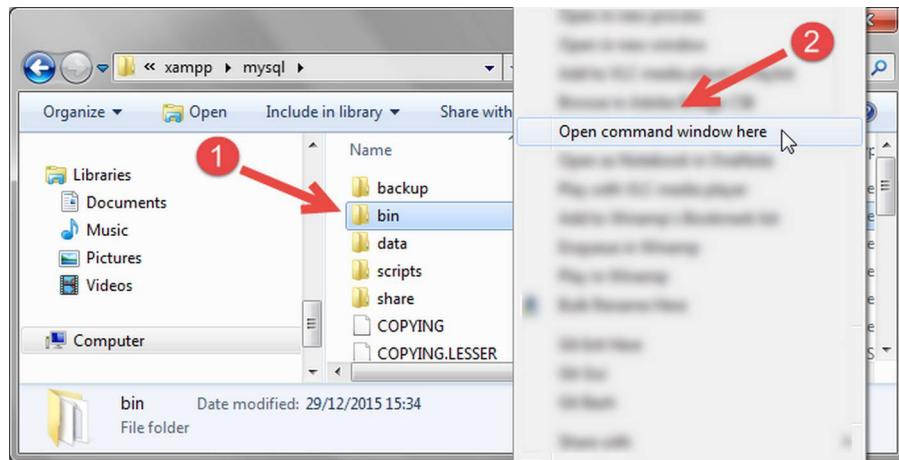
Jika tidak ada masalah, saatnya kita menjalankan aplikasi **MySQL Client** untuk berkomunikasi dengan MySQL Server ini.

Terdapat berbagai aplikasi yang bisa digunakan. Kita bisa menggunakan PHP sebagai MySQL Client. Inilah yang akan kita pelajari nantinya, yakni bagaimana cara mengakses database MySQL (yang berada di MySQL Server) dari kode program PHP.

Sebelum sampai kesana, saya ingin membahas terlebih dahulu bahasa SQL yang digunakan untuk berkomunikasi dengan MySQL Server. Ini akan lebih mudah jika menggunakan MySQL Client yang dijalankan dari **cmd** Windows. Yup, saya akan mengajak anda mengakses database MySQL menggunakan layar hitam dengan teks putih layaknya di film-film hacker.

Jika anda menggunakan Windows Vista keatas (Windows 7, 8, dan 10) silahkan buka folder instalasi xampp lalu buka folder mysql menggunakan Windows Explorer. Karena saya menginstall aplikasi XAMPP di C, maka alamatnya adalah C:\xampp\mysql.

Setelah folder tersebut terbuka, tahan tombol **SHIFT** dan **klik kanan** folder **bin**. Akan tampil beberapa menu, pilihlah **“Open Command Windows Here”**, seperti gambar berikut:



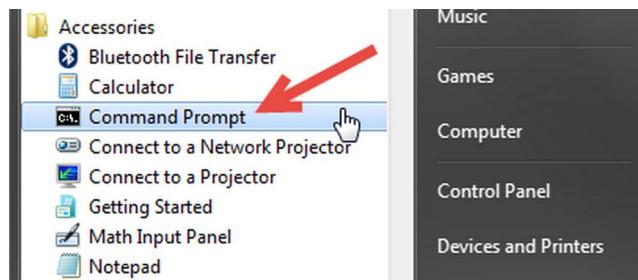
Gambar: Tahan tombol SHIFT, lalu klik kanan. Akan muncul menu “Open Command Windows Here”

Setelah itu akan tampil jendela cmd windows dengan lokasi alamat di c:\xampp\mysql\bin:



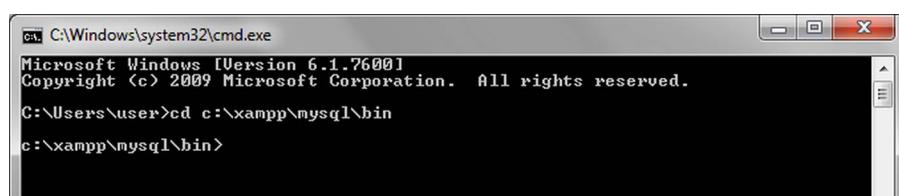
Gambar: Jendela cmd windows terbuka di alamat c:\xampp\mysql\bin

Sebagai alternatif, anda bisa buka manual aplikasi cmd dari menu Start → All programs → Accessories → Command Prompt. Atau bisa juga dengan mengetik cmd di kotak pencarian Start Menu.



Gambar: Cari aplikasi command prompt dari Start Menu

Setelah terbuka, ketik kode cd c:\xampp\mysql\bin. Sekarang folder aktif akan pindah ke c:\xampp\mysql\bin. Kode “cd” adalah perintah khusus di dalam cmd windows untuk pindah folder, perintah ini merupakan singkatan dari “*change directory*”.

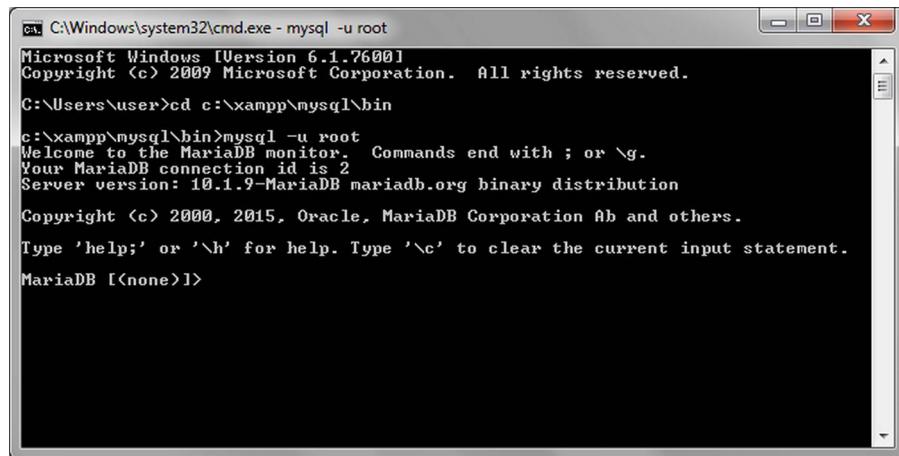


Gambar: Alamat cmd windows akan pindah ke c:\xampp\mysql\bin

Sekarang kita sudah bisa mulai menjalankan MySQL Client. Silahkan ketik perintah berikut lalu

tekan tombol *Enter*: **mysql -u root**.

Jika tidak ada masalah, akan tampil pesan pembuka dari MySQL (atau lebih tepatnya MariaDB):



```
C:\Windows\system32\cmd.exe - mysql -u root
Microsoft Windows [Version 6.1.7601]
Copyright <c> 2009 Microsoft Corporation. All rights reserved.

C:\Users\user>cd c:\xampp\mysql\bin

c:\xampp\mysql\bin>mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 10.1.9-MariaDB mariadb.org binary distribution

Copyright <c> 2000, 2015, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Gambar: Welcome to the MariaDB monitor

Di dalam folder **c:\xampp\mysql\bin** terdapat file **mysql.exe**, file inilah yang kita panggil dengan perintah **mysql -u root**. Tambahan perintah **-u root** berarti saya ingin masuk sebagai user **root**. User **root** adalah user yang memiliki hak akses tertinggi di dalam MySQL, dimana ia dapat mengakses seluruh database yang ada (sering juga disebut sebagai *superadmin*).

Secara default, XAMPP membuat user root MySQL **tanpa password**. Oleh karena itulah kita bisa masuk hanya dengan perintah **mysql -u root** (tidak perlu menulis password). Dari sisi keamanan, sebuah user tanpa pasword sangat tidak disarankan. Tapi karena disini kita hanya belajar cara penggunaan MySQL secara offline, menurut saya ini tidak terlalu masalah. Nantinya kita bisa memberikan password kepada user root maupun user lain.

## 23.5 Sekilas Tentang Teori Database

Sebelum kita mulai membuat database dengan MySQL, saya akan jelaskan secara singkat sedikit teori seputar database.

MySQL adalah salah satu aplikasi database yang menerapkan konsep **relasional**. Atau istilah lengkapnya, MySQL adalah sebuah aplikasi **RDBMS (Relational Database Management System)**. Di dalam defenisi ini, sebuah database terdiri dari kumpulan tabel yang saling berhubungan (saling ber-relasi).

Tabel yang ada di dalam relational database mirip seperti tabel dalam aplikasi Microsoft Excel. Berikut contohnya:

NIM	Nama	Tempat Lahir	Tanggal Lahir	Fakultas	Jurusan	IPK
15002032	Rina Kumala Sari	Jakarta	28/06/1997	Ekonomi	Akuntansi	3.4
13012012	James Situmorang	Medan	02/04/1995	Kedokteran	Kedokteran Gigi	2.7
14005011	Riana Putria	Padang	23/11/1996	FMIPA	Kimia	3.1
15021044	Rudi Permana	Bandung	22/08/1994	FASILKOM	Ilmu Komputer	2.9
15003036	Sari Citra Lestari	Jakarta	31/12/1997	Ekonomi	Manajemen	3.5

Gambar: Tabel Mahasiswa

Tabel diatas berisi data mahasiswa yang terdiri dari beberapa kolom dan baris. Satu kolom berisi data dengan jenis dan format tertentu. Sebagai contoh, kolom **Tanggal Lahir** berisi data tanggal lahir mahasiswa. Data ini harus diinput dengan format **dd/mm/yyyy**.

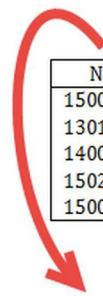
Di dalam MySQL, kita juga harus mendefenisikan apa saja jenis data yang bisa diinput pada masing-masing kolom, apakah data teks (*string*), angka (*integer* atau *float*), tanggal (*date*), atau tipe data lainnya. Karena menggunakan perintah bahasa inggris, kolom ini dikenal dengan sebutan “**column**”.

Baris tabel dikenal juga sebagai “**row**” atau “**record**”. Satu baris tabel berisi data yang lengkap, misalnya data baris pertama terdiri dari NIM: 15002032, Nama: Rina Kumala Sari, Tempat Lahir: 28/06/1997, dst.

Sebuah **sel tabel** di kenal juga dengan istilah **field**. Sebagai contoh Riana Putria adalah *field*, FASILKOM adalah *field*, dst. Di dalam MS Excel, ini biasanya disebut sebagai sel (*cell*).

Sebuah database bisa terdiri dari kumpulan beberapa tabel yang saling berhubungan. Walaupun dalam prakteknya kita bisa saja membuat database yang berisi hanya 1 tabel seperti contoh diatas.

Mari kita bahas kembali kolom-kolom yang ada di tabel mahasiswa. Kolom **IPK** merupakan singkatan dari *Indeks Prestasi Kumulatif*, yakni nilai rata-rata mahasiswa dalam setiap semester. Seharusnya nilai IPK ini bukan ditulis manual, tetapi di generate secara otomatis. Untuk ini kita bisa membuat 1 buah tabel baru, seperti gambar di bawah ini:



NIM	Nama	Tempat Lahir	Tanggal Lahir	Fakultas	Jurusan
15002032	Rina Kumala Sari	Jakarta	28/06/1997	Ekonomi	Akuntansi
13012012	James Situmorang	Medan	02/04/1995	Kedokteran	Kedokteran Gigi
14005011	Riana Putria	Padang	23/11/1996	FMIPA	Kimia
15021044	Rudi Permana	Bandung	22/08/1994	FASILKOM	Ilmu Komputer
15003036	Sari Citra Lestari	Jakarta	31/12/1997	Ekonomi	Manajemen

No	NIM	IP Semester 1	IP Semester 2	IP Semester 3	...	IPK
1	15002032	3.7	3.1			3.4
2	13012012	2.5	3.0	2.9		2.7
3	14005011	3.5	2.9	3.3		3.1
4	15021044	2.2	3.2	3.1		2.9
5	15003036	3.8	3.2			3.5

Gambar: Tabel data mahasiswa dan tabel IPK

Disini saya memiliki tabel kedua yang berisi **nilai IP** setiap mahasiswa untuk setiap semester. Perhatikan tanda panah merah, inilah hubungan antara tabel pertama dengan tabel kedua. Jika

saya ingin mencari IPK dari mahasiswa bernama **Sari Citra Lestari**, saya akan buka tabel pertama, mencatat NIM dari Sari Citra Lestari, lalu melihat tabel kedua berdasarkan NIM tadi. Hasil IPK akan didapat di kolom terakhir.

Dalam tabel pertama yang berisi data mahasiswa, kolom **NIM** disebut sebagai **primary key**. Yakni kolom yang berfungsi sebagai identitas dari setiap baris di tabel tersebut. Kolom yang berfungsi sebagai **primary key** harus unik, artinya tidak boleh ada data yang berulang.

Kita bisa mendefenisikan kolom apa saja sebagai primary key, selama dapat dipastikan datanya tidak berulang. Di tabel kedua, kolom **No** lah yang akan saya defenisikan sebagai **primary key**. Setiap data IPK akan memiliki nomor yang berbeda-beda.

Kolom **NIM** di tabel kedua dikenal juga dengan istilah **foreign key**. Yakni kolom yang berfungsi sebagai penghubung ke tabel lain (dalam hal ini untuk menghubungkan dengan kolom pertama).

Untuk data yang cukup kompleks, kita mungkin butuh 5, 10 atau 20 tabel untuk menyimpan seluruh informasi. Semuanya saling berhubungan dengan **primary key** dan **foreign key**. Kumpulan dari tabel inilah yang membentuk **sebuah database**.



Istilah-istilah yang saya bahas disini baru sebagian kecil dari teori database. Materi tentang database sendiri sangat luas. Di jurusan ilmu komputer / IT, terdapat mata kuliah khusus yang membahas database. **DBA (Database Administrator)** juga bisa menjadi pilihan karir yang kerjanya hanya menangani masalah database. Jika anda berminat untuk mempelajari hal ini lebih lanjut, bisa mencari buku tentang teori database.

## 23.6 Mengenal Perintah Dasar SQL di MySQL

Untuk berkomunikasi dengan database server, kita harus menggunakan bahasa **SQL (Structured Query Language)**. Bahasa SQL mirip dengan bahasa inggris sehari-hari, seperti **SHOW DATABASE**, **USE DATABASE**, **DROP TABLE**, **SELECT**, dst.

SQL sendiri merupakan bahasa universal yang digunakan hampir oleh seluruh aplikasi database, termasuk **MySQL**, **Oracle**, **MS SQL Server**, dll. Namun setiap aplikasi memiliki *dialek* atau ciri khas masing-masing. Kali ini kita akan mempelajari bahasa SQL yang digunakan oleh MySQL.

Secara umum, perintah bahasa SQL atau sering disingkat sebagai “*query*” saja, terbagi menjadi 4 jenis:

- Perintah untuk membuat data (**CREATE**).
- Perintah untuk membaca data (**READ**).
- Perintah untuk mengubah data (**UPDATE**).
- Perintah untuk menghapus data (**DELETE**).

Keempat istilah ini sering disingkat sebagai **CRUD** (*CREATE*, *READ*, *UPDATE*, dan *DELETE*). Oleh karena itu aplikasi yang berfungsi untuk menjalankan perintah-perintah ini disebut juga sebagai aplikasi **CRUD**.

Baik, mari kita mulai mempelajari perintah-perintah SQL. Silahkan buka MySQL Client dari cmd Windows, lalu masuk sebagai user root:

Gambar: Masuk sebagai user root di cmd MySQL Client

## Membuat, Mengakses dan Menghapus Database MySQL

Hal pertama yang harus kita lakukan adalah membuat sebuah database. Untuk membuatnya, silahkan ketik perintah berikut, lalu akhiri dengan tombol Enter:

```
MariaDB [(none)]> CREATE DATABASE universitas;
```

Jika tidak ada masalah, tampilannya adalah sebagai berikut:

Gambar: Database “universitas” sudah berhasil dibuat

Sekarang, di dalam MySQL sudah terbentuk sebuah database “universitas”. Isinya tentu belum ada tabel apapun karena kita memang belum membuatnya.

Perhatikan cara penulisan perintah SQL di dalam MySQL. Sama seperti PHP, setiap perintah harus diakhiri dengan tanda titik koma ( ; ). Seluruh perintah di dalam MySQL bersifat case insensitive, yang tidak membedakan huruf besar dan huruf kecil. Perintah diatas juga bisa ditulis dengan huruf kecil semua, seperti:

```
create database universitas;
```

Namun sudah menjadi standar tidak resmi kalau perintah SQL ditulis dengan HURUF BESAR, sedangkan untuk nama variabel seperti “**universitas**” tetap ditulis huruf kecil. Dengan demikian kita bisa membedakan mana perintah SQL dan mana variabel yang dibuat oleh programmer.

Khusus untuk nama database, nama tabel, nama kolom tabel, dan nama variabel lain yang bisa kita tentukan sendiri, sebaiknya tidak menukar huruf besar atau kecil. Misalkan saya membuat database “**unversitas**”. Di sistem windows, database ini bisa diakses dengan “**Universitas**” maupun “**UNIVERSITAS**” yang artinya tidak membedakan huruf besar dan kecil. Tapi di sistem **Linux** ini tidak berlaku.

Jadi sebaiknya konsisten untuk penamaan variabel. Jika memutuskan untuk menggunakan huruf kecil, gunakan huruf kecil untuk semua variabel.

Setelah kita menekan tombol **Enter**, akan tampil keterangan: **Query OK, 1 row affected (0.00 sec)**. Ini adalah *feedback* atau laporan dari perintah yang baru saja kita ketik. “**Query OK**” berarti query atau perintah tersebut tidak ada yang salah. “**1 row affected**” berarti ada 1 baris yang diubah akibat perintah ini. “**(0.00 sec)**” adalah waktu yang diperlukan MySQL untuk menjalankannya.

Karena proses pembuatan database cukup sederhana, MySQL Server hanya butuh sepersekian detik untuk menjalankan proses ini. Ini juga karena komputer yang saya gunakan tidak ada proses lain yang sedang berjalan.

Bagaimana jika perintah (query) yang ditulis terdapat kesalahan? Mari kita coba. Silahkan jalankan perintah berikut:

```
MariaDB [(none)]> CREAT DATABASE universitas;
```

```
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual  
that corresponds to your MariaDB server version for the right syntax to  
use near 'CREAT DATABASE universitas' at line 1
```

Disini saya sengaja menghapus huruf “E” dari “CREATE” menjadi “CREAT”. MySQL akan langsung komplain dan menampilkan bagian yang salah.

Perintah **CREATE DATABASE** digunakan untuk membuat database. Agar bisa menggunakan-nya, kita harus memilih database ini dengan perintah **USE**. Berikut querynya:

```
MariaDB [(none)]> USE universitas;
```

**Database** changed

Perhatikan perubahan dari **MariaDB [(none)]** menjadi **MariaDB [universitas]**. Ini artinya kita sudah menetapkan sebuah database. Setiap query yang ditulis akan berefek ke database universitas saja.

Bagaimana jika saya memutuskan untuk menukar database? Cukup jalankan perintah yang sama dengan format **USE nama\_database**.

Untuk melihat database apa saja yang saat ini tersedia di dalam MySQL, bisa menggunakan query **SHOW DATABASES**, seperti berikut:

```
MariaDB [universitas]> SHOW DATABASES;
```

```
+-----+
| Database      |
+-----+
| information_schema |
| mysql          |
| performance_schema |
| phpmyadmin      |
| test           |
| universitas    |
+-----+
7 rows in set (0.11 sec)
```

Perlu diperhatikan bahwa perintahnya adalah **SHOW DATABASES**, bukan **SHOW DATABASE** (ada tambahan huruf “S” menjadi “**DATABASES**”). Kesalahan akibat lupa menulis “S” ini cukup sering terjadi.

Dari tampilan yang ada terlihat selain database **universitas**, sudah ada beberapa database lain. Diantaranya merupakan database bawaan MySQL (seperti **information\_schema** dan **mysql**), dan database bawaan XAMPP (seperti **phpmyadmin**). Database ini sebaiknya dibiarkan saja, dan disarankan untuk tidak mengubah atau menghapusnya.

Untuk menghapus database, bisa menggunakan query **DROP DATABASE**. Misalkan saya ingin menghapus database **universitas**, perintahnya adalah:

```
MariaDB [universitas]> DROP DATABASE universitas;
```

```
Query OK, 0 rows affected (0.13 sec)
```

Perintah ini sangat powerful. Database akan dihapus **beserta seluruh tabel** di dalamnya (jika ada). Di dalam MySQL juga tidak dikenal perintah “*undo*”. Jadi anda perlu berhati-hati terhadap query **DROP** ini.



Untuk mempercepat pengetikan, tekan tanda panah atas di keyboard. Jendela **cmd** windows akan menampilkan perintah-perintah yang sebelumnya sudah diinput. Silahkan anda buat lagi database **universitas** jika ikut terhapus karena perintah **DROP** ini.

## Membuat dan Menghapus Tabel MySQL

Setelah database **universitas** dipilih, kita bisa mulai membuat tabel.

Sebagai informasi tambahan, membuat database dan tabel tidak sering kita lakukan. Umumnya kegiatan ini hanya perlu sekali di awal, dan selesai. Oleh karena itu jangan heran jika anda sering lupa query untuk membuat tabel. Sepanjang pengembangan website kita hanya sering menggunakan perintah **READ**, **UPDATE** dan **DELETE**.

Untuk membuat tabel, querynya cukup sederhana. Berikut format dasar pembuatan tabel di dalam MySQL:

```
CREATE TABLE nama_tabel ( nama_kolom1 jenis_kolom1,
nama_kolom2 jenis_kolom2, nama_kolom3 jenis_kolom3, ... );
```

Yang cukup kompleks adalah menentukan jenis kolom. Ini sama halnya dengan mempelajari tipe data di dalam PHP. Apakah itu *string*, *integer*, *date*, dll. Selain itu kita juga bisa memberikan syarat tambahan untuk setiap kolom, apakah tidak boleh kosong (NOT NULL), tanpa tanda (UNSIGNED), atau penomoran otomatis (AUTO\_INCREMENT).

Berikut contoh query untuk membuat sebuah tabel **data\_mahasiswa**:

```
MariaDB [universitas]> CREATE TABLE data_mahasiswa ( nim char(9),
nama char(50), umur int, tempat_lahir char(50), jurusan char (30) );
```

```
Query OK, 0 rows affected (0.40 sec)
```

Dengan query diatas, saya membuat tabel **data\_mahasiswa** yang terdiri dari 5 kolom: **nim**, **nama**, **umur**, **tempat\_lahir**, dan **jurusan**. Penulisan nama kolom diikuti dengan jenis dari kolom tersebut dan dipisah dengan tanda koma.

Untuk kolom yang berisi huruf (*string*), bisa menggunakan tipe data **char**. Angka di dalam kurung adalah jumlah maksimal karakter yang bisa ditampung, misalnya untuk kolom nama, saya menginstruksikan MySQL untuk menyediakan sebanyak 50 karakter.

Untuk kolom yang berisi angka, MySQL menyediakan beberapa tipe data. Dalam contoh diatas, saya memilih **integer**, yang bisa disingkat menjadi **int**. Mengenai jenis-jenis kolom akan kita bahas sesaat lagi.

Setelah tabel dibuat, mari kita pastikan dengan perintah **SHOW TABLES**:

```
MariaDB [universitas]> SHOW TABLES;
```

```
+-----+
| Tables_in_universitas |
+-----+
| data_mahasiswa        |
+-----+
1 row in set (0.07 sec)
```

Untuk melihat struktur dari tabel (seperti nama kolom dan tipe datanya) bisa menggunakan query **DESCRIBE** atau **DESC**, seperti berikut ini:

```
MariaDB [universitas]> DESCRIBE data_mahasiswa;
```

```
+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| nim        | char(9)   | YES  |     | NULL    |       |
| nama       | char(50)  | YES  |     | NULL    |       |
| umur       | int(11)   | YES  |     | NULL    |       |
| tempat_lahir | char(50) | YES  |     | NULL    |       |
| jurusan    | char(30)  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
5 rows in set (0.18 sec)
```

Bagaimana dengan menghapus tabel? Bisa menggunakan query **DROP TABLE**. Berikut contohnya:

```
MariaDB [universitas]> DROP TABLE data_mahasiswa;
Query OK, 0 rows affected (0.34 sec)
```

```
MariaDB [universitas]> SHOW TABLES;
Empty set (0.00 sec)
```

Sekarang tabel **data\_mahasiswa** sudah dihapus. Sama seperti **DROP DATABASE**, query ini harus digunakan dengan hati-hati. Tabel yang sudah terhapus tidak dapat dikembalikan.

## Jenis Tipe Data untuk Kolom Tabel MySQL

MySQL mendukung banyak tipe data yang bisa digunakan untuk membuat kolom. Mari kita bahas beberapa diantaranya.

### Tipe Data Angka Bulat (Integer)

Untuk kolom dengan data angka bulat (*integer*), MySQL menyediakan 5 pilihan yang dibedakan berdasarkan jangkauannya:

- **TINYINT**: Mendukung angka bulat dari -128 hingga 127 (*signed*), atau 0 hingga 255 (*unsigned*).
- **SMALLINT**: Mendukung angka bulat dari -32768 hingga 32767 (*signed*), atau 0 hingga 65535 (*unsigned*).
- **MEDIUMINT**: Mendukung angka bulat dari -8388608 hingga 8388607 (*signed*), atau 0 hingga 16777215 (*unsigned*).
- **INT**: Mendukung angka bulat dari -2147483648 hingga 2147483647 (*signed*), atau 0 hingga 4294967295 (*unsigned*).
- **BIGINT**: Mendukung angka bulat dari -9223372036854775808 hingga 9223372036854775807 (*signed*), atau 0 hingga 1844674407370951615 (*unsigned*).

Pemilihan tipe data ini tergantung kebutuhan. Misalkan untuk nomor urut absensi pada suatu kelas yang memiliki maksimal 30 orang, tipe data **TINYINT** sudah mencukupi.

Namun jika kita bermaksud membuat absensi seluruh rakyat indonesia, tipe data **TINYINT** sampai dengan **MEDIUMINT** tidak akan cukup. Kita harus memilih **INT** atau **BIGINT**.

Format penulisan query untuk kolom dengan tipe data integer adalah:

```
INT[(M)] [UNSIGNED] [ZEROFILL]
```

Dalam deklarasi tipe data integer, nilai **M** digunakan untuk mengatur jumlah digit yang disediakan untuk menampilkan data. Misalkan saya mendefinisikan suatu kolom dengan **INT(5)**, ketika diinput angka 102, MySQL akan menambahkan spasi sebanyak 2 buah di depan angka agar tampilan data menjadi 5 digit (istilah programmingnya: *padding left*).

Nilai **M** tidak mempengaruhi jangkauan dari kolom tersebut. Jika angka yang diinput melebihi digit **M**, MySQL tetap menyimpan hasilnya (selama masih dalam jangkauan tipe data tersebut). Jika saya deklarasikan **INT(4)**, nilai maksimal tetap 2,147,483,647 bukan 9999.

Setiap tipe data integer dapat di definisikan sebagai **UNSIGNED**, dimana kita mengorbankan nilai negatif untuk mendapatkan jangkauan nilai positif yang lebih besar. Artinya kolom tersebut hanya bisa diisi dengan angka positif saja.

Selain tambahan **UNSIGNED**, kolom integer juga bisa ditambah atribut **ZEROFILL**. **ZEROFILL** berhubungan dengan nilai **M** yang akan mengganti tempat spasi menjadi angka 0. Misalkan jika saya mendefinisikan **INT(5) ZEROFILL** dan menginput angka 102, hasilnya akan menjadi 00102.

## Tipe Data Angka Pecahan (Float)

Untuk tipe data angka pecahan (*float*), terdapat 3 pilihan: **FLOAT**, **DOUBLE**, dan **DECIMAL**.

Tipe data **FLOAT** dan **DOUBLE** menggunakan pendekatan presisi dengan tingkat ketelitian tertentu. Jangkauan tipe data **DOUBLE** lebih besar daripada **FLOAT**.

Untuk penggunaan sehari-hari, sebaiknya kita menggunakan tipe data **DECIMAL** karena terdapat perbedaan pembulatan. Tipe data **FLOAT** dan **DOUBLE** lebih cocok untuk perhitungan data-data sains\*.



\*Beberapa tipe data butuh pembahasan yang cukup detail, seperti apa perbedaan dari **FLOAT**, **DOUBLE**, dan **DECIMAL**? Kemudian kenapa saya menyarankan hanya menggunakan **DECIMAL** saja? Pembahasan seperti ini sepertinya lebih cocok saya jelaskan di eBook **MySQL Uncover** :)

Atau anda bisa membacanya di duniaIlkom: [Tipe Data Numerik MySQL<sup>3</sup>](#).

Penulisan tipe data pecahan di MySQL sedikit berbeda dengan integer. Format penulisannya adalah sebagai berikut:

<sup>3</sup><http://www.duniaIlkom.com/tutorial-mysql-tipe-data-numerik-integer-fixed-point-dan-floating-point/>

```
DECIMAL [(M[,D])] [UNSIGNED] [ZEROFILL]
```

Besar dari jangkaian tipe data **float** tergantung kepada angka [M,D]. M adalah total jumlah digit keseluruhan, sedangkan D adalah jumlah digit dibelakang koma (pecahan). Contohnya **DECIMAL [6,2]** akan mendefinisikan suatu kolom agar memuat 6 digit angka, dengan 4 angka di depan koma, dan 2 digit angka di belakang koma.

Untuk tipe data **DECIMAL**, maksimal nilai untuk M = 65, dan D = 30.

## Tipe Data Teks (String)

Untuk menyimpan tipe data teks atau string, MySQL juga menyediakan banyak tipe data. Saya akan membahas yang sering digunakan, yakni **CHAR**, **VARCHAR**, **TINYTEXT**, **TEXT**, **MEDIUMTEXT**, dan **LONGTEXT**.

Tipe data **CHAR** dan **VARCHAR** adalah tipe data karakter (string) yang akan sering kita gunakan. Format penulisannya adalah sebagai berikut:

```
CHAR [(M)]  
VARCHAR [(M)]
```

Digit M akan menentukan jumlah karakter maksimum yang akan disiapkan MySQL. Sebagai contoh, nilai M=5, maka MySQL menyediakan 5 karakter untuk kolom tersebut. Nilai maksimal M adalah 255 karakter untuk **CHAR**, dan 65535 karakter untuk **VARCHAR**. Jika nilai M tidak di nyatakan, nilai defaultnya adalah M = 1.

Perbedaan antara **CHAR** dan **VARCHAR** terletak dari cara MySQL mengalokasikan ukuran memory untuk menyimpan data. Sebagai contoh, jika saya mendefinisikan kolom bertipe **CHAR(5)**, walaupun huruf atau karakter yang kita inputkan hanya 1 karakter, MySQL tetap menyimpan kolom tersebut untuk 5 karakter.

Namun jika saya menggunakan **VARCHAR(5)** dan menginput data dengan jumlah karakter 2, ukuran penyimpanan hanya akan menggunakan 2 karakter. Disini terlihat **VARCHAR** lebih fleksibel dan efisien. Tetapi untuk fleksibilitas ini, tipe **VARCHAR** memerlukan proses tambahan untuk menyimpan ukuran dari masing-masing data.

Baik **CHAR** maupun **VARCHAR** menyimpan data secara **case insensitif**, dimana huruf besar dan kecil tidak dibedakan.

Bagaimana jika jumlah huruf yang diinput melebihi 65535 karakter? Kita bisa menggunakan tipe data **TEXT**. Tipe data **TEXT** hadir dengan 5 “rasa”:

- **TINYTEXT**: maksimum karakter 255
- **TEXT**: maksimum karakter 65.535
- **MEDIUMTEXT**: maksimum karakter 16.777.215
- **LONGTEXT**: maksimum karakter 4.294.967.295

Terlihat tipe data **LONGTEXT** sanggup menampung hingga **4GB teks**. Sudah sangat besar untuk keperluan menyimpan sebuah teks.

Setiap tipe data text disimpan berdasarkan berapa banyak data yang diinput, bukan berdasarkan jangkauan maksimalnya (ini sama seperti **VARCHAR**). Jika saya mendefinisikan suatu kolom sebagai **LONGTEXT** dan isinya hanya 100 karakter, ukuran penyimpanan yang digunakan hanya sekitar 100 byte, bukan 4GB.

### Tipe Data Biner (Binary)

Tipe data biner adalah jenis kolom yang datanya disimpan secara biner (bit per bit), bukan per karakter seperti **CHAR** atau **TEXT**. Sederhananya, hal ini akan berefek pada case-sensitif data (perbedaan penggunaan huruf besar dan huruf kecil).

Walaupun tidak umum dipraktekkan, kita bisa menyimpan sebuah gambar ke dalam MySQL. Secara teknis, sebuah gambar dibentuk dari ribuan atau jutaan bit-bit data. Data seperti ini hanya bisa disimpan ke dalam kolom bertipe biner.

MySQL menyediakan berbagai tipe data biner: **BINARY**, **VARBINARY**, **TINY BLOB**, **BLOB**, **MEDIUM BLOB**, dan **LONGBLOB**. Semua tipe data ini bersesuaian dengan pasangan non-biner. Misalnya **BINARY** mirip dengan **CHAR**, **VARBINARY** dengan **VARCHAR**, dan **BLOB** dengan **TEXT**.

### Tipe Data Tanggal (DATE)

MySQL menyediakan berbagai tipe data tanggal: **DATE**, **TIME**, **DATETIME**, **TIMESTAMP**, dan **YEAR**. Perbedaannya terletak di bentuk format data. Berikut rinciannya:

- **DATE**: Memiliki format **YYYY-MM-DD**, dengan jangkauan dari 1000-01-01 hingga 9999-12-31. Tanggal 29 November 2016 akan disimpan sebagai 2016-11-29.
- **DATETIME**: Memiliki format **YYYY-MM-DD HH:MM:SS**, dengan jangkauan dari 1000-01-01 00:00:00 hingga 9999-12-31 23:59:59. Tanggal 29 November 2016 pukul 3 lebih 20 menit 45 detik sore akan disimpan sebagai 2016-11-29 15:20:45.
- **TIMESTAMP**: Memiliki format **YYYYMMDDHHMMSS** dengan jangkauan 1 Januari 1970 hingga 19 Januari 2038. Tanggal 29 November 2016 pukul 3 lebih 20 menit 45 detik sore akan disimpan sebagai 20161129152045.
- **TIME**: Memiliki format **HH:MM:SS** dengan jangkauan 00:00:00 hingga 23:59:59. Pukul 3 lebih 20 menit 45 detik sore akan disimpan sebagai 15:20:45
- **YEAR(M)**: Memiliki format **YYYY** atau **YY** tergantung angka M. Jika ditulis sebagai **YEAR(2)**, bisa menyimpan data dari 70 hingga 69 (tahun 1970 hingga 2069). Jika ditulis sebagai **YEAR(4)**, bisa menyimpan data dari 1901 hingga 2155. Nilai M default adalah 4.

## Membuat Tabel mahasiswa

Masih ingat dengan tabel mahasiswa yang saya gunakan di awal bab? Mari kita buat tabelnya.

NIM	Nama	Tempat Lahir	Tanggal Lahir	Fakultas	Jurusan	IPK
15002032	Rina Kumala Sari	Jakarta	28/06/1997	Ekonomi	Akuntansi	3.4
13012012	James Situmorang	Medan	02/04/1995	Kedokteran	Kedokteran Gigi	2.7
14005011	Riana Putria	Padang	23/11/1996	FMIPA	Kimia	3.1
15021044	Rudi Permana	Bandung	22/08/1994	FASILKOM	Ilmu Komputer	2.9
15003036	Sari Citra Lestari	Jakarta	31/12/1997	Ekonomi	Manajemen	3.5

Gambar: Tabel Mahasiswa

```
MariaDB [universitas]> CREATE TABLE mahasiswa ( nim CHAR(8),
nama VARCHAR(100), tempat_lahir VARCHAR(50), tanggal_lahir DATE,
jurusan VARCHAR(50), ipk DECIMAL(3,2), PRIMARY KEY (nim));
Query OK, 0 rows affected (0.06 sec)
```

```
MariaDB [universitas]> DESC mahasiswa;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| nim | char(8) | NO | PRI | NULL | |
| nama | varchar(100) | YES | | NULL | |
| tempat_lahir | varchar(50) | YES | | NULL | |
| tanggal_lahir | date | YES | | NULL | |
| jurusan | varchar(50) | YES | | NULL | |
| ipk | decimal(3,2) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)
```

Tabel mahasiswa sudah dibuat, kini saatnya kita input beberapa data. Kolom **nim** saya set sebagai **primary key**, sehingga kolom ini tidak bisa diisi karakter yang sama (harus unik). Agar tabel kita tidak terlalu panjang, saya tidak menulis kolom **fakultas**.

## Cara Menginput Data ke Tabel MySQL

Untuk menginput data ke dalam tabel MySQL, kita menggunakan query **INSERT**. Terdapat beberapa alternatif penulisan.

Yang pertama, *tanpa menuliskan nama kolomnya*. Jika ditulis seperti ini, data yang diinput harus sesuai dengan jumlah serta urutan kolom dari tabel, seperti contoh berikut:

```
MariaDB [universitas]> INSERT INTO mahasiswa VALUES
('15002032', 'Rina Kumala Sari', 'Jakarta', '1997-06-28', 'Akuntansi', 3.4);
Query OK, 1 row affected (0.04 sec)
```

Perhatikan cara penulisan query **INSERT** ini. Formatnya adalah:

```
INSERT INTO nama_tabel VALUES (nilai_kolom1, nilai_kolom2,..dst);
```

Karena tabel mahasiswa memiliki 7 kolom, maka saya harus menuliskan nilai untuk setiap kolom. Selain itu urutannya juga harus sesuai dengan struktur tabel.

Untuk nilai string/char/date, harus ditulis dalam tanda kutip. Tanda kutip ini bisa **tanda kutip satu** ( ' ) maupun **tanda kutip dua** ( " ). Biasanya yang sering digunakan adalah tanda kutip satu agar tidak bentrok dengan tanda kutip dua yang biasa digunakan PHP.

Untuk nilai angka baik integer maupun float, tidak perlu menggunakan tanda kutip meskipun MySQL tetap akan menerima nilai ini. Namun tidak sebaliknya untuk tipe data string/char/date, jika kita lupa menulis tanda kutip, MySQL akan mengeluarkan pesan error.

Khusus untuk kolom **tanggal\_lahir**, cara penulisan format date harus sesuai dengan aturan MySQL, yakni dengan format **yyyy-mm-dd**.

Untuk memastikan data tersebut telah sukses diinput, kita bisa melihatnya menggunakan query **SELECT**:

```
MariaDB [universitas]> SELECT * FROM mahasiswa;
+-----+-----+-----+-----+-----+
| nim | nama | tempat_lahir | tanggal_lahir | jurusan | ipk |
+-----+-----+-----+-----+-----+
| 15002032 | Rina Kumala Sari | Jakarta | 1997-06-28 | Akuntansi | 3.40 |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Terlihat data “**Rina Kumala Sari**” sudah sukses diinput. Penjelasan tentang query **SELECT** akan saya bahas setelah ini.

Sekarang, bagaimana jika saya ingin menginput sebagian kolom saja? Untuk ini nama kolom tabel harus ditulis seperti contoh query berikut:

```
MariaDB [universitas]> INSERT INTO mahasiswa (nim, nama, tempat_lahir, tanggal_lahir) VALUES ('13012012', 'James Situmorang', 'Medan', '1995-04-02');
Query OK, 1 row affected (0.13 sec)
```

```
MariaDB [universitas]> SELECT * FROM mahasiswa;
+-----+-----+-----+-----+-----+
| nim | nama | tempat_lahir | tanggal_lahir | jurusan | ipk |
+-----+-----+-----+-----+-----+
| 13012012 | James Situmorang | Medan | 1995-04-02 | NULL | NULL |
| 15002032 | Rina Kumala Sari | Jakarta | 1997-06-28 | Akuntansi | 3.40 |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Sebelum query **VALUES**, kita harus menulis apa saja nama kolom yang ingin diinput. Format dasarnya adalah sebagai berikut:

```
INSERT INTO nama_tabel (nama_kolom1, nama_kolom2, nama_kolom3, ..dst)
VALUES (nilai_kolom1, nilai_kolom2, nilai_kolom3,..dst);
```

Dari hasil query **SELECT** terlihat kolom yang tidak diinput datanya ditampilkan sebagai **NULL**.

Dengan menuliskan nama kolom, kita juga tidak harus menginput data sesuai urutan struktur tabel, berikut contohnya:

```
MariaDB [universitas]> INSERT INTO mahasiswa (tanggal_lahir, ipk, nama, nim)
VALUES ('1996-11-23', 3.1, 'Riana Putria', '14005011');
Query OK, 1 row affected (0.00 sec)
```

```
MariaDB [universitas]> SELECT * FROM mahasiswa;
+-----+-----+-----+-----+-----+
| nim | nama | tempat_lahir | tanggal_lahir | jurusan | ipk |
+-----+-----+-----+-----+-----+
| 13012012 | James Situmorang | Medan | 1995-04-02 | NULL | NULL |
| 14005011 | Riana Putria | NULL | 1996-11-23 | NULL | 3.10 |
| 15002032 | Rina Kumala Sari | Jakarta | 1997-06-28 | Akuntansi | 3.40 |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Disini saya mengacak urutan kolom. Ini bisa dilakukan karena nama kolom yang ditulis tetap bersesuaian dengan nilainya.

Bagaimana cara menginput banyak data sekaligus? Tentunya kita bisa menjalankan perintah **INSERT** beberapa kali, atau bisa juga dipisah dengan tanda koma seperti contoh berikut ini:

```
MariaDB [universitas]> INSERT INTO mahasiswa
(nim, nama, tempat_lahir, tanggal_lahir, jurusan) VALUES
('15021044', 'Rudi Permana', 'Bandung', '1994-08-22', 'Ilmu Komputer'),
('15003036', 'Sari Citra Lestari', 'Jakarta', '1997-12-31', 'Manajemen');

Query OK, 2 rows affected (0.00 sec)
Records: 2 Duplicates: 0 Warnings: 0
```

```
MariaDB [universitas]> SELECT * FROM mahasiswa;
+-----+-----+-----+-----+-----+
| nim | nama | tempat_lahir | tanggal_lahir | jurusan | ipk |
+-----+-----+-----+-----+-----+
| 13012012 | James Situmorang | Medan | 1995-04-02 | NULL | NULL |
| 14005011 | Riana Putria | NULL | 1996-11-23 | NULL | 3.10 |
| 15002032 | Rina Kumala Sari | Jakarta | 1997-06-28 | Akuntansi | 3.40 |
| 15003036 | Sari Citra Lestari | Jakarta | 1997-12-31 | Manajemen | NULL |
| 15021044 | Rudi Permana | Bandung | 1994-08-22 | Ilmu Komputer | NULL |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

Disini saya menginput dua data sekaligus. Setiap data dipisah dengan tanda koma.

Yang juga perlu menjadi catatan, kolom **nim** tidak boleh diinput dengan data yang sama (karena kita telah men-setnya sebagai **PRIMARY KEY**). Jika ini terjadi, MySQL akan mengeluarkan pesan error:

```
MariaDB [universitas]> INSERT INTO mahasiswa VALUES
('15002032','Rina Kumala Sari','Jakarta','1997-06-28','Akuntansi', 3.4);

ERROR 1062 (23000): Duplicate entry '15002032' for key 'PRIMARY'
```

Error seperti ini umum terjadi. Untuk menanganinya, sebelum melakukan query **INSERT** kita harus memeriksa dulu apakah sudah ada data **nim** yang sama atau tidak. Nantinya ini bisa dilakukan dari PHP.

## Cara Menampilkan Data dari Tabel MySQL

Dari semua perintah query MySQL, menampilkan data tabel merupakan perintah yang paling sering dilakukan. Untuk ini kita menggunakan query **SELECT**. Query **SELECT** memiliki banyak variasi, namun secara umum format dasarnya adalah sebagai berikut:

```
SELECT nama_kolom FROM nama_tabel [WHERE kondisi];
```

Untuk menampilkan seluruh kolom dari sebuah tabel, **nama\_kolom** bisa diganti dengan *tanda bintang* (\*). Jika saya ingin menampilkan seluruh kolom dari tabel mahasiswa, querynya adalah sebagai berikut:

```
MariaDB [universitas]> SELECT * FROM mahasiswa;
```

```
+-----+-----+-----+-----+-----+-----+
| nim | nama | tempat_lahir | tanggal_lahir | jurusan | ipk |
+-----+-----+-----+-----+-----+-----+
| 13012012 | James Situmorang | Medan | 1995-04-02 | NULL | NULL |
| 14005011 | Riana Putria | NULL | 1996-11-23 | NULL | 3.10 |
| 15002032 | Rina Kumala Sari | Jakarta | 1997-06-28 | Akuntansi | 3.40 |
| 15003036 | Sari Citra Lestari | Jakarta | 1997-12-31 | Manajemen | NULL |
| 15021044 | Rudi Permana | Bandung | 1994-08-22 | Ilmu Komputer | NULL |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.03 sec)
```

Bagaimana jika ingin menampilkan kolom **nim** dan **nama** saja? Caranya dengan menuliskan nama kolom setelah perintah **SELECT**:

```
MariaDB [universitas]> SELECT nim,nama FROM mahasiswa;
```

```
+-----+-----+
| nim | nama |
+-----+-----+
| 13012012 | James Situmorang |
| 14005011 | Riana Putria |
| 15002032 | Rina Kumala Sari |
| 15003036 | Sari Citra Lestari |
| 15021044 | Rudi Permana |
+-----+
5 rows in set (0.04 sec)
```

Untuk menampilkan data yang lebih spesifik, kita bisa menambahkan kondisi. Misalnya saya ingin menampilkan seluruh data hanya untuk mahasiswa yang bernama **Rudi Permana**, berikut query yang bisa digunakan:

```
MariaDB [universitas]> SELECT * FROM mahasiswa WHERE nama='Rudi Permana';
```

```
+-----+-----+-----+-----+-----+
| nim | nama | tempat_lahir | tanggal_lahir | jurusan | ipk |
+-----+-----+-----+-----+-----+
| 15021044 | Rudi Permana | Bandung | 1994-08-22 | Ilmu Komputer | NULL |
+-----+-----+-----+-----+-----+
1 row in set (0.16 sec)
```

Atau untuk menampilkan data dimana nimnya adalah **13012012**:

```
MariaDB [universitas]> SELECT * FROM mahasiswa WHERE nim='13012012';
```

```
+-----+-----+-----+-----+-----+
| nim | nama | tempat_lahir | tanggal_lahir | jurusan | ipk |
+-----+-----+-----+-----+-----+
| 13012012 | James Situmorang | Medan | 1995-04-02 | NULL | NULL |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Perintah kondisi dari query **SELECT** ini bisa menjadi cukup rumit sekaligus powerful. Misalkan saya ingin menampilkan data mahasiswa yang lahir di **Bandung** atau **Jakarta**. Querynya adalah sebagai berikut:

```
MariaDB [universitas]> SELECT * FROM mahasiswa WHERE tempat_lahir='Bandung' OR tempat_lahir='Jakarta';

+-----+-----+-----+-----+-----+
| nim | nama | tempat_lahir | tanggal_lahir | jurusan | ipk |
+-----+-----+-----+-----+-----+
| 15002032 | Rina Kumala Sari | Jakarta | 1997-06-28 | Akuntansi | 3.40 |
| 15003036 | Sari Citra Lestari | Jakarta | 1997-12-31 | Manajemen | NULL |
| 15021044 | Rudi Permana | Bandung | 1994-08-22 | Ilmu Komputer | NULL |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Bagaimana untuk mengurutkan data? Misalkan saya ingin menampilkan kolom **nama** dan **tanggal\_lahir** dan diurutkan berdasarkan abjad dari nama. Berikut querynya:

```
MariaDB [universitas]> SELECT nama,tanggal_lahir FROM mahasiswa ORDER BY nama ASC ;

+-----+-----+
| nama | tanggal_lahir |
+-----+-----+
| James Situmorang | 1995-04-02 |
| Riana Putria | 1996-11-23 |
| Rina Kumala Sari | 1997-06-28 |
| Rudi Permana | 1994-08-22 |
| Sari Citra Lestari | 1997-12-31 |
+-----+-----+
5 rows in set (0.07 sec)
```

Perintah **ORDER BY** digunakan untuk mengurutkan data berdasarkan kolom tertentu. Tambahan perintah **ASC** berarti saya ingin data ditampilkan berurutan dari A-Z (*ascending*).

Bagaimana jika dibalik misalnya dari Z ke A? Kita bisa menggunakan perintah **ORDER BY nama\_kolom DESC**. Perintah **DESC** merupakan singkatan dari *descending*. Berikut contohnya:

```
MariaDB [universitas]> SELECT tempat_lahir,nama FROM mahasiswa ORDER BY tempat_lahir DESC ;

+-----+-----+
| tempat_lahir | nama |
+-----+-----+
| Medan | James Situmorang |
| Jakarta | Rina Kumala Sari |
| Jakarta | Sari Citra Lestari |
| Bandung | Rudi Permana |
| NULL | Riana Putria |
+-----+-----+
```

```
+-----+-----+
5 rows in set (0.00 sec)
```

Sekarang kolom `tempat_lahir` disusun secara terbalik.

Untuk membatasi berapa banyak data yang tampil, kita bisa menggunakan perintah `LIMIT`. Misalkan saya ingin menampilkan data mahasiswa yang disusun berdasarkan abjad, tapi hanya 3 nama pertama saja. Berikut perintahnya:

```
MariaDB [universitas]> SELECT * FROM mahasiswa ORDER BY nama ASC LIMIT 3;
```

```
+-----+-----+-----+-----+-----+
|nim |nama |tempat_lahir|tanggal_lahir|jurusan |ipk |
+-----+-----+-----+-----+-----+
|13012012|James Situmorang|Medan |1995-04-02 |NULL |NULL |
|14005011|Riana Putria |NULL |1996-11-23 |NULL |3.10 |
|15002032|Rina Kumala Sari|Jakarta |1997-06-28 |Akuntansi |3.40 |
+-----+-----+-----+-----+-----+
3 rows in set (0.02 sec)
```

Query `SELECT` juga bisa digunakan untuk membuat proses pencarian. Perintah tambahannya adalah `LIKE`. Misalkan saya ingin mencari mahasiswa yang bernama “James Situmorang”, berikut perintahnya:

```
MariaDB [universitas]> SELECT * FROM mahasiswa WHERE nama LIKE 'James Situmorang';
```

```
+-----+-----+-----+-----+-----+
|nim |nama |tempat_lahir|tanggal_lahir|jurusan |ipk |
+-----+-----+-----+-----+-----+
|13012012|James Situmorang|Medan |1995-04-02 |NULL |NULL |
+-----+-----+-----+-----+-----+
1 row in set (0.06 sec)
```

Kondisi `LIKE` ini juga mendukung khusus atau *wildcard* untuk pencarian dengan pola. Karakter tersebut adalah ‘\_’ dan ‘%’:

- \_ (underscore): karakter ganti yang cocok untuk satu karakter apa saja.
- % (percent) : karakter ganti yang cocok untuk karakter apa saja dengan panjang karakter tidak terbatas, termasuk tidak ada karakter.

Misalkan untuk mencari nama mahasiswa yang namanya berawalan “R”, saya bisa menggunakan query berikut:

```
MariaDB [universitas]> SELECT * FROM mahasiswa WHERE nama LIKE 'R%';
```

```
+-----+-----+-----+-----+-----+
| nim | nama | tempat_lahir | tanggal_lahir | jurusan | ipk |
+-----+-----+-----+-----+-----+
| 14005011 | Riana Putria | NULL | 1996-11-23 | NULL | 3.10 |
| 15002032 | Rina Kumala Sari | Jakarta | 1997-06-28 | Akuntansi | 3.40 |
| 15021044 | Rudi Permana | Bandung | 1994-08-22 | Ilmu Komputer | NULL |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Atau bagaimana jika saya ingin mencari mahasiswa yang namanya mengandung huruf “sa”:

```
MariaDB [universitas]> SELECT * FROM mahasiswa WHERE nama LIKE '%sa%';
```

```
+-----+-----+-----+-----+-----+
| nim | nama | tempat_lahir | tanggal_lahir | jurusan | ipk |
+-----+-----+-----+-----+-----+
| 15002032 | Rina Kumala Sari | Jakarta | 1997-06-28 | Akuntansi | 3.40 |
| 15003036 | Sari Citra Lestari | Jakarta | 1997-12-31 | Manajemen | NULL |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

## Cara Mengubah Data dari Tabel MySQL

Setelah sebuah data diinput, kadang kita butuh mengubah atau meng-update data tersebut. Untuk keperluan ini, MySQL menyedian query **UPDATE**. Berikut format dasarnya:

```
UPDATE nama_tabel SET nama_kolom = data_baru WHERE kondisi
```

Misalkan saya ingin mengupate tempat lahir “Riana Putria” menjadi “Padang”. Querynya adalah sebagai berikut:

```
MariaDB [universitas]> SELECT * FROM mahasiswa WHERE nama='Riana Putria';
```

```
+-----+-----+-----+-----+-----+
| nim | nama | tempat_lahir | tanggal_lahir | jurusan | ipk |
+-----+-----+-----+-----+-----+
| 14005011 | Riana Putria | NULL | 1996-11-23 | NULL | 3.10 |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

```
MariaDB [universitas]> UPDATE mahasiswa SET tempat_lahir='Padang'  
WHERE nama='Riana Putria';
```

```
Query OK, 1 row affected (0.08 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [universitas]> SELECT * FROM mahasiswa WHERE nama='Riana Putria';

+-----+-----+-----+-----+-----+
| nim | nama | tempat_lahir | tanggal_lahir | jurusan | ipk |
+-----+-----+-----+-----+-----+
| 14005011 | Riana Putria | Padang | 1996-11-23 | NULL | 3.10 |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Sebelum saya menjalankan query **UPDATE**, saya ingin memastikan data asal terlebih dahulu, dimana kolom **tempat\_lahir** dari “Riana Putria” masih kosong (NULL). Setelah query **UPDATE**, terlihat kolom **tempat\_lahir** telah di update menjadi “Padang”.

Yang perlu diperhatikan untuk query **UPDATE** adalah bagian kondisi. Jika kita lupa menyeretkan bagian ini, Perintah **UPDATE** akan mengubah seluruh data yang ada di tabel.

Misalkan saya menjalankan perintah berikut:

```
UPDATE mahasiswa SET tempat_lahir='Padang';
```

Maka kolom **tempat\_lahir** untuk semua mahasiswa akan berubah menjadi “Padang”.

## Cara Menghapus Data dari Tabel MySQL

Query terakhir yang akan kita pelajari adalah bagaimana cara menghapus sebuah data. Untuk keperluan ini kita akan menggunakan perintah **DELETE**. Berikut format dasarnya:

```
DELETE FROM nama_tabel WHERE kondisi
```

Misalkan saya ingin menghapus data mahasiswa yang bertempat lahir di Jakarta. Quernya adalah:

```
MariaDB [universitas]> SELECT * FROM mahasiswa;
```

```
+-----+-----+-----+-----+-----+
| nim | nama | tempat_lahir | tanggal_lahir | jurusan | ipk |
+-----+-----+-----+-----+-----+
| 13012012 | James Situmorang | Medan | 1995-04-02 | NULL | NULL |
| 14005011 | Riana Putria | Padang | 1996-11-23 | NULL | 3.10 |
| 15002032 | Rina Kumala Sari | Jakarta | 1997-06-28 | Akuntansi | 3.40 |
| 15003036 | Sari Citra Lestari | Jakarta | 1997-12-31 | Manajemen | NULL |
+-----+-----+-----+-----+-----+
```

```
| 15021044|Rudi Permana      | Bandung      | 1994-08-22  | Ilmu Komputer|NULL|
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

```
MariaDB [universitas]> DELETE FROM mahasiswa WHERE tempat_lahir="Jakarta";
Query OK, 2 rows affected (0.05 sec)
```

```
MariaDB [universitas]> SELECT * FROM mahasiswa;
+-----+-----+-----+-----+
| nim  | nama      | tempat_lahir|tanggal_lahir|jurusan      | ipk |
+-----+-----+-----+-----+
| 13012012|James Situmorang|Medan      |1995-04-02  |NULL        |NULL|
| 14005011|Riana Putria    |Padang      |1996-11-23  |NULL        |3.10|
| 15021044|Rudi Permana    |Bandung      |1994-08-22  |Ilmu Komputer|NULL|
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Terlihat data dari **Rina Kumala Sari** dan **Sari Citra Lestari** sudah dihapus hasil dari perintah diatas. Ini karena kedua mahasiswa lahir di **Jakarta**, dan cocok dengan kondisi **WHERE**.

Bagian **WHERE** ini bisa diisi dengan berbagai kriteria. Misalkan saya ingin menghapus data mahasiswa yang namanya diawali dengan huruf “R”, berikut perintahnya

```
MariaDB [universitas]> SELECT * FROM mahasiswa;
+-----+-----+-----+-----+
| nim  | nama      | tempat_lahir|tanggal_lahir|jurusan      | ipk |
+-----+-----+-----+-----+
| 13012012|James Situmorang|Medan      |1995-04-02  |NULL        |NULL|
| 14005011|Riana Putria    |Padang      |1996-11-23  |NULL        |3.10|
| 15021044|Rudi Permana    |Bandung      |1994-08-22  |Ilmu Komputer|NULL|
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

```
MariaDB [universitas]> DELETE FROM mahasiswa WHERE nama LIKE 'R%';
Query OK, 2 rows affected (0.05 sec)
```

```
MariaDB [universitas]> SELECT * FROM mahasiswa;
+-----+-----+-----+-----+
| nim  | nama      | tempat_lahir|tanggal_lahir|jurusan|ipk |
+-----+-----+-----+-----+
| 13012012|James Situmorang|Medan      |1995-04-02  |NULL    |NULL|
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Hasil dari query ini, mahasiswa dengan nama awal “R” akan terhapus.

Sama seperti query **UPDATE**, jika kita lupa membuat kondisi where, maka semua data akan ikut terhapus, seperti perintah berikut:

```
MariaDB [universitas]> DELETE FROM mahasiswa;  
Query OK, 1 row affected (0.05 sec)
```

```
MariaDB [universitas]> SELECT * FROM mahasiswa;  
Empty set (0.00 sec)
```

Perintah **DELETE FROM** mahasiswa akan menghapus seluruh isi data tabel **mahasiswa**;



Walaupun sudah 20 halaman lebih, query MySQL yang kita bahas disini masih termasuk materi dasar. Sebagai contoh, saya belum membahas cara menampilkan data dari 2 tabel yang saling berhubungan.

Sekali lagi, materi tentang SQL Query sangat luas. Silahkan anda cari buku atau referensi lain yang khusus membahas MySQL agar bisa mempelajarinya dengan lebih detail lagi.

## 23.7 Mengenal MySQL API (mysql, mysqli, dan PDO)

Di dalam pemrograman, terdapat istilah **API** (*Application Programming Interface*). Secara sederhana, **API** adalah kumpulan perintah, kode program, atau fungsi yang berguna sebagai sarana komunikasi dari sebuah perangkat (bisa berupa hardware maupun software).

Analoginya seperti *remote TV*. Remote digunakan untuk mengganti channel TV serta pengaturan lain seperti *brightness*, *contras*, dan volume suara. Dengan menekan tombol di remote, kita sebenarnya sedang ‘berkomunikasi’ dengan TV. Disini remote merupakan API dari TV.

Untuk berkomunikasi dengan database MySQL dari PHP, kita juga memerlukan **MySQL API**. Terdapat 3 jenis MySQL API yang bisa digunakan: **mysql extension**, **mysqli extension**, dan **PDO extension**. Mari kita bahas perbedaan dari ketiga MySQL API ini.



Kata ‘*extension*’ disini menunjukkan bahwa **MySQL API** bukanlah bagian dari PHP, tetapi merupakan tambahan (*extension*) ke dalam PHP. Selain MySQL API, PHP memiliki banyak *extension* lain.

Umumnya *extension* ini harus diaktifkan terlebih dahulu dari **php.ini**. Tapi yang cukup populer seperti MySQL, sudah langsung aktif. File yang diperlukan oleh extension berada di folder **xampp\php\ext**. Sebagai contoh, **mysqli extension** berasal dari file **xampp\php\ext\php\_mysqli.dll**.

### Mysql Extension API

**Mysql extension** merupakan *MySQL API* yang pertama kali muncul di PHP, tepatnya sejak PHP 2.0. Ketika saya pertama kali mempelajari PHP (sekitar tahun 2008), inilah satu-satunya extension yang tersedia. Mysql extension dijalankan menggunakan fungsi-fungsi seperti **mysql\_connect()**, **mysql\_query()**, dan **mysql\_fetch\_array()**.

Akan tetapi karena sudah cukup “tua”, **mysql extension** saat ini sudah memasuki masa pensiun. Sejak PHP versi 5.5, *mysql extension* sudah berstatus **deprecated**, yang artinya disarankan untuk tidak digunakan lagi dan mungkin akan dihapus. Dan itulah yang terjadi. **Pada PHP 7.0, mysql extension sepenuhnya telah dihapus dari PHP** (tidak bisa lagi digunakan).

Akan tetapi karena diluar sana masih banyak script, tutorial dan buku-buku PHP yang belum update, saya masih sering menjumpai pertanyaan seputar *mysql extension*. Kebanyakan bertanya kenapa tidak terhubung ke MySQL. Ini terjadi karena sudah menggunakan **mysqli extension**.

Untuk anda yang baru pertama kali belajar PHP, sebaiknya langsung saja menggunakan **mysqli extension**.

## Mysqli Extension API

**Mysqli** merupakan singkatan dari **mysql improved**. **Improved** berarti peningkatan dari *mysql extension* sebelumnya. Salah satu alasan kenapa PHP membuat **mysqli** dari awal dan tidak mengembangkan *mysql extension* saja adalah karena banyaknya fitur-fitur baru di MySQL versi 4 keatas. Fitur-fitur ini mengharuskan perubahan besar sehingga tidak cocok dimasukkan ke dalam *mysql extension*.

**Mysqli extension** diperkenalkan sejak PHP 5.0. **Syntax** dan fungsi-fungsi yang digunakan tidak jauh berbeda dengan *mysql extension*. Umumnya kita hanya perlu menambahkan huruf “i” di nama fungsi dan membalik urutan argumen fungsi. Contohnya seperti **mysql\_connect()** menjadi **mysqli\_connect()**.

Jika anda sebelumnya sudah pernah mempelajari *mysql extension*, tidak akan butuh waktu lama untuk beralih ke *mysqli extension*.

**Mysqli extension** juga hadir dengan 2 rasa: **prosedural** (dengan fungsi-fungsi) dan **pemrograman objek** (*Object Oriented Programming* – OOP). Perbedaan keduanya sebenarnya tidak banyak, namun karena di buku **PHP Uncover** ini saya tidak membahas tentang OOP, kita akan fokus membahas *mysqli extension* yang dijalankan melalui fungsi-fungsi (prosedural).



**Mysqli** rasa OOP lebih pas dipelajari jika anda sudah paham tentang konsep pemrograman berbasis objek. Untuk programmer pemula, sebaiknya jangan terburu-buru lompat ke OOP. Silahkan kuasai dulu cara penggunaan PHP berbasis prosedural (silahkan buat 1 atau 2 proyek), lalu baru beralih ke pemrograman berbasis objek.

Untuk aplikasi yang relatif sederhana seperti tugas kuliah, skripsi, atau sistem informasi sederhana, akan lebih cepat jika dibuat dengan pemrograman prosedural. OOP lebih cocok untuk aplikasi besar yang perlu perubahan dan penambahan fitur berkala.

## PDO Extension API

**PDO** merupakan singkatan dari **(PHP Data Objects)**. **PDO** adalah sebuah database API universal yang dirancang bukan hanya untuk MySQL saja, tetapi juga untuk aplikasi database lain seperti **Oracle** dan **PostgreSQL**.

PDO hadir sejak PHP 5.1 yang dibuat agar memudahkan programmer untuk migrasi dari suatu aplikasi database ke database lain. Misalkan saat ini saya membuat sebuah website dengan database **MySQL** menggunakan *PDO Extension*. Jika nanti ingin migrasi ke **PostgreSQL**, saya tidak perlu mengubah seluruh kode program, tapi cukup beberapa baris saja.

Walaupun terkesan praktis, dalam prakteknya tidak semudah itu. Bisa jadi kita menggunakan fitur atau perintah query yang hanya ada di MySQL dan tidak tersedia di PostgreSQL. Belum lagi proses memindahkan seluruh data ke database lain sangat beresiko. Besar kemungkinan ada data yang terpotong, terhapus, atau tidak bisa diinput ke database yang baru. Perubahan database setelah situs ‘online’ relatif jarang dilakukan.

**PDO** hanya bisa ditulis dengan menggunakan pemrograman objek. Oleh karena itu dalam bab ini saya juga tidak akan membahas PDO. Pembahasan detail tentang PDO akan menjadi jatah buku PHP DuniaIlkom berikutnya.



Jika anda butuh tutorial singkat mengenai PDO (dan juga mysqli object) bisa membacanya di situs duniaIlkom: [Tutorial Belajar PHP Lanjutan: Cara Membuat Koneksi PHP ke Database MySQL<sup>4</sup>](#). Khusus untuk PDO dimulai dari Part 19-23.

Tabel berikut membandingkan fitur dari ketiga **MySQL API**. Tabel ini yang saya ambil dari PHP Manual:

	mysqli	PDO	mysql
PHP version introduced	5.0	5.1	2.0
Included with PHP 5.x	Yes	Yes	Yes
Included with PHP 7.x	Yes	Yes	No
Development status	Active	Active	Maintenance only in 5.x; removed in 7.x
Lifecycle	Active	Active	Deprecated in 5.x; removed in 7.x
Recommended for new projects	Yes	Yes	No
OOP Interface	Yes	Yes	No
Procedural Interface	Yes	No	Yes
API supports non-blocking, asynchronous queries with mysqlnd	Yes	No	No
Persistent Connections	Yes	Yes	Yes
API supports Charsets	Yes	Yes	Yes
API supports server-side Prepared Statements	Yes	Yes	No
API supports client-side Prepared Statements	No	Yes	No
API supports Stored Procedures	Yes	Yes	No
API supports Multiple Statements	Yes	Most	No
API supports Transactions	Yes	Yes	No
Transactions can be controlled with SQL	Yes	Yes	Yes
Supports all MySQL 5.1+ functionality	Yes	Most	No

Terlihat bahwa mysql extension sudah sepenuhnya dihapus pada PHP 7.. Juga perhatikan nilai “**Most**” untuk PDO: *Supports all MySQL 5.1+ functionality*. Jika anda fokus hanya menggunakan database MySQL, **mysqli extension** adalah pilihan paling pas dan direkomendasikan (jika

<sup>4</sup><http://www.duniaIlkom.com/tutorial-php-cara-membuat-koneksi-php-ke-database-mysql/>

dibandingkan PDO). Karena *mysqli extension* mendukung semua fitur-fitur dari MySQL 5.1 keatas. Inilah yang akan kita bahas berikutnya.

## 23.8 Langkah-langkah Koneksi PHP-MySQL

Dari ketiga MySQL API yang disediakan PHP, dalam bab ini saya memilih menggunakan **mysqli extension**, tepatnya *prosedural mysqli extension*. Disini kita akan belajar cara berkomunikasi dengan MySQL melalui fungsi-fungsi **mysqli**.

Jika nantinya anda sudah paham prosedural *mysqli extension*, tidak terlalu sulit untuk beralih ke *object mysqli extension* maupun ke PDO, karena fungsi dasar MySQL API di PHP relatif mirip.

Untuk berkomunikasi dengan MySQL dari PHP idealnya membutuhkan 5 langkah:

1. Buat koneksi dengan MySQL.
2. Jalankan query (SQL), hasilnya disimpan ke memory web server.
3. Proses hasil query dengan PHP (jika ada).
4. Kosongkan memory web server (opsional).
5. Tutup koneksi dengan MySQL (opsional).

**Langkah pertama** adalah membuat koneksi dengan MySQL Server. Disini kita akan menginput alamat server, nama *username* serta *password* user MySQL. Jika menggunakan **mysqli**, kita juga bisa langsung memilih nama dababase yang akan digunakan.

Setelah terambung dengan MySQL Server, **langkah kedua** adalah menjalankan query. Apakah itu membuat tabel (CREATE), menampilkan isi tabel (SELECT), mengubah data (UPDATE) atau menghapus data (DELETE). Query yang kita ketik selanjutnya dikirim ke MySQL Server untuk diproses. Hasil dari query kemudian dikembalikan ke PHP (tepatnya dikembalikan ke web server).

Dalam tahap ini, hasil query masih berada di dalam memory web server. Agar bisa diakses, kita harus memprosesnya dari PHP. Apakah hasil query ini akan ditampilkan, disimpan, atau diproses lebih jauh tergantung kode program yang dibuat. Inilah **langkah ketiga** dari list diatas.

**Langkah keempat** adalah menghapus data hasil query dari memory web server. Tentunya ini dilakukan setelah data selesai di proses. Ruang memory yang tadinya digunakan bisa kembali kosong untuk keperluan lain. Jika tidak dihapus, memory web server bisa penuh karena data dari MySQL yang saling bertumpuk.

Akan tetapi langkah ini sebenarnya bersifat opsional. PHP memiliki fitur “*garbage collection*”. Fitur ini otomatis menghapus data query dari memory jika tidak lagi dibutuhkan. Namun tidak ada salahnya jika kita melakukan perintah ini secara langsung.

**Langkah kelima** adalah memutus sambungan ke MySQL. Sama seperti langkah ke-4, PHP otomatis memutus komunikasi dengan MySQL tepat setelah halaman selesai diproses. Jadi perintah ini juga boleh tidak ditulis (opsional). Walaupun begitu, merupakan kebiasaan programming yang baik jika kita memutusnya secara manual (*good programming practice*).

Langkah pertama dan kelima umumnya hanya cukup dilakukan sekali, yakni di awal dan diakhir kode program. Untuk langkah ke-2, 3 dan 4, bisa dijalankan berulang-ulang selama diperlukan.

Secara default, komunikasi antara PHP dengan MySQL dijalankan per-halaman. Ketika 1 halaman PHP selesai diproses, koneksi langsung diputus. Saat kita buka halaman baru, koneksi kembali dibuka dan akan diputus setelah halaman tersebut selesai diproses, begitu seterusnya. Untuk website yang ramai, proses putus-nyambung ini bisa berlangsung ribuan kali pada saat yang bersamaan.

Baik, mari kita masuk ke kode program PHP.

## 23.9 Function mysqli\_connect()

Fungsi **mysqli** pertama yang akan kita bahas adalah **mysqli\_connect()**. Sesuai dengan namanya, fungsi ini digunakan untuk membuat koneksi dari PHP ke MySQL Server. Fungsi **mysqli\_connect()** membutuhkan beberapa argumen dengan format dasar sebagai berikut:

```
$koneksi = mysqli_connect("alamat_host", "nama_user", "password_user",
"nama_database")
```

Variabel `$koneksi` berfungsi untuk menampung hasil dari fungsi **mysqli\_connect()**. Variabel yang dikenal dengan istilah “*handle*” ini akan digunakan oleh berbagai fungsi **mysqli** lainnya. Anda bisa menggunakan nama variabel apapun, yang cukup umum adalah `$db`, `$connection`, `$conn`, `$mysql` atau `$link`.

Argumen pertama dari fungsi **mysqli\_connect()** adalah “`alamat_host`”. Alamat host merupakan alamat komputer di mana MySQL Server berada. Alamat ini bisa berupa *IP address* seperti 192.168.0.3 atau berupa domain seperti [duniaillkom.com](http://duniaillkom.com). Karena kita menjalankan MySQL Server di komputer local (menggunakan XAMPP), nilainya bisa diisi dengan “`localhost`” atau `127.0.0.1`.

Argumen kedua adalah “`nama_user`”. Ini adalah nama user MySQL yang akan digunakan untuk login ke MySQL Server. Dalam pembahasan tentang SQL sebelumnya, kita sudah mengetahui bahwa terdapat user “`root`”. User inilah yang akan kita gunakan sepanjang bab ini.

Argumen ketiga diisi dengan “`password_user`”. Password user ini bersesuaian dengan nama user dari argumen kedua. Di dalam XAMPP, user “`root`” secara default tidak memiliki password.

Argumen keempat adalah `nama_database` yang akan kita gunakan. Nama database ini harus sudah tersedia di dalam MySQL. Dalam bagian tentang query MySQL, saya membuat tabel **mahasiswa** yang berada di dalam database **universitas**. Jadi argumen keempat ini bisa diisi dengan “`universitas`”.

Anda mungkin bertanya, jika kita harus menentukan database di awal koneksi, bagaimana jika nanti ingin mengubah nama database?

Argumen keempat ini sebenarnya optional. Kita tidak harus menulis nama database di awal. Namun mengubah nama database sepanjang kode program tidak umum dilakukan. Karena seharusnya kita hanya menggunakan satu database sepanjang penggeraan aplikasi.

Walaupun nanti jumlah tabelnya berjumlah ratusan atau ribuan, selama berada dalam 1 proyek atau 1 website, semua tabel ini seharusnya tetap berada di dalam 1 database.

Berikut contoh kode program untuk membuat koneksi dari PHP ke MySQL:

```
1 <?php
2   $koneksi = mysqli_connect("localhost", "root", "", "universitas");
3 ?>
4 <!DOCTYPE html>
5 <html>
6 <head>
7   <meta charset="UTF-8">
8   <title>Belajar PHP</title>
9 </head>
10 <body>
11   <h1>Koneksi PHP dengan MySQL</h1>
12 </body>
13 </html>
```

Disini saya menempatkan fungsi `mysqli_connect()` ditaris paling awal. Tapi ini sebenarnya tidak wajib (berbeda dengan fungsi `header()` atau `setcookie()` yang memang harus di posisi pertama kode program). Jika koneksi dengan MySQL gagal, saya bisa langsung menghentikan proses sebelum kode HTML dijalankan.

Jika anda menjalankan kode program diatas dan tidak menemukan error, maka tidak ada masalah. Namun jika terdapat error, periksa kembali hal-hal berikut:

- Pastikan MySQL Server sudah berjalan (cek dari XAMPP Control Panel).
- Periksa kembali penulisan seluruh argumen fungsi `mysqli_connect()`, pastikan tidak ada salah ketik.
- Pastikan database yang ditulis sudah tersedia di dalam MySQL.

Sebagai contoh, saya akan mengubah fungsi `mysqli_connect()` menjadi seperti berikut:

```
1 <?php
2   $koneksi = mysqli_connect("localhost", "root", "rahasia", "universitas");
3 ?>
4 <!DOCTYPE html>
5 <html>
6 <head>
7   <meta charset="UTF-8">
8   <title>Belajar PHP</title>
9 </head>
10 <body>
11   <h1>Koneksi PHP dengan MySQL</h1>
12 </body>
13 </html>
```



Gambar: Error pada saat koneksi PHP dengan MySQL

Dapatkah anda menemukan dimana letak kesalahannya?

Disini saya menginput “**rahasia**” sebagai argumen ketiga fungsi **mysqli\_connect()**. Artinya, “**rahasia**” saya input sebagai password user “root”. Seharusnya user root tidak memiliki password. Oleh karena itulah MySQL server menolak koneksi dan menampilkan pesan **error: Access denied for user ‘root’@‘localhost’ (using password: YES)**.

Pesan error seperti ini cukup sering saya temui. Jika anda juga menemukan kasus seperti ini, kemungkinan besar terdapat kesalahan dalam menulis username dan/atau password user MySQL.

Agar lebih rapi, banyak programmer yang memisahkan setiap argumen menjadi variabel tersendiri, seperti contoh berikut:

```
1 <?php
2 // buat koneksi dengan database mysql
3 $dbhost = "localhost";
4 $dbuser = "root";
5 $dbpass = "";
6 $dbname = "universitas";
7 $koneksi = mysqli_connect($dbhost,$dbuser,$dbpass,$dbname);
8 ?>
```

Kode diatas identik dengan kode program kita sebelumnya. Hanya saja setiap argumen fungsi **mysqli\_connect()** disimpan kedalam variabel.

## 23.10 Function **mysqli\_connect\_errno()** dan **mysqli\_connect\_error()**

Kedua fungsi ini digunakan untuk menampilkan pesan error apabila koneksi antara PHP dengan MySQL mengalami masalah. Fungsi **mysqli\_connect\_errno()** akan menampilkan empat digit angka kode error MySQL (**errno** merupakan singkatan dari *error number*). Sedangkan fungsi **mysqli\_connect\_error()** akan menampilkan pesan errornya.

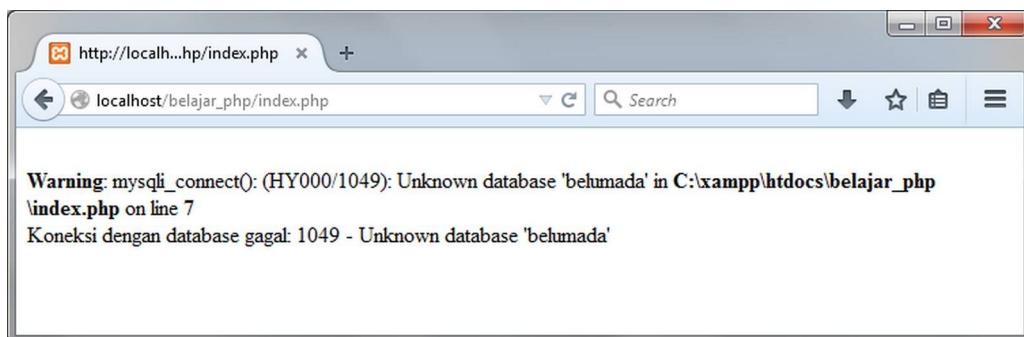
Sebenarnya, pesan error dari MySQL tetap ditampilkan oleh PHP melalui “**Warning**”. Misalkan jika database yang dipilih dari fungsi **mysqli\_connect()** ternyata tidak ada, PHP akan mengeluarkan error:

```
"Warning: mysqli_connect(): (HY000/1049): Unknown database 'belumada'  
in C:\xampp\htdocs\belajar_php\index.php on line 7"
```

Tapi bagaimana jika fitur debugging PHP dinonaktifkan? Pesan warning tersebut tidak akan tampil. Ini umum terjadi di web hosting, dimana atas alasan keamanan pihak web hosting mematikan fitur `error_reporting` dari PHP. Untuk kasus seperti inilah fungsi `mysqli_connect_errno()` dan `mysqli_connect_error()` bisa berguna.

Kedua fungsi ini baru bermanfaat ketika terjadi masalah dengan koneksi antara PHP dengan MySQL Server. Jika koneksi baik-baik saja, kita tidak perlu memanggilnya. Oleh karena itu saya bisa membuat kondisi untuk memeriksa apakah terjadi kesalahan atau tidak. Berikut contoh penggunaannya:

```
1  <?php  
2  // buat koneksi dengan database mysql  
3  $dbhost = "localhost";  
4  $dbuser = "root";  
5  $dbpass = "";  
6  $dbname = "belumada";  
7  $koneksi = mysqli_connect($dbhost,$dbuser,$dbpass,$dbname);  
8  
9  //periksa koneksi, tampilkan pesan kesalahan jika gagal  
10 if(!$koneksi){  
11     die ("Koneksi dengan database gagal: ".mysqli_connect_errno().  
12         " - ".mysqli_connect_error());  
13 }  
14 ?>  
15 <!DOCTYPE html>  
16 <html>  
17 <head>  
18     <meta charset="UTF-8">  
19     <title>Belajar PHP</title>  
20 </head>  
21 <body>  
22     <h1>Koneksi PHP dengan MySQL</h1>  
23 </body>  
24 </html>
```



Gambar: Pesan kesalahan ditampilkan ketika koneksi dengan MySQL tidak berhasil

Fungsi `mysqli_connect_errno()` dan `mysqli_connect_error()` tidak membutuhkan argumen. Sebelum memanggil kedua fungsi ini, saya memeriksa terlebih dahulu isi dari variabel `$koneksi`. Jika koneksi berhasil dilakukan, variabel ini akan berisi sebuah objek `mysqli`. Jika koneksi gagal, variabel `$koneksi` akan berisi nilai boolean `FALSE`.

Kondisi inilah yang bisa dimanfaatkan untuk menentukan apakah perlu ditampilkan pesan kesalahan atau tidak. Jika koneksi sukses, kita bisa lompat ke kode program selanjutnya. Namun jika terjadi masalah, baru tampilkan pesan kesalahan. Pengecekan kondisi ini di proses oleh kode program berikut:

```
1 if(!$koneksi){  
2     die ("Koneksi dengan database gagal: ".mysqli_connect_errno().  
3           " - ".mysqli_connect_error());  
4 }
```

Kondisi IF baru akan dijalankan ketika variabel `$koneksi` berisi `FALSE` (perhatikan operator negasi “`!`” di dalam kondisi IF). Jika ini yang terjadi, fungsi `die()` akan mengentikan kode program dan menampilkan kode dan pesan kesalahan MySQL. Dalam contoh diatas, kode errornya adalah **1049**, dan pesan error: **Unknown database ‘belumada’**. Yang artinya MySQL tidak menemukan database “belumada”.

Perlu saya tambahkan, walaupun pesan error ini akan sangat membantu selama perancangan kode program (selama proses web development), untuk situs “live” sebaiknya anda tidak melakukan hal ini.

Pesan error MySQL bisa dipelajari untuk mencari tahu struktur database yang digunakan oleh sebuah website. Ini bisa menjadi celah untuk membobol web kita. Oleh karena itu, jika seluruh kode program sudah selesai dan website akan launcing, sebaiknya hapus fungsi `mysqli_connect_errno()` dan `mysqli_connect_error()` menjadi seperti berikut ini:

```
1 <?php
2 // buat koneksi dengan database mysql
3 $koneksi = mysqli_connect("localhost","root","","universitas");
4
5 //periksa koneksi, tampilkan pesan kesalahan jika gagal
6 if(!$koneksi){
7     die ("Koneksi dengan database gagal");
8 }
9 ?>
10 <!DOCTYPE html>
11 <html>
12 <head>
13     <meta charset="UTF-8">
14     <title>Belajar PHP</title>
15 </head>
16 <body>
17     <h1>Koneksi PHP dengan MySQL</h1>
18 </body>
19 </html>
```

Dengan kode program seperti ini, jika terjadi gagal koneksi, pengujung web hanya melihat pesan error “Koneksi dengan database gagal”. Orang yang memiliki niat jahat tidak bisa mendapat informasi tambahan lain (misalnya nama database yang kita gunakan).

Akan tatapi selama proses web development, fungsi `mysqli_connect_errno()` dan `mysqli_connect_error()` akan sangat berguna. Tanpa informasi ini kita bisa menghabiskan waktu berjam-jam mencari letak kesalahan kode program yang ternyata disebabkan salah mengetik nama user MySQL.

## 23.11 Function `mysqli_close()`

Fungsi `mysqli_close()` merupakan kebalikan dari fungsi `mysqli_connect()`. Disini `mysqli_close()` akan menutup koneksi antara PHP dengan MySQL dan membebaskan memory yang digunakan untuk proses koneksi tersebut.

Fungsi ini biasanya diletakkan dibaris paling akhir atau dimana kita tidak butuh lagi mengakses database. Fungsi `mysqli_close()` sebenarnya bersifat opsional dan boleh tidak ditulis. PHP otomatis memutus koneksi (dan membebaskan memory) begitu halaman PHP selesai diproses. Tapi menulisnya langsung dianggap sebagai kebiasaan programming yang baik (*good programming practice*).

Fungsi `mysqli_close()` membutuhkan 1 argumen, yakni variabel handle yang digunakan ketika memanggil fungsi `mysqli_connect()`. Menggunakan contoh kita sebelumnya, variabel ini adalah `$koneksi`.

Berikut contoh penggunaan fungsi `mysqli_close()`:

```
1 <?php
2 // buat koneksi dengan database mysql
3 $koneksi = mysqli_connect("localhost","root","","universitas");
4
5 //periksa koneksi, tampilkan pesan kesalahan jika gagal
6 if(!$koneksi){
7     die ("Koneksi dengan database gagal");
8 }
9 ?>
10 <!DOCTYPE html>
11 <html>
12 <head>
13     <meta charset="UTF-8">
14     <title>Belajar PHP</title>
15 </head>
16 <body>
17     <h1>Koneksi PHP dengan MySQL</h1>
18 </body>
19 </html>
20 <?php
21 // tutup koneksi dengan database mysql
22 mysqli_close($koneksi);
23 ?>
```

Disini saya menempatkan fungsi `mysqli_close($koneksi)` di baris paling akhir setelah tag penutup `</html>`. Anda bisa menempatkan fungsi ini dimana saja jika memang tidak butuh lagi akses ke database MySQL.

## 23.12 Function `mysqli_query()`

Setelah koneksi dengan MySQL berhasil dibuat, langkah berikutnya adalah menjalankan perintah query (perintah SQL). Inilah kegunaan dari fungsi `mysqli_query()`. Fungsi ini membutuhkan 2 buah argumen. Argumen pertama berupa variabel *handle* dari pemanggilan fungsi `mysqli_connect()`, sedangkan argumen kedua berisi perintah query dalam bentuk string.

Berikut contoh penggunaan fungsi `mysqli_query()`:

```

1  <?php
2  // buat koneksi dengan database mysql
3  $koneksi = mysqli_connect("localhost","root","","universitas");
4
5  //periksa koneksi, tampilkan pesan kesalahan jika gagal
6  if(!$koneksi){
7      die ("Koneksi dengan database gagal: ".mysqli_connect_errno().
8          " - ".mysqli_connect_error());
9  }
10
11 //jalankan query
12 $hasil_query = mysqli_query($koneksi, "SELECT * FROM mahasiswa");
13
14 // tutup koneksi dengan database mysql
15 mysqli_close($koneksi);
16 ?>

```

Dalam kode program kali ini saya tidak menyertakan bagian HTML agar menghemat tempat. Perhatikan penulisan fungsi `mysqli_query()`:

```
$hasil_query = mysqli_query($koneksi, "SELECT * FROM mahasiswa");
```

Variabel `$hasil_query` digunakan untuk menampung hasil pemanggilan fungsi `mysqli_query()`. Kebanyakan buku-buku dan tutorial PHP biasanya menggunakan nama variabel `$result`. Di sini saya ingin menunjukkan bahwa nama variabel ini bisa apa saja, seperti `$hasil_query`.

Variabel `$hasil_query` berisi data khusus yang tidak bisa diakses langsung. PHP menyebutnya sebagai tipe data **resources**.

Untuk menampilkan data yang terdapat di variabel `$hasil_query`, kita butuh fungsi MySQL lainnya seperti `mysqli_fetch_row()`, `mysqli_fetch_assoc()` atau `mysqli_fetch_array()`. Ketika fungsi ini akan kita bahas sesaat lagi.

Salah satu sumber error yang paling sering terjadi adalah di fungsi `mysqli_query()` ini. Lebih spesifik lagi dari query yang kita jalankan.

Saya menerima banyak pertanyaan tentang error PHP yang sebenarnya disebabkan salah menulis query MySQL. Sebagai contoh, silahkan anda ubah baris kode program `mysqli_query()` diatas menjadi sebagai berikut:

```
$hasil_query = mysqli_query($koneksi, "SELEC * FRM mahasiswa");
```

Anda bisa tebak letak salahnya? Yup disini query MySQL yang saya jalankan tidak valid. `SELECT` ditulis menjadi `SELEC` dan `FROM` menjadi `FRM`. Tapi, coba anda jalankan. PHP tidak akan menampilkan pesan error apapun!

Pesan error baru tampil ketika kita mulai memproses data. Tentu saja tidak ada data yang dihasilkan karena querynya sendiri juga sudah salah. Yang membuat pusing, pesan error biasanya tampil bukan di baris `mysqli_query()`, tapi di baris lain.

Contoh kasus lain, biasanya di dalam query MySQL terdapat variabel, seperti contoh berikut:

```
mysqli_query($koneksi, "SELECT * FROM mahasiswa WHERE nama='$nama'");
```

Variabel \$nama ini bisa berasal dari form atau sumber lain. Apakah anda yakin \$nama ini sudah berisi ‘sesuatu’?

Untuk meminimalisir masalah-masalah seperti ini, silahkan ikuti panduan berikut:

Pertama, pisahkan penulisan query ke dalam sebuah variabel. Variabel inilah yang akan menjadi argumen kedua fungsi **mysqli\_query()**, seperti contoh berikut:

```
$query = "SELECT * FROM mahasiswa";
$hasil_query = mysqli_query($koneksi, $query);
```

Kedua, periksa query dengan menjalankan perintah echo ke variabel \$query:

```
1 $query = "SELECT * FROM mahasiswa";
2 echo $query; // SELECT * FROM mahasiswa
3
4 $hasil_query = mysqli_query($koneksi, $query);
```

Ketiga, copy hasil dari echo() ini dan jalankan di cmd MySQL atau dari **Phpmyadmin**. Langkah ketiga ini sangat penting. Pastikan query berjalan seperti yang kita harapkan.

Sebagai contoh, apakah hasil dari perintah “SELECT \* FROM mahasiswa” akan menampilkan seluruh data dari tabel mahasiswa? Jika tidak, tentu penulisan query harus diperbaiki. Apabila sudah sesuai, silahkan hapus perintah echo dan lanjut ke kode program berikutnya.

Dengan memastikan setiap query berjalan dengan semestinya, kita bisa mengantisipasi error yang terjadi. Langkah ini sangat penting apalagi untuk query yang cukup kompleks seperti contoh berikut:

```
1 $nama = "James Situmorang";
2 $query = "SELECT * FROM mahasiswa WHERE nama=$nama";
3
4 echo $query;
5 // SELECT * FROM mahasiswa WHERE nama=James Situmorang
6
7 $hasil_query = mysqli_query($koneksi, $query);
```

Apakah anda melihat ada sesuatu yang salah? Sepertinya tidak. Tapi mari kita jalankan di cmd MySQL. Berikut hasilnya:

```
MariaDB [universitas]> SELECT * FROM mahasiswa WHERE nama=James Situmorang;
```

```
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual
that corresponds to your MariaDB server version for the right syntax to use
near 'Situmorang' at line 1
```

Ini terjadi karena saya lupa menambahkan tanda kutip di bagian `nama=James Situmorang`. Seharusnya ini ditulis menjadi `nama='James Situmorang'` :

```
1 //jalankan query
2 $nama = "James Situmorang";
3 $query = "SELECT * FROM mahasiswa WHERE nama='$nama'";
4
5 echo $query;
6 // SELECT * FROM mahasiswa WHERE nama='James Situmorang'
7
8 $hasil_query = mysqli_query($koneksi, $query);
```

Kesalahan seperti inilah yang sering “tidak terduga”. Jika boleh, saya **mewajibkan** anda untuk menjalankan setiap perintah query di MySQL dulu, baru kemudian pindahkan ke PHP.

Setelah memastikan query berjalan lancar, anda bisa menghapus baris `echo $query` dan lanjut ke kode program berikutnya.

Fungsi `mysqli_query()` hanya digunakan untuk menjalankan query MySQL. Bagaimana cara menampilkan hasilnya? kita butuh bantuan fungsi-fungsi `mysqli` lain yang akan kita pelajari sesaat lagi.

## 23.13 Function `mysqli_errno()` dan `mysqli_error()`

Kedua fungsi ini mirip dengan fungsi `mysqli_connect_errno()` dan `mysqli_connect_error()`. Bedanya, fungsi `mysqli_errno()` dan `mysqli_error()` digunakan untuk menampilkan error dari query SQL, bukan pada saat proses pembuatan koneksi.

Walaupun sifatnya opsional (tidak harus ditulis), fungsi `mysqli_errno()` dan `mysqli_error()` sangat bermanfaat untuk mengetahui apakah ada yang salah dari query SQL.

Apalagi seperti yang telah kita lihat, PHP tidak menampilkan error untuk kesalahan penulisan SQL. Berbeda dengan error saat koneksi yang walaupun tanpa fungsi `mysqli_connect_errno()` dan `mysqli_connect_error()`, error tetap ditampilkan.

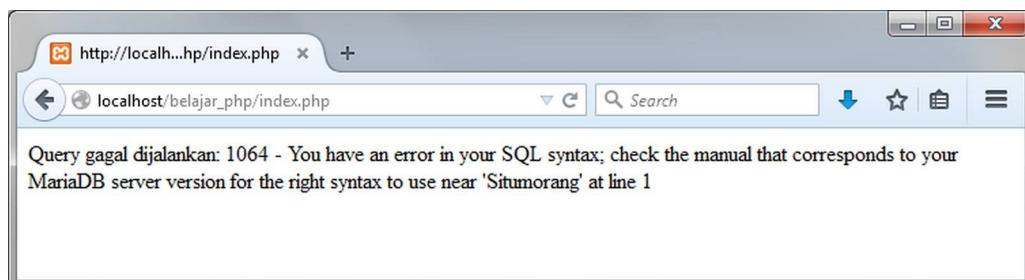
Fungsi `mysqli_errno()` dan `mysqli_error()` membutuhkan 1 argumen, yakni variabel handle hasil pemanggilan fungsi `mysqli_connect()`.

Berikut contoh penggunaan `mysqli_errno()` dan `mysqli_error()`:

```

1 <?php
2 // buat koneksi dengan database mysql
3 $koneksi = mysqli_connect("localhost","root","","universitas");
4
5 //periksa koneksi, tampilkan pesan kesalahan jika gagal
6 if (!$koneksi){
7     die ("Koneksi dengan database gagal: ".mysqli_connect_errno().
8          " - ".mysqli_connect_error());
9 }
10
11 //jalankan query
12 $nama = "James Situmorang";
13 $query = "SELECT * FROM mahasiswa WHERE nama=$nama";
14 $hasil_query = mysqli_query($koneksi, $query);
15
16 //periksa query, tampilkan pesan kesalahan jika gagal
17 if (!$hasil_query){
18     die ("Query gagal dijalankan: ".mysqli_errno($koneksi).
19          " - ".mysqli_error($koneksi));
20 }
21
22 // tutup koneksi dengan database mysql
23 mysqli_close($koneksi);
24 ?>

```



Gambar: Error penulisan query ditampilkan dengan fungsi mysqli\_errno() dan mysqli\_error()

Dari kode program diatas, terlihat cara penggunaan fungsi `mysqli_errno()` dan `mysqli_error()` nyaris sama seperti fungsi `mysqli_connect_errno()` dan `mysqli_connect_error()`.

Disini saya mengecek apakah sebuah query sukses dijalankan atau tidak menggunakan kondisi:

```

<?php
if (! $hasil_query) {

}
?>

```

Variabel `$hasil_query` akan bernilai `FALSE` jika fungsi `mysqli_query()` gagal dijalankan. Dengan me-negasi-kannya menggunakan operator “!”, fungsi `mysqli_errno()` dan `mysqli_error()`

hanya akan diproses ketika query SQL yang ditulis bermasalah.

Jika query tersebut tidak bermasalah, variabel `$hasil_query` akan berisi sesuatu (dan dikonversi menjadi TRUE) sehingga fungsi `mysqli_errno()` dan `mysqli_error()` tidak perlu dijalankan.

Juga sama seperti sebelumnya, untuk alasan keamanan pada web “live”, sebaiknya anda tidak menampilkan pesan error seperti ini, tapi cukup “*Query gagal dijalankan*”.

## 23.14 Function `mysqli_free_result()`

Fungsi `mysqli_free_result()` berfungsi untuk mengosongkan memory ketika hasil query sudah selesai dijalankan. Dengan kata lain, fungsi ini akan menghapus memory yang digunakan oleh fungsi `mysqli_query()`.

Sama seperti fungsi `mysqli_close()`, fungsi `mysqli_free_result()` juga bersifat opsional. Apabila tidak ditulis, PHP secara otomatis mengosongkan memory pada saat halaman selesai diproses.

Jika anda menjalankan banyak query di satu halaman, fungsi `mysqli_free_result()` akan membuat penggunaan memory lebih efisien. Karena setelah query selesai diproses, kita langsung menghapusnya dari memory.

Fungsi `mysqli_free_result()` membutuhkan 1 buah argumen, yakni variabel *handle* hasil pemanggilan fungsi `mysqli_query()`. Dalam contoh kita sebelumnya, variabel ini adalah `$hasil_query`.

Berikut contoh penggunaan fungsi `mysqli_free_result()`:

```
1 <?php
2 // buat koneksi dengan database mysql
3 $koneksi = mysqli_connect("localhost", "root", "", "universitas");
4
5 //periksa koneksi, tampilkan pesan kesalahan jika gagal
6 if (!$koneksi){
7     die ("Koneksi dengan database gagal: ".mysqli_connect_errno().
8          " - ".mysqli_connect_error());
9 }
10
11 //jalankan query
12 $nama = "James Situmorang";
13 $query = "SELECT * FROM mahasiswa WHERE nama='$nama'";
14 $hasil_query = mysqli_query($koneksi, $query);
15
16 //periksa query, tampilkan pesan kesalahan jika gagal
17 if (!$hasil_query){
18     die ("Query gagal dijalankan: ".mysqli_errno($koneksi).
19          " - ".mysqli_error($koneksi));
20 }
21
```

```
22 // bebaskan memory
23 mysqli_free_result($hasil_query);
24
25 // tutup koneksi dengan database mysql
26 mysqli_close($koneksi);
27 ?>
```

Jika di dalam satu halaman terdapat beberapa kali pemanggilan fungsi `mysqli_query()`, kita bisa menjalankan fungsi `mysqli_free_result()` secara bergantian, seperti contoh berikut:

```
1 //jalankan query pertama
2 $query = "SELECT * FROM mahasiswa";
3 $hasil_query = mysqli_query($koneksi, $query);
4
5 // .... pemrosesan query disini
6 // .... pemrosesan query disini
7
8 // bebaskan memory
9 mysqli_free_result($hasil_query);
10
11 //jalankan query kedua
12 $query = "SELECT * FROM jurusan";
13 $hasil_query = mysqli_query($koneksi, $query);
14
15 // .... pemrosesan query disini
16 // .... pemrosesan query disini
17
18 // bebaskan memory
19 mysqli_free_result($hasil_query);
```

## 23.15 Mempersiapkan Tabel Mahasiswa

Sedikit review, sampai disini kita telah membahas 4 dari 5 langkah cara komunikasi PHP dengan MySQL:

1. Buat koneksi dengan MySQL. Ini dilakukan menggunakan fungsi `mysqli_connect()`.
2. Jalankan query (SQL), hasilnya disimpan ke memory web server. Ini dilakukan menggunakan fungsi `mysqli_query()`.
3. Proses hasil query dengan PHP (jika ada). Belum kita bahas.
4. Kosongkan memory web server (opsional). Ini dilakukan menggunakan fungsi `mysqli_free_result()`.
5. Tutup koneksi dengan MySQL (opsional). Ini dilakukan menggunakan fungsi `mysqli_close()`.

Dalam sisa pembahasan bab ini saya akan fokus membahas langkah ke-3, yakni cara memproses hasil query. Disinilah nantinya kita akan menampilkan tabel MySQL ke dalam website menggunakan PHP.

Tapi sebelum itu, saya akan membuat tabel mahasiswa lengkap dengan isinya. Data ini akan menjadi tabel praktek kita. Silahkan buka MySQL client dari cmd, lalu jalankan 3 query berikut:

```
DROP TABLE IF EXISTS mahasiswa;

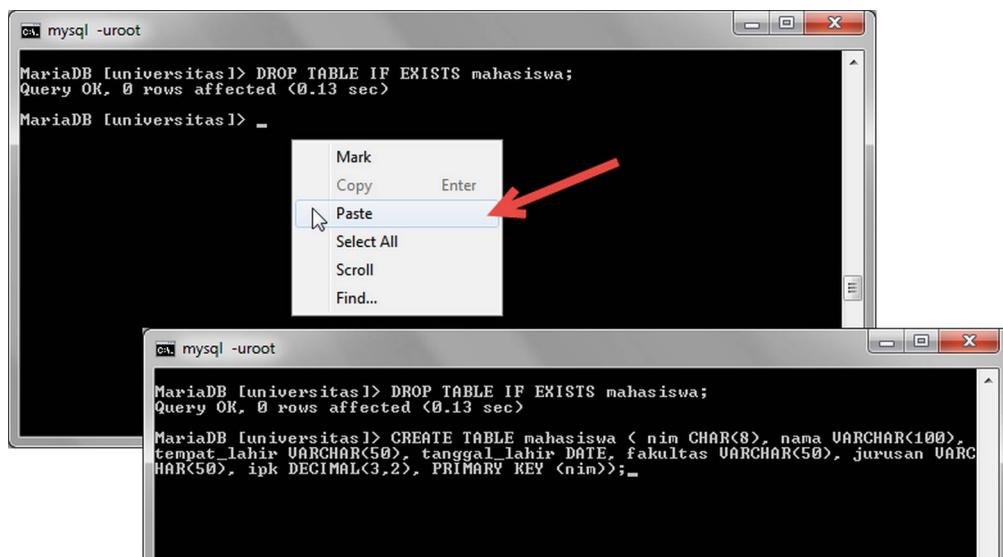
CREATE TABLE mahasiswa
( nim CHAR(8), nama VARCHAR(100), tempat_lahir VARCHAR(50),
  tanggal_lahir DATE, fakultas VARCHAR(50), jurusan VARCHAR(50),
  ipk DECIMAL(3,2), PRIMARY KEY (nim));

INSERT INTO mahasiswa VALUES
('15002032', 'Rina Kumala Sari', 'Jakarta', '1997-06-28', 'Ekonomi',
 'Akuntansi', 3.4),
('13012012', 'James Situmorang', 'Medan', '1995-04-02', 'Kedokteran',
 'Kedokteran Gigi', 2.7),
('14005011', 'Riana Putria', 'Padang', '1996-11-23', 'FMIPA',
 'Kimia', 3.1),
('15021044', 'Rudi Permana', 'Bandung', '1994-08-22', 'FASILKOM',
 'Ilmu Komputer', 2.9),
('15003036', 'Sari Citra Lestari', 'Jakarta', '1997-12-31', 'Ekonomi',
 'Manajemen', 3.5);
```

Query pertama digunakan untuk menghapus tabel mahasiswa. Perhatikan penambahan perintah IF EXISTS, ini digunakan jika tabel mahasiswa memang belum ada, query DROP TABLE tidak menghasilkan error.

Query kedua digunakan untuk membuat tabel mahasiswa dengan 7 kolom. Sedangkan query ketiga untuk menginput 5 data ke dalam tabel ini.

Anda bisa mencopy query diatas, lalu di pastekan ke cmd Windows. Caranya, blok teks diatas (satu per satu), lalu pada cmd MySQL Client, klik kanan -> paste. Berikut contohnya:



Gambar: Cara copy paste perintah query ke cmd MySQL Client

Jika tidak ada error, pastikan data mahasiswa sudah masuk semua dengan menjalankan query:

```
SELECT * FROM mahasiswa;
```

Seharusnya tampil seluruh data mahasiswa yang baru saja kita input. Baik, kita akan lanjut membahas fungsi-fungsi mysqli lainnya.

Jika anda menjalankan query **SELECT \* FROM mahasiswa** diatas, hasilnya akan menempatkan data *James Situmorang* di baris pertama, padahal pada saat proses **INSERT**, *Rina Kumala Sari* adalah data yang kita input pertama kali.

Ini terjadi karena kolom **nim** saya set sebagai **PRIMARY KEY**. Akibatnya, setiap data yang masuk akan diurutkan berdasarkan kolom **nim**, tidak peduli data itu diinput belakangan. James Situmorang memiliki angka **nim** yang paling kecil, karena itulah ia tampil di urutan pertama.

Urutan fisik suatu data di dalam database MySQL sebenarnya tidak begitu penting, karena kita bisa dengan mudah mengubah urutan ini pada saat menampilkan data. Misalnya dengan menggunakan perintah **ORDER BY**.

## 23.16 Function mysqli\_fetch\_rows()

Fungsi **mysqli\_fetch\_rows()** digunakan untuk mengambil 1 baris data tabel MySQL. Fungsi ini membutuhkan 1 argumen berupa variabel hasil pemanggilan fungsi **mysqli\_query()**, yakni variabel **\$hasil\_query**.

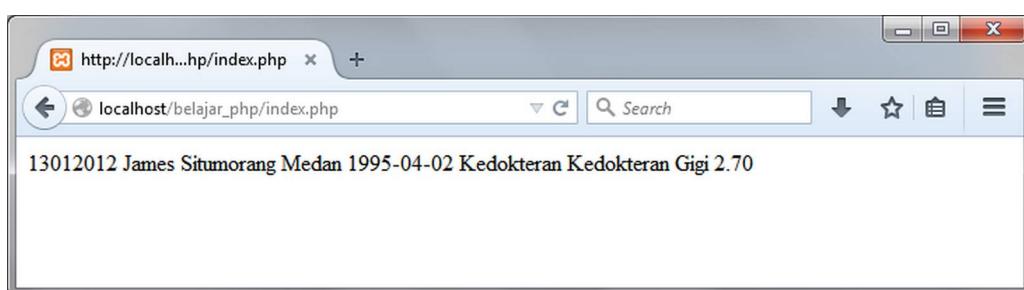
Fungsi **mysqli\_fetch\_rows()** mengembalikan 1 baris data berbentuk array. Key dari array ini berupa nomor urut kolom tabel MySQL. Mulai dari angka 0 untuk kolom pertama, angka 1 untuk kolom kedua, dst.

Agar mudah dipahami, langsung saja kita lihat cara penggunaannya:

```

1  <?php
2  // buat koneksi dengan database mysql
3  $koneksi = mysqli_connect("localhost","root","","universitas");
4
5  //periksa koneksi, tampilkan pesan kesalahan jika gagal
6  if (!$koneksi){
7      die ("Koneksi dengan database gagal: ".mysqli_connect_errno().
8          " - ".mysqli_connect_error());
9  }
10
11 //jalankan query
12 $query = "SELECT * FROM mahasiswa";
13 $hasil_query = mysqli_query($koneksi, $query);
14
15 //periksa query, tampilkan pesan kesalahan jika gagal
16 if (!$hasil_query){
17     die ("Query gagal dijalankan: ".mysqli_errno($koneksi).
18          " - ".mysqli_error($koneksi));
19 }
20
21 //tampilkan isi tabel mahasiswa
22 $data = mysqli_fetch_row($hasil_query);
23 echo "$data[0] $data[1] $data[2] $data[3] $data[4] $data[5] $data[6]";
24
25 // bebaskan memory
26 mysqli_free_result($hasil_query);
27
28 // tutup koneksi dengan database mysql
29 mysqli_close($koneksi);
30 ?>

```



Gambar: Data tabel MySQL ditampilkan dengan fungsi `mysql_fetch_rows()`

Pada kode program diatas, saya menjalankan query “`SELECT * FROM mahasiswa`”. Perintah ini akan mengembalikan seluruh isi tabel mahasiswa. Ini dijalankan menggunakan perintah:

```

1 <?php
2   $query = "SELECT * FROM mahasiswa";
3   $hasil_query = mysqli_query($koneksi, $query);
4 ?>

```

Disini, variabel \$hasil\_query sudah berisi data tabel mahasiswa. Hanya saja kita butuh fungsi tambahan untuk mengambil dan menampilkannya. Salah satu fungsi tersebut adalah **mysqli\_fetch\_row()**. Cara penggunaannya sebagai berikut:

```

1 $data = mysqli_fetch_row($hasil_query);
2 echo "$data[0] $data[1] $data[2] $data[3] $data[4] $data[5] $data[6]";
3
4 // hasil:
5 13012012 James Situmorang Medan 1995-04-02 Kedokteran Kedokteran Gigi 2.70

```

Variabel \$data berbentuk array dengan key nomor urut kolom. Sebagai contoh, \$data[0] akan menampilkan data pada kolom pertama, yakni **nim**, begitu seterusnya.

Jika anda ingin tampilan yang lebih detail, silahkan cek menggunakan fungsi **var\_dump()** atau **print\_r()**:

```

1 <?php
2   //tampilkan isi tabel mahasiswa
3
4   $data = mysqli_fetch_row($hasil_query);
5   echo "<pre>";
6   print_r($data);
7   echo "<pre>";
8 ?>
9
10  /* Output-----
11  Array
12  (
13    [0] => 13012012
14    [1] => James Situmorang
15    [2] => Medan
16    [3] => 1995-04-02
17    [4] => Kedokteran
18    [5] => Kedokteran Gigi
19    [6] => 2.70
20  )
-----*/

```

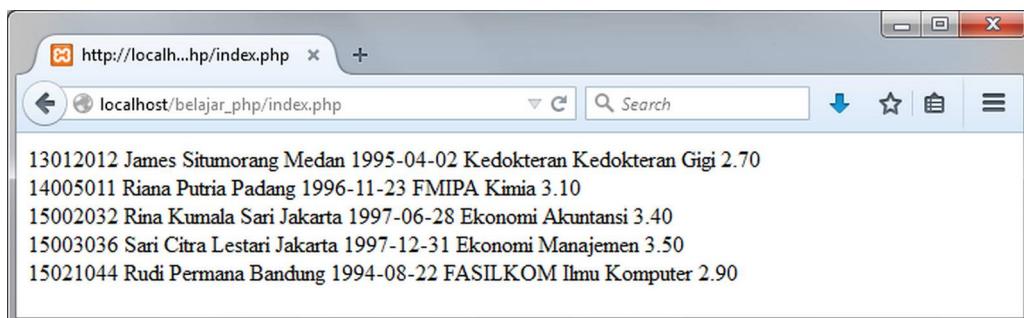
Baik, tapi bukankah tabel mahasiswa berisi 5 buah data? Kemana sisanya?

Seperti yang telah kita bahas, fungsi **mysqli\_fetch\_row()** hanya menambil 1 baris pada setiap pemanggilan. Untuk menampilkan 5 data, kita harus memanggil fungsi ini sebanyak 5 kali. Seperti contoh berikut:

```

1  <?php
2  //jalankan query
3  $query = "SELECT * FROM mahasiswa";
4  $hasil_query = mysqli_query($koneksi, $query);
5
6  //ambil dan tampilkan data baris 1
7  $data = mysqli_fetch_row($hasil_query);
8  echo "$data[0] $data[1] $data[2] $data[3] $data[4] $data[5] $data[6]";
9  echo "<br>";
10
11 //ambil dan tampilkan data baris 2
12 $data = mysqli_fetch_row($hasil_query);
13 echo "$data[0] $data[1] $data[2] $data[3] $data[4] $data[5] $data[6]";
14 echo "<br>";
15
16 //ambil dan tampilkan data baris 3
17 $data = mysqli_fetch_row($hasil_query);
18 echo "$data[0] $data[1] $data[2] $data[3] $data[4] $data[5] $data[6]";
19 echo "<br>";
20
21 //ambil dan tampilkan data baris 4
22 $data = mysqli_fetch_row($hasil_query);
23 echo "$data[0] $data[1] $data[2] $data[3] $data[4] $data[5] $data[6]";
24 echo "<br>";
25
26 //ambil dan tampilkan data baris 5
27 $data = mysqli_fetch_row($hasil_query);
28 echo "$data[0] $data[1] $data[2] $data[3] $data[4] $data[5] $data[6]";
29 ?>

```



Gambar: Semua isi tabel mahasiswa ditampilkan dengan memanggil fungsi `mysqli_fetch_row()` sebanyak 5 kali

Disini saya tidak lagi menulis lengkap kode PHP seperti fungsi `mysqli_connect()`, `mysqli_error()`, dll. Ini agar kode programnya tidak terlalu panjang, dan saya yakin anda sudah paham dimana menempatkan kode ini.

Karena saya menulisnya berulang-ulang, maka akan lebih efisien jika menggunakan perulangan:

```

1 <?php
2 //jalankan query
3 $query = "SELECT * FROM mahasiswa";
4 $hasil_query = mysqli_query($koneksi, $query);
5
6 //ambil dan tampilkan 5 baris tabel mahasiswa
7 for($i=1;$i<=5;$i++)
8 {
9     $data = mysqli_fetch_row($hasil_query);
10    echo "$data[0] $data[1] $data[2] $data[3] $data[4] $data[5] $data[6]";
11    echo "<br>";
12 }
13 ?>

```

Seperti yang terlihat, dengan memanggil fungsi `mysqli_fetch_row()` sebanyak 5 kali, saya bisa menampilkan 5 baris data mahasiswa. Ini bisa berjalan karena fungsi `mysqli_fetch_row()` sebenarnya juga melakukan “sesuatu” setiap kali diproses.

Selain menembalikan nilai, fungsi `mysqli_fetch_row()` secara otomatis menaikkan posisi pointer dari variabel `$hasil_query` pada tiap pemanggilan.

Masih ingat dengan konsep **pointer** pada array? Saya pernah membahas ini di bab **Array Function PHP**. Fungsi yang saya maksud adalah `current()`, `next()`, `prev()`, `end()` dan `reset()`.

Pada saat perintah `mysqli_fetch_row($hasil_query)` dijalankan, yang terjadi kira-kira sebagai berikut: “*ambil 1 baris data dari variabel \$hasil\_query, lalu next(\$hasil\_query)*”.

Dengan demikian dalam setiap pemanggilan, data yang diambil merupakan baris berikutnya. Ketika pointer sudah berada di posisi terakhir, pemanggilan fungsi `mysqli_fetch_row($hasil_query)` akan mengembalikan nilai **NULL**.

Karena alasan inilah kita bisa menggunakan perulangan berikut:

```

1 <?php
2 //jalankan query
3 $query = "SELECT * FROM mahasiswa";
4 $hasil_query = mysqli_query($koneksi, $query);
5
6 //ambil dan tampilkan seluruh tabel mahasiswa
7 while($data = mysqli_fetch_row($hasil_query))
8 {
9     echo "$data[0] $data[1] $data[2] $data[3] $data[4] $data[5] $data[6]";
10    echo "<br>";
11 }
12 ?>

```

Perhatikan kode program yang ada di dalam tanda kurung setelah perintah `while`. Biasanya, disini merupakan tempat untuk sebuah kondisi, seperti `while ($i<=0)`, atau `while ($harga > 50000)`.

Baris `while($data = mysqli_fetch_row($hasil_query))`, artinya: “selama `$data` tidak berisi nilai `NULL` (yang akan dikonversi menjadi boolean `FALSE`), ambil setiap baris dari variabel `$hasil_query` menggunakan fungsi `mysqli_fetch_row()`”.

Perulangan ini akan terus dijalankan selama masih ada data di dalam `$hasil_query`. Ingat, fungsi `mysqli_fetch_row()` otomatis menaikkan nilai pointer, jadi kita tidak perlu instruksi tambahan lain.

Menampilkan hasil query menggunakan perulangan `while` seperti ini sangat sangat umum digunakan. Tetapi tidak banyak yang membahas cara kerjanya.

Bagaimana dengan perulangan `foreach`? Bukankah variabel `$hasil_query` berbentuk array? Jadi seharusnya kita juga bisa menggunakan `foreach` bukan?

Variabel `$hasil_query` bukanlah sebuah array, tapi merupakan variabel khusus yang *mirip seperti array*. Oleh karena itu kita tidak bisa menggunakan perulangan `foreach` untuk menampilkan hasil query MySQL. Kita bisa menggunakan `foreach` untuk variabel `$data`, tapi tidak bisa langsung ke variabel `$hasil_query`.

## 23.17 Function `mysqli_fetch_assoc()`

Fungsi `mysqli_fetch_assoc()` sangat mirip dengan `mysqli_fetch_row()`. Bedanya, array yang dikembalikan berupa *associative array*. Disini key array bukan lagi angka, tapi nama kolom tabel.

Apabila sebelumnya saya mengakses kolom `nim` menggunakan `$data[0]`, dengan fungsi `mysqli_fetch_assoc()`, sekarang bisa diakses dari `$data[nim]`. Mari kita test menggunakan fungsi `print_r()`:

```
1 <?php
2 //tampilkan isi tabel mahasiswa
3 $data = mysqli_fetch_assoc($hasil_query);
4
5 echo "<pre>";
6 print_r($data);
7 echo "<pre>";
8 ?>
9
10 /* Output-----
11 Array
12 (
13     [nim] => 13012012
14     [nama] => James Situmorang
15     [tempat_lahir] => Medan
16     [tanggal_lahir] => 1995-04-02
17     [fakultas] => Kedokteran
18     [jurusan] => Kedokteran Gigi
19     [ipk] => 2.70
```

```
20  )
21  -----*/
```

Seperti yang terlihat, key dari array ini adalah nama kolom dari tabel. Selain perbedaan ini, cara penggunaan fungsi `mysqli_fetch_row()` sama persis dengan fungsi `mysqli_fetch_assoc()`. Termasuk menggunakan perulangan `while` sebagai berikut:

```
1  <?php
2      //jalankan query
3      $query = "SELECT * FROM mahasiswa";
4      $hasil_query = mysqli_query($koneksi, $query);
5
6      //ambil dan tampilkan seluruh tabel mahasiswa
7      while($data = mysqli_fetch_assoc($hasil_query))
8      {
9          echo "$data[nim] $data[nama] $data[tempat_lahir] ";
10         echo "$data[tanggal_lahir] $data[fakultas] ";
11         echo "$data[jurusan] $data[ipk]";
12         echo "<br>";
13     }
14 ?>
```

Jika anda menjalankan kode program diatas, hasilnya akan menampilkan 5 data dari tabel mahasiswa. Persis seperti hasil ketika menggunakan fungsi `mysqli_fetch_row()`.

Menggunakan fungsi `mysqli_fetch_assoc()` terasa lebih mudah dibandingkan fungsi `mysqli_fetch_row()`. Sebelumnya, kita harus menebak apakah isi dari `$data[3]`. Tapi jika ditulis dengan `$data[fakultas]`, sudah jelas ini berisi data dari kolom fakultas.

Dari beberapa referensi, ada yang menyatakan fungsi `mysqli_fetch_assoc()` sedikit lebih lambat daripada fungsi `mysqli_fetch_row()`. Karena PHP harus mengambil data tentang nama kolom untuk dijadikan sebagai key dari *associative array*.

Akan tetapi perbedaan kecepatan ini baru terasa jika anda melakukan pemrosesan ratusan ribu query pada saat yang bersamaan. Untuk penggunaan sehari-hari, perbedaan kecepatan ini tidak akan terasa. Karena lebih nyaman digunakan, saya merekomendasikan menggunakan `mysqli_fetch_assoc()` untuk menampilkan data tabel MySQL.

## 23.18 Function `mysqli_fetch_array()`

Fungsi `mysqli_fetch_array()` bisa dibilang sebagai gabungan dari fungsi `mysqli_fetch_row()` dan `mysqli_fetch_assoc()`. Disini, data dari fungsi `mysqli_fetch_array()` bisa diakses baik menggunakan nomor key maupun nama kolom.

Berikut hasil yang didapat ketika menggunakan fungsi `print_r()`:

```

1  <?php
2  //tampilkan isi tabel mahasiswa
3  $data = mysqli_fetch_array($hasil_query);
4
5  echo "<pre>";
6  print_r($data);
7  echo "<pre>";
8 ?>
9
10 /* Output-----
11   Array
12 (
13     [0] => 13012012
14     [nim] => 13012012
15     [1] => James Situmorang
16     [nama] => James Situmorang
17     [2] => Medan
18     [tempat_lahir] => Medan
19     [3] => 1995-04-02
20     [tanggal_lahir] => 1995-04-02
21     [4] => Kedokteran
22     [fakultas] => Kedokteran
23     [5] => Kedokteran Gigi
24     [jurusan] => Kedokteran Gigi
25     [6] => 2.70
26     [ipk] => 2.70
27 )
28 -----*/

```

Terlihat array \$data berisi dua buah data, satu untuk array biasa, dan satu lagi berupa *associative array*. Sehingga saya bisa menulis kode program berikut:

```

1  <?php
2  //jalankan query
3  $query = "SELECT * FROM mahasiswa";
4  $hasil_query = mysqli_query($koneksi, $query);
5
6  //ambil dan tampilkan seluruh tabel mahasiswa
7  while($data = mysqli_fetch_array($hasil_query))
8  {
9    echo "$data[0] $data[nama] $data[tempat_lahir] ";
10   echo "$data[3] $data[4] $data[jurusan] $data[6]";
11   echo "<br>";
12 }
13 ?>

```

Disini saya menggunakan 2 jenis key array: penomoran dan nama kolom.

Fungsi `mysqli_fetch_array()` memiliki argumen kedua yang bersifat opsional. Argumen ini bisa diisi dengan salah satu dari 3 konstanta. Fungsinya, untuk mengatur bagaimana struktur array yang dikembalikan, apakah angka saja, nama kolom saja, atau keduanya:

- **MYSQLI\_BOTH**: Ini merupakan nilai default. Fungsi `mysqli_fetch_array()` akan mengembalikan array yang bisa diakses dengan key angka maupun nama kolom.
- **MYSQLI\_NUM**: Array yang dikembalikan hanya bisa diakses dengan angka. Hasil ini sama persis dengan fungsi `mysqli_fetch_row()`.
- **MYSQLI\_ASSOC**: Array yang dikembalikan berupa associative array, dengan key berupa nama kolom. Hasil ini sama persis dengan fungsi `mysqli_fetch_assoc()`.

Sebagai contoh, saya bisa mencopy kode program dari `mysqli_fetch_row()` menjadi seperti berikut:

```
1 <?php
2 //jalankan query
3 $query = "SELECT * FROM mahasiswa";
4 $hasil_query = mysqli_query($koneksi, $query);
5
6 //ambil dan tampilkan 1 baris tabel mahasiswa
7 $data = mysqli_fetch_array($hasil_query, MYSQLI_NUM);
8 echo "$data[0] $data[1] $data[2] $data[3] $data[4] $data[5] $data[6]";
9 // 13012012 James Situmorang Medan 1995-04-02
10 // Kedokteran Kedokteran Gigi 2.70
11 ?>
```

Di balik kenyamanan yang ditawarkan, fungsi `mysqli_fetch_array()` butuh penggunaan memory 2 kali lebih banyak daripada fungsi `mysqli_fetch_row()` maupun `mysqli_fetch_assoc()`, karena seperti yang kita lihat dari hasil `print_r()`, array yang diambil juga 2 kali: satu untuk array normal, satu lagi berupa associative array.

Jika anda tidak memerlukan kode program yang harus menampilkan kedua jenis array ini, sebaiknya tetap menggunakan fungsi `mysqli_fetch_row()` atau `mysqli_fetch_assoc()`.

## 23.19 Function `mysqli_num_rows()`

Fungsi `mysqli_num_rows()` digunakan untuk mencari informasi berapa banyak baris data (record) dari pemanggilan query `SELECT`. Fungsi ini membutuhkan 1 argumen, yakni variabel `$hasil_query` dan mengembalikan nilai integer, tergantung berapa banyak data yang dimakan.

Berikut contoh penggunaan fungsi `mysqli_num_rows()`:

```

1 <?php
2 //jalankan query
3 $query = "SELECT * FROM mahasiswa";
4 $hasil_query = mysqli_query($koneksi, $query);
5
6 // cek jumlah record (baris)
7 $jumlah_data = mysqli_num_rows($hasil_query);
8 echo $jumlah_data; // 5
9 ?>

```

Query **SELECT \* FROM mahasiswa** akan mengembalikan seluruh data yang terdapat di dalam tabel mahasiswa. Karena di dalam tabel ini ada 5 record, fungsi *mysqli\_num\_rows()* akan mengembalikan angka 5. Angka ini saya simpan ke dalam variabel `$jumlah_data`.

Fungsi **mysqli\_num\_rows()** juga sering digunakan untuk mendeteksi apakah suatu data sudah ada di dalam tabel atau tidak. Berikut contohnya:

```

1 <?php
2 //jalankan query
3 $query = "SELECT * FROM mahasiswa WHERE nim='15021044'";
4 $hasil_query = mysqli_query($koneksi, $query);
5
6 // cek jumlah record (baris)
7 $jumlah_data = mysqli_num_rows($hasil_query);
8 if ($jumlah_data == 1) {
9     echo "Data ditemukan";
10 }
11 else {
12     echo "Data tidak ditemukan";
13 }
14 ?>

```

Kali ini query yang saya jalankan adalah **SELECT \* FROM mahasiswa WHERE nim='15021044'**. Artinya saya mencari apakah di dalam tabel mahasiswa ada yang memiliki nomor nim '15021044'. Jika ada, fungsi **mysqli\_num\_rows()** akan mengembalikan nilai 1. Nilai inilah yang kemudian saya periksa. Jika hasilnya 1, berarti ada di dalam database. Jika tidak ada, tentu hasilnya 0.

Teknik seperti ini sering digunakan untuk pengecekan *username*. Contohnya pada saat menginput data baru kedalam tabel. Sebelum menjalankan query **INSERT**, kita harus pastikan **nim** yang diinput belum ada di database. Jika sudah ada, kode program akan error, karena untuk kolom **nim** tidak boleh ada data yang sama (karena telah di set sebagai **PRIMARY KEY**).

## 23.20 Function **mysqli\_affected\_rows()**

Dari semua fungsi **mysqli** yang sudah kita pelajari, saya baru menjelaskan cara menampilkan data dari database (query **SELECT**). Bagaimana dengan perintah **INSERT**, **UPDATE**, **CREATE** dan **DELETE**?

Pada dasarnya sama dengan cara menjalankan query **SELECT**, dimana kita tetap menggunakan fungsi **mysqli\_query()**. Bedanya, jika di query **SELECT** kita mengharapkan ‘sesuatu’ untuk ditampilkan, untuk query **INSERT**, **UPDATE** dan **CREATE** kita hanya butuh konfirmasi apakah query tersebut berhasil dijalankan atau tidak. Konfirmasi ini bisa diperiksa dari hasil fungsi **mysqli\_query()**.

Sebagai contoh, saya ingin menambahkan data baru ke dalam tabel mahasiswa. Berikut kode programnya:

```
1 <?php
2 // buat koneksi dengan database mysql
3 $koneksi = mysqli_connect("localhost", "root", "", "universitas");
4
5 //periksa koneksi, tampilkan pesan kesalahan jika gagal
6 if (!$koneksi) {
7     die ("Koneksi dengan database gagal: ".mysqli_connect_errno().
8          " - ".mysqli_connect_error());
9 }
10
11 //jalankan query
12 $query = "INSERT INTO mahasiswa VALUES ";
13 $query .= "('14021022', 'Surya Permata', 'Ambon', ";
14 $query .= "'1996-11-06', 'FASILKOM', 'Ilmu Komputer', 2.8)";
15
16 $hasil_query = mysqli_query($koneksi, $query);
17
18 //periksa query, tampilkan pesan kesalahan jika gagal
19 if($hasil_query) {
20     // query berhasil dijalankan
21     echo "Data baru sukses ditambah";
22 }
23 else {
24     die ("Query gagal dijalankan: ".mysqli_errno($koneksi).
25          " - ".mysqli_error($koneksi));
26 }
27
28 // bebaskan memory
29 mysqli_free_result($hasil_query);
30
31 // tutup koneksi dengan database mysql
32 mysqli_close($koneksi);
33 ?>
```

Terdapat beberapa modifikasi dari kode program kita sebelumnya. Untuk query MySQL saya memisahkannya menjadi 3 baris string:

```

1 $query = "INSERT INTO mahasiswa VALUES ";
2 $query .= "('14021022', 'Surya Permata', 'Amboin', ";
3 $query .= "'1996-11-06', 'FASILKOM', 'Ilmu Komputer', 2.8)";

```

Masih ingat dengan operator “.=”, ini sama artinya dengan “\$query = \$query . string”. Disini string \$query saya sambung dalam tiap baris. Ini semata-mata karena keterbatasan panjang buku. Di dalam teks editor, anda bisa menulis seluruh query ini dalam 1 baris panjang.

Selanjutnya, query diatas dijalankan menggunakan fungsi *mysqli\_query()*. Tapi bagaimana cara memastikan data telah berhasil diinput?

Caranya dengan memeriksa apakah variabel *\$hasil\_query* bernilai TRUE atau FALSE.

Ketika kita menjalankan query **SELECT**, variabel *\$hasil\_query* akan bertipe *resources* (bersisi seluruh data dari tabel MySQL). Akan tetapi jika yang dijalankan adalah query **INSERT**, **UPDATE**, **CREATE** atau **DELETE**, variabel *\$hasil\_query* hanya berisi 2 nilai: TRUE kalau query sukses dijalankan, atau FALSE jika query gagal dieksekusi.

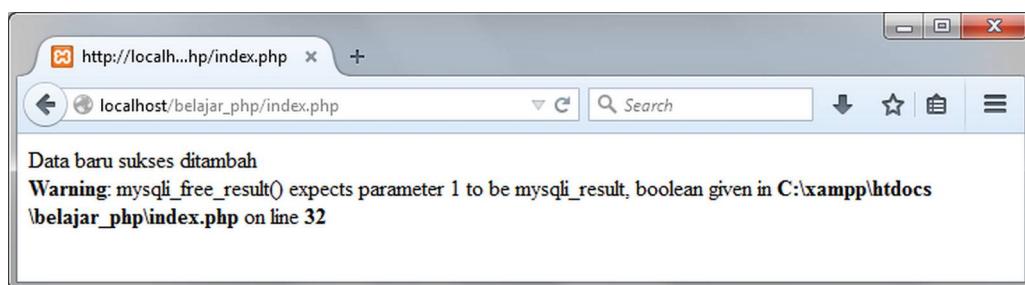
Kita bisa menggunakan output ini untuk menampilkan pesan konfirmasi:

```

1 <?php
2 //periksa query, tampilkan pesan kesalahan jika gagal
3 if($hasil_query) {
4     // query berhasil dijalankan
5     echo "Data baru sukses ditambah";
6 }
7 else {
8     die ("Query gagal dijalankan: ".mysqli_errno($koneksi).
9           " - ".mysqli_error($koneksi));
10 }
11 ?>

```

Baik, bagaimana hasilnya?



Gambar: Data berhasil ditambahkan. Tapi kenapa ada pesan error?

Pertama, terdapat baris “*Data baru sukses ditambah*”. Ini artinya data “*Surya Permata*” telah berhasil ditambahkan ke dalam tabel mahasiswa. Tapi, kenapa ada error? Lebih jauh lagi, PHP menginfokan error terdapat di fungsi *mysqli\_free\_result()*.

Seperti yang telah kita bahas, fungsi *mysqli\_free\_result()* digunakan untuk mengosongkan memory yang terpakai. Akan tetapi jika query yang dijalankan bukan **SELECT**, tidak ada data

yang mesti dihapus. Variabel `$hasil_query` tidak lagi berupa *resources*, tapi hanya boolean TRUE atau FALSE. Kesimpulannya, kita tidak perlu menjalankan fungsi `mysqli_free_result()` untuk query selain SELECT.

Silahkan hapus fungsi `mysqli_free_result()`, dan kita jalankan lagi kode program diatas:



Gambar: Ng.... Masih ada error lagi?

Masih ada pesan error, tapi berbeda dari sebelumnya:

```
Query gagal dijalankan: 1062 - Duplicate entry '14021022' for key 'PRIMARY'
```

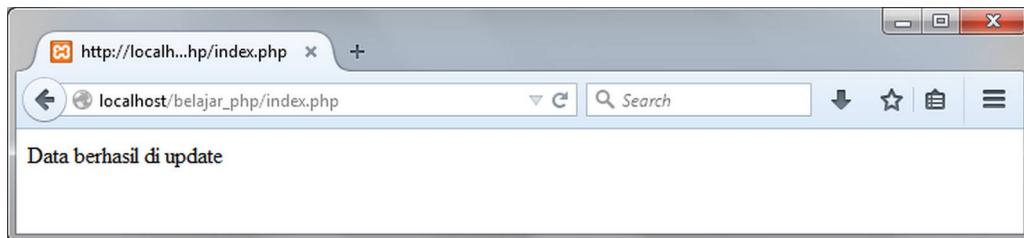
Error ini bukan berasal dari PHP, tapi dari MySQL. Di dalam kode program sebelumnya saya menambahkan data baru. Ketika halaman di reload, data yang ingin ditambahkan masih data "Surya Permata".

Akibatnya, MySQL akan complain karena data tersebut sudah ada di dalam database, ini disebabkan kolom **nim** sudah diset sebagai **PRIMARY KEY**, sehingga tidak boleh ada data yang sama. Jika saya tidak menggunakan fungsi `mysqli_error()`, pesan kesalahan seperti ini tidak akan terdeteksi.

Bagaimana dengan query **UPDATE**? Caranya sama persis, kita tinggal mengganti perintah SQL saja. Misalkan saya ingin mengupdate nilai **ipk** *Riana Putria* menjadi 4.0, saya bisa menggunakan kode program berikut:

```

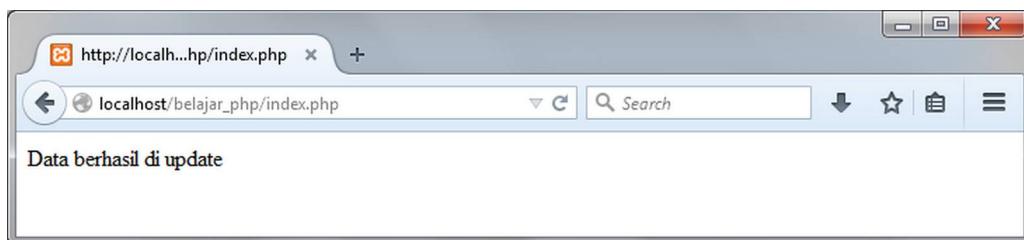
1  <?php
2  //jalankan query
3  $query = "UPDATE mahasiswa SET ipk=4.0 WHERE nama='Riana Putria'";
4  $hasil_query = mysqli_query($koneksi, $query);
5
6  //periksa query, tampilkan pesan kesalahan jika gagal
7  if($hasil_query) {
8      // update berhasil
9      echo "Data berhasil di update";
10 }
11 else {
12     die ("Query gagal dijalankan: ".mysqli_errno($koneksi).
13          " - ".mysqli_error($koneksi));
14 }
15 ?>
```



Gambar: Data tabel mahasiswa berhasil di Update

Silahkan anda jalankan kode program diatas kemudian periksa isi tabel mahasiswa. Seharusnya nilai ipk *Riana Putria* sudah naik menjadi 4.0.

Mari kita jalankan kode tersebut beberapa kali (tanpa mengubah query). Disini anda mungkin memprediksi akan terdapat error, yang penyebabnya sama seperti kasus pada query **INSERT**. Karena tentu saja nilai ipk saat ini sudah 4.0.



Gambar: Tidak ada error, walaupun data tidak diupdate

Tapi apa yang terjadi? Kenapa tidak error?

Ketika kita menjalankan query **UPDATE** dan **DELETE**, fungsi **mysqli\_query()** tetap mengembalikan nilai **TRUE**, walaupun tidak ada perubahan di sisi MySQL. Tentunya sepanjang penulisan query tersebut benar.

Jadi bagaimana cara memeriksa apakah query **UPDATE** ini mengupdate sesuatu di tabel, atau tidak ada data yang diupdate? Kita bisa menggunakan fungsi **mysqli\_affected\_rows()**.

Fungsi **mysqli\_affected\_rows()** mengembalikan nilai berapa banyak kolom yang diubah oleh sebuah query. Jika anda perhatikan, setiap perintah query yang kita input dari cmd, ada tambahan keterangan seperti "*Query OK, 2 rows affected (0.05 sec)*", berikut contohnya:

```

mysql -u root
MariaDB [universitas]> UPDATE mahasiswa SET ipk=4.0 WHERE nama='Riana Putria';
Query OK, 1 row affected <0.04 sec>
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [universitas]> UPDATE mahasiswa SET ipk=3.8 WHERE nama='Riana Putria';
Query OK, 1 row affected <0.04 sec>
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [universitas]> UPDATE mahasiswa SET ipk=3.8 WHERE nama='Riana Putria';
Query OK, 0 rows affected <0.04 sec>
Rows matched: 1  Changed: 0  Warnings: 0
MariaDB [universitas]>
  
```

Gambar: Keterangan berapa baris yang terkena proses (affected)

Dalam gambar diatas, saya mengubah nilai **ipk** beberapa kali. Ketika di dalam tabel terdapat data yang diupdate, dibawah query ada keterangan: *Query OK, 1 row affected*. Namun jika ada data yang diubah, keterangannya menjadi *Query OK, 0 row affected*. Angka inilah yang bisa kita deteksi menggunakan fungsi **mysqli\_affected\_rows()**.

Fungsi `mysqli_affected_rows()` membutuhkan 1 buah argumen berupa variabel `$koneksi`, (ingat, bukan variabel `$hasil_query`).

Baik, mari kita modifikasi kode program sebelumnya:

```

1 <?php
2 //jalankan query
3 $query = "UPDATE mahasiswa SET ipk=3.1 WHERE nama='Riana Putria'";
4 $hasil_query = mysqli_query($koneksi, $query);
5
6 //periksa query, tampilkan pesan kesalahan jika gagal
7 if($hasil_query) {
8     // update berhasil, cek apakah ada data yang diupdate
9     if ($jumlah_update = mysqli_affected_rows($koneksi)) {
10         echo "Query berhasil, terdapat $jumlah_update data yang diupdate.";
11     }
12     else {
13         echo "Query berhasil, tidak ada data yang diupdate.";
14     }
15 }
16 else {
17     die ("Query gagal dijalankan: ".mysqli_errno($koneksi).
18          " - ".mysqli_error($koneksi));
19 }
20 ?>

```

Silahkan anda pelajari sebentar kode program diatas, terutama pada penulisan kondisi IF. Disini saya menggunakan if bersarang (*nested if*), yakni ada if di dalam if.

Alur dari kode program diatas adalah:

1. Periksa variabel `$hasil_query`. Jika bernilai **TRUE** (query tidak bermasalah), cek apakah fungsi `mysqli_affected_rows()` mengembalikan nilai angka atau 0. Jika berupa angka (dikonversi menjadi boolean **TRUE**), tampilkan pesan “*Query berhasil, terdapat \$jumlah\_update data yang diupdate*.”
2. Periksa variabel `$hasil_query`. Jika bernilai **TRUE** (query tidak bermasalah), cek apakah fungsi `mysqli_affected_rows()` mengembalikan nilai angka atau 0. Jika berisi angka 0 (dikonversi menjadi boolean **FALSE**), tampilkan pesan “*Query berhasil, tidak ada data yang diupdate*.”
3. Periksa variabel `$hasil_query`. Jika bernilai **FALSE** (berarti query bermasalah), tampilkan pesan error dari fungsi `mysqli_errno()` dan `mysqli_error()`.

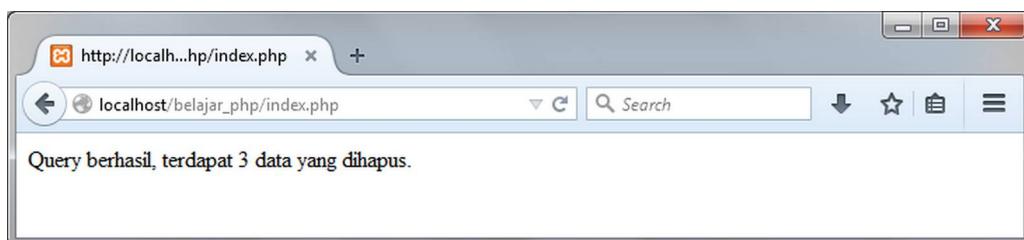
Setelah query **SELECT**, **INSERT** dan **UPDATE**, mari kita lihat penggunaan query **DELETE**. Sekali lagi, tidak ada yang berubah dari fungsi-fungsi PHP. Kita tinggal menukar string untuk variabel `$query` dengan perintah **DELETE**, kemudian mengatur tampilan pesan `echo`.

Sebagai contoh, saya ingin menghapus beberapa data dari tabel mahasiswa. Berikut kode programnya:

```

1  <?php
2      //jalankan query
3      $query = "DELETE FROM mahasiswa WHERE fakultas='FASILKOM' ";
4      $query .= "OR fakultas='Ekonomi'";
5      $hasil_query = mysqli_query($koneksi, $query);
6
7      //periksa query, tampilkan pesan kesalahan jika gagal
8      if($hasil_query) {
9          // delete berhasil, cek apakah ada data yang dihapus
10         if ($jumlah_delete = mysqli_affected_rows($koneksi)) {
11             echo "Query berhasil, terdapat $jumlah_delete data yang dihapus.";
12         }
13         else {
14             echo "Query berhasil, tidak ada data yang dihapus.";
15         }
16     }
17     else {
18         die ("Query gagal dijalankan: ".mysqli_errno($koneksi).
19             " - ".mysqli_error($koneksi));
20     }
21 ?>

```



Gambar: 3 data sudah dihapus dari tabel mahasiswa

Selain menukar query menjadi **DELETE**, saya juga mengubah beberapa variabel agar pas penamaannya. Disini fungsi **mysqli\_affected\_rows()** tetap berperan untuk menghitung berapa data yang terhapus akibat query **DELETE** ini. Terlihat ada 3 data yang sesuai dengan kondisi **WHERE** dari query, yakni mahasiswa yang kolom fakultas berisi *FASILKOM* atau *Ekonomi*.

## 23.21 Case Study: Membuat Tabel **mahasiswa\_baru**

Dari semua fungsi **mysqli** yang telah kita pelajari, bisa dibilang sudah mencukupi untuk penggunaan “sehari-hari”. Misalnya apakah itu untuk tugas, skripsi, atau proyek-proyek sederhana.

Namun fungsi-fungsi **mysqli** ini tidak akan efektif jika anda juga belum menguasai query MySQL. Sebagian besar masalah atau error terjadi disebabkan karena salah menulis query.

Sebagai bahan praktek, disini saya akan merancang sebuah kode program untuk membuat, menampilkan dan menghapus tabel **mahasiswa\_baru**. Tabel **mahasiswa\_baru** sebenarnya sama persis seperti tabel mahasiswa yang saya gunakan sebelumnya.

Kode program berikut cukup panjang, silahkan pelajari dengan perlahan:

```
1 <?php
2 // buat koneksi dengan database mysql
3 $dbhost = "localhost";
4 $dbuser = "root";
5 $dbpass = "";
6 $dbname = "universitas";
7 $koneksi = mysqli_connect($dbhost,$dbuser,$dbpass,$dbname);
8
9 //periksa koneksi, tampilkan pesan kesalahan jika gagal
10 if(!$koneksi){
11     die ("Koneksi dengan database gagal: ".mysqli_connect_errno().
12          " - ".mysqli_connect_error());
13 }
14
15 // cek apakah tabel mahasiswa_baru sudah ada. jika ada, hapus tabel
16 $query = "DROP TABLE IF EXISTS mahasiswa_baru";
17 $hasil_query = mysqli_query($koneksi, $query);
18
19 if(!$hasil_query){
20     die ("Query Error: ".mysqli_errno($koneksi).
21          " - ".mysqli_error($koneksi));
22 }
23
24 // buat query untuk CREATE tabel mahasiswa_baru
25 $query = "CREATE TABLE mahasiswa_baru (nim CHAR(8), nama VARCHAR(100), ";
26 $query .= "tempat_lahir VARCHAR(50), tanggal_lahir DATE, ";
27 $query .= "fakultas VARCHAR(50), jurusan VARCHAR(50), ";
28 $query .= "ipk DECIMAL(3,2), PRIMARY KEY (nim))";
29
30 $hasil_query = mysqli_query($koneksi, $query);
31
32 if(!$hasil_query){
33     die ("Query Error: ".mysqli_errno($koneksi).
34          " - ".mysqli_error($koneksi));
35 }
36
37 // buat query untuk INSERT data ke tabel mahasiswa_baru
38 $query = "INSERT INTO mahasiswa_baru VALUES ";
39 $query .= "('14005011', 'Riana Putria', 'Padang', '1996-11-23', ";
40 $query .= "'FMIPA', 'Kimia', 3.1), ";
41 $query .= "('15021044', 'Rudi Permana', 'Bandung', '1994-08-22', ";
42 $query .= "'FASILKOM', 'Ilmu Komputer', 2.9), ";
43 $query .= "('15003036', 'Sari Citra Lestari', 'Jakarta', '1997-12-31', ";
44 $query .= "'Ekonomi', 'Manajemen', 3.5)";
```

```
45
46     $hasil_query = mysqli_query($koneksi, $query);
47
48     if(!$hasil_query){
49         die ("Query Error: ".mysqli_errno($koneksi) .
50              " - ".mysqli_error($koneksi));
51     }
52     ?>
53     <!DOCTYPE html>
54     <html>
55     <head>
56         <meta charset="UTF-8">
57         <title>Belajar PHP</title>
58     </head>
59     <body>
60         <h1>Tabel Mahasiswa</h1>
61         <table border="1">
62             <tr>
63                 <th>NIM</th>
64                 <th>Nama</th>
65                 <th>Tempat Lahir</th>
66                 <th>Tanggal Lahir</th>
67                 <th>Fakultas</th>
68                 <th>Jurusan</th>
69                 <th>IPK</th>
70             </tr>
71             <?php
72             // buat query untuk menampilkan seluruh data tabel mahasiswa_baru
73             $query = "SELECT * FROM mahasiswa_baru";
74             $hasil_query = mysqli_query($koneksi, $query);
75
76             if(!$hasil_query){
77                 die ("Query Error: ".mysqli_errno($koneksi) .
78                      " - ".mysqli_error($koneksi));
79             }
80
81             //buat perulangan untuk element tabel dari data mahasiswa_baru
82             while($data = mysqli_fetch_assoc($hasil_query))
83             {
84                 echo "<tr>";
85                 echo "<td>$data[nim]</td>";
86                 echo "<td>$data[nama]</td>";
87                 echo "<td>$data[tempat_lahir]</td>";
88                 echo "<td>$data[tanggal_lahir]</td>";
89                 echo "<td>$data[fakultas]</td>";
90                 echo "<td>$data[jurusan]</td>";
```

```

91     echo "<td>$data[ipk]</td>";
92     echo "</tr>";
93 }
94
95 // bebaskan memory
96 mysqli_free_result($hasil_query);
97 ?>
98 </table>
99 </body>
100 </html>
101 <?php
102 // tutup koneksi dengan database mysql
103 mysqli_close($koneksi);
104 ?>

```



The screenshot shows a web browser window titled 'Belajar PHP'. The address bar displays 'localhost/belajar\_php/index.php'. The main content area shows a table titled 'Tabel Mahasiswa' with the following data:

NIM	Nama	Tempat Lahir	Tanggal Lahir	Fakultas	Jurusan	IPK
14005011	Riana Putria	Padang	1996-11-23	FMIPA	Kimia	3.10
15003036	Sari Citra Lestari	Jakarta	1997-12-31	Ekonomi	Manajemen	3.50
15021044	Rudi Permana	Bandung	1994-08-22	FASILKOM	Ilmu Komputer	2.90

Gambar: Tampilan isi tabel Mahasiswa Baru

Dalam halaman ini saya membuat 4 buah query: Untuk menghapus tabel *mahasiswa\_baru* jika sebelumnya sudah ada (query **DROP**), membuat tabel *mahasiswa\_baru* (query **CREATE**), menambah data baru (**INSERT**) dan menampilkan data (**SELECT**).

Untuk 3 query pertama, caranya relatif sederhana. Buat query, jalankan dengan fungsi **mysqli\_query()**, lalu cek apakah ada error. Untuk query yang panjang seperti **CREATE table** dan **INSERT**, saya memecahnya menjadi beberapa baris. Anda bisa membuatnya ke dalam 1 baris panjang. Ini semata2 karena keterbatasan lebar buku ini.

Pada query **SELECT**, saya menampilkannya ke dalam sebuah tabel HTML. Perhatikan bagaimana saya menggabungkan tag-tag tabel HTML dan menyelipkan variabel \$data diantara tag **<td>** dan **</td>**.

Jika anda sudah menguasai HTML dan PHP, saya rasa sudah bisa memahami kode program diatas. Sebagai latihan, bisakah anda menampilkan data ini kedalam list? Atau bisa juga uji skill CSS anda dengan mengubah desain tabel diatas agar tampak lebih menarik.

## 23.22 SQL Injection

**SQL Injection** adalah sebuah cara menyisipkan perintah SQL ke dalam kode program yang sudah jadi. Perintah SQL ini biasanya untuk tujuan jahat. Entah ingin mensabotase database (menghapus seluruh tabelnya) atau bisa juga mengambil data-data yang ada, seperti password login atau data kartu kredit.

Teknik *SQL injection* memanfaatkan celah keamanan dari kode program yang kita buat. Sebagai contoh, perhatikan kode program berikut ini:

```
1 <?php
2 // buat data seolah-olah berasal dari form
3 $_POST['nama'] = "Riana Putria";
4
5 // buat query
6 $nama = $_POST['nama']; // ambil nilai dari form
7 $query = "SELECT * FROM mahasiswa WHERE nama = '{$nama}'";
8 $hasil_query = mysqli_query($koneksi, $query);
9 ?>
```

Disini saya merancang query yang kondisi **WHERE**-nya berasal dari variabel \$nama. Penulisan query seperti ini umum dilakukan. Biasanya isi dari variabel \$nama berasal dari form.



Apakah anda masih ingat kegunaan dari tanda kurung kurawal “{ “ dan “ } “? Tanda ini digunakan untuk memisahkan penulisan variabel di dalam string. Terutama jika kita mengakses element array dari dalam string.

Untuk query yang sederhana seperti diatas, tanda kurung tersebut tidak perlu ditulis, tapi akan memudahkan pembacaan query karena lebih jelas mana yang perintah SQL dan mana variabel PHP.

Jika kode program diatas diproses, query yang dijalankan adalah sebagai berikut:

```
SELECT * FROM mahasiswa WHERE nama = 'Riana Putria';
```

Tidak ada masalah. Tapi bagaimana jika ada yang menginput variabel \$nama seperti ini?

```
$_POST['nama'] = "Riana Putria'; DROP TABLE mahasiswa;"
```

Akibatnya, query yang akan dijalankan adalah sebagai berikut:

```
SELECT * FROM mahasiswa WHERE nama = 'Riana Putria'; DROP TABLE mahasiswa;
```

Saya yakin anda bisa menebak apa yang akan terjadi. Pada saat query diatas dijalankan, terdapat 2 perintah query, **SELECT** dan **DROP TABLE**. Inilah contoh dari *SQL Injection*!

Jadi bagaimana cara menanggulanginya? Terdapat berbagai cara. Yang paling sederhana kita bisa menggunakan fungsi **mysqli\_real\_escape\_string()**.



Jika anda mencoba menjalankan contoh kasus diatas menggunakan fungsi *mysqli\_query()*, PHP akan mengeluarkan error. Ini terjadi karena PHP sudah menambahkan fitur keamanan ke dalam fungsi *mysqli\_query()*.

Fungsi *mysqli\_query()* tidak bisa mengeksekusi 2 perintah query pada 1 kali pemanggilan. Walaupun begitu, anda tetap harus waspada dengan *SQL Injection*. Di versi yang lama (mysql extension), query seperti diatas bisa dijalankan.

## 23.23 Function **mysqli\_real\_escape\_string()**

Dari penjelasan tentang *SQL Injection*, kita bisa melihat bahwa hal tersebut disebabkan karena PHP salah memahami tanda kutip. Di dalam query tersebut, tanda kutip ( ' ) seharusnya bagian dari string, bukan sebagai perintah penutup query.

Kasus ini mirip ketika kita membuat string yang di dalamnya terdapat tanda kutip, seperti contoh berikut:

```
$kalimat = 'Hari ini adalah hari jum'at';
```

Bisakah anda menebak dimana letak salahnya? Benar, tanda kutip satu di hari "jum'at" akan diproses sebagai penutup string. Seharusnya tanda kutip tersebut merupakan bagian dari string. Solusi untuk masalah ini adalah dengan men-*escape* tanda kutip menggunakan karakter backslash ( \ ):

```
$kalimat = 'Hari ini adalah hari jum\'at';
```

Hal ini juga berlaku ketika kita membuat query SQL:

```
1 <?php
2 // buat query
3 $nama = "Sa'id Rahmat";
4 $query = "SELECT * FROM mahasiswa WHERE nama = '{$nama}'";
5
6 echo $query;
7 // SELECT * FROM mahasiswa WHERE nama = 'Sa'id Rahmat'
8
9 $hasil_query = mysqli_query($koneksi, $query);
10 // Query Error: 1064 - You have an error in your SQL syntax;
11 // check the manual that corresponds to your MariaDB server version
12 // for the right syntax to use near 'id Rahmat' at line 1
13 ?>
```

Ini terjadi karena tanda kutip di nama “sa'id” dianggap sebagai penutup perintah query. Solusinya adalah dengan menambahkan karakter escape:

```
$nama = "Sa\'id Rahmat";
```

Namun bagaimana jika variabel \$nama berasal dari form? Kita tentu tidak bisa melakukan ini secara manual. Inilah salah satu kegunaan dari fungsi `mysqli_real_escape_string()`.

Fungsi `mysqli_real_escape_string()` akan mengkonversi seluruh karakter yang dianggap “bermasalah” jika digunakan di dalam query MySQL. Selain tanda kutip, fungsi ini akan meng-escape karakter-karakter lain, yakni: NULL (ASCII 0), \n, \r, \, “, dan Control-Z.



PHP sebenarnya juga memiliki fungsi `addslashes()` untuk men-escape karakter seperti tanda kutip. Tapi khusus query MySQL, sebaiknya menggunakan fungsi `mysqli_real_escape_string()`.

Fungsi `mysqli_real_escape_string()` membutuhkan 2 argumen, yang pertama variabel koneksi MySQL, dan yang kedua berupa string yang ingin diesecape. Sekarang, saya bisa menulis query sebelumnya menjadi sebagai berikut:

```
1 <?php
2 // buat query
3 $nama = "Sa'id Rahmat";
4 $nama = mysqli_real_escape_string($koneksi, $nama);
5 $query = "SELECT * FROM mahasiswa WHERE nama = '{$nama}'";
6
7 echo $query;
8 // SELECT * FROM mahasiswa WHERE nama = 'Sa\'id Rahmat'
9 ?>
```

Fungsi `mysqli_real_escape_string` juga bisa digunakan untuk mencegah SQL Injection. Mari kita coba:

```
1 <?php
2 // buat data seolah-olah dari form
3 $_POST['nama'] = "Riana Putria'; DROP TABLE mahasiswa;";
4
5 // buat query
6 $nama = mysqli_real_escape_string($koneksi, $_POST['nama']);
7 $query = "SELECT * FROM mahasiswa WHERE nama = '{$nama}'";
8
9 echo $query;
10 // SELECT * FROM mahasiswa WHERE nama = 'Riana Putria\'';
11 // DROP TABLE mahasiswa;
12 ?>
```

Sekarang, string “Riana Putria”; `DROP TABLE mahasiswa;` akan diproses sebagai 1 nama yang panjang, bukan sebagai 2 buah query. Tanda kutip akan di-escape dan dianggap sebagai bagian dari string, bukan penanda akhir query.

Mungkin anda bertanya, seperti namanya fungsi `mysqli_real_escape_string()` hanya digunakan untuk memproteksi **string**. Bagaimana dengan tipe data seperti **integer**?

Biasanya untuk tipe data integer (dan float) tidak perlu diproteksi. Karena, untuk tipe angka kita tidak menggunakan tanda kutip di dalam penulisan query. Jika ada yang coba menginput string, query SQL akan langsung error.

Tapi untuk jaga-jaga, kita bisa menambahkan perintah **type casting (int)** atau **(float)** untuk memastikan data yang diinput hanya berupa angka:

```

1 <?php
2 // buat data seolah-olah dari form
3 $_POST['ipk'] = 3.2 ."'; DROP TABLE mahasiswa';
4
5 // buat query
6 $ipk = (float) $_POST['ipk'] ; // type casting menjadi float
7 $query = "SELECT * FROM mahasiswa WHERE ipk = {$ipk}";
8
9 echo $query;
10 // SELECT * FROM mahasiswa WHERE ipk = 3.2
11 ?>

```

Disini string tambahan setelah angka 3.2 akan otomatis dibuang oleh type casting (**float**). Namun yang juga perlu diperhatikan, saya merancang query tanpa tanda kutip. Untuk tipe data angka, MySQL membolehkan penulisan dengan tanda kutip seperti berikut:

```
$query = "SELECT * FROM mahasiswa WHERE ipk = '{$ipk}'";
```

Tapi penulisan tanda kutip di `ipk = '{$ipk}'` membuka peluang untuk **SQL Injection**. Jadi hindari menambahkan tanda kutip untuk tipe data angka.

## 23.24 Mengenal Prepared Statement

**Prepared Statement** merupakan fitur lanjutan yang didukung oleh *mysqli extension*. Tujuan dari *prepared statement* adalah memisahkan perintah query SQL dengan variabel PHP. Sehingga jika kita menggunakan *prepared statement*, otomatis sudah ‘kebal’ terhadap SQL Injection.

Karena termasuk materi lanjutan, disini saya hanya akan membahasnya sekilas. Jika anda butuh panduan lebih rinci tentang *prepared statement*, bisa membacanya di referensi lain seperti [PHP Manual](#)<sup>5</sup>.

Proses pembuatan prepared statements membutuhkan 3 langkah: **Prepared**, **Bind**, dan **Execute**.

---

<sup>5</sup><http://php.net/manual/en/mysqli.quickstart.prepared-statements.php>

Apabila dibandingkan dengan cara “biasa” yang sudah kita pelajari, prepared statement memecah fungsi `mysqli_query()` menjadi 3 fungsi :

- `mysqli_prepare()`: mempersiapkan query awal (*prepared*).
- `mysqli_stmt_bind_param()`: menghubungkan data ke query awal (*bind*).
- `mysqli_stmt_execute()`: jalankan query (*execute*).

Sebagai contoh, saya ingin mengkonversi query berikut menjadi *prepared statement*:

```

1 $nama = "Riana Putria";
2 $ipk = 3.10;
3 $query = "SELECT * FROM mahasiswa WHERE nama='{$nama}' AND ipk={$ipk}";
4 $hasil_query = mysqli_query($koneksi, $query);

```

Disini saya ingin mencari data mahasiswa dengan nama “**Riana Putria**” dan memiliki ipk **3.10**.

## Proses Pertama: Prepared

Pada proses **prepared**, kita menulis query yang akan dijalankan, tetapi tanpa ‘data’. Bagian dimana data biasanya ditulis digantikan dengan tanda tanya (?), seperti contoh berikut:

```
"SELECT * FROM mahasiswa WHERE nama = ? AND ipk = ?";
```

Untuk menjalankan perintah diatas, kita menggunakan fungsi `mysqli_prepare()`:

```

1 <?php
2   $query = "SELECT * FROM mahasiswa WHERE nama = ? AND ipk = ?";
3   $stmt = mysqli_prepare($koneksi, $query);
4 ?>

```

Fungsi `mysqli_prepare()` membutuhkan 2 argumen, yakni variabel hasil pemanggilan fungsi `mysqli_connect()`, dan string query yang akan dijalankan.

Hasil pemanggilan fungsi `mysqli_prepare()` ini selanjutnya disimpan ke dalam variabel `$stmt`. Variabel ini akan digunakan di dalam proses berikutnya (**bind** dan **execute**). Anda bebas jika ingin menggunakan nama variabel lain.

Secara internal, ketika fungsi `mysqli_prepare()` dijalankan, query tersebut sudah langsung dikirim ke MySQL Server. Selanjutnya MySQL Server akan menunggu proses **bind**.

## Proses Kedua: Bind

Proses **bind** bertujuan untuk mengisi tanda tanya “?” dengan data yang sebenarnya. Untuk ini kita menggunakan fungsi **mysqli\_stmt\_bind\_param()**.

Fungsi *mysqli\_stmt\_bind\_param()* membutuhkan sekurang-kurangnya 3 argumen:

- Argumen pertama diisi dengan variabel hasil fungsi **mysqli\_prepare()**. Dimana dalam contoh sebelumnya saya menggunakan variabel `$stmt`.
- Argumen kedua adalah string yang menunjukkan jenis tipe data argumen ketiga, yakni data yang akan diinput ke dalam query (saya akan membahas isi argumen kedua ini sesaat lagi).
- Argumen ketiga diisi dengan variabel data yang akan menggantikan tanda “?”. Jika data yang digantikan ada 2, maka ditempatkan menjadi argumen keempat, demikian seterusnya.

Melanjutnya contoh kita, berikut pemanggilan fungsi *mysqli\_stmt\_bind\_param()*:

```

1 <?php
2 // siapkan data
3 $nama = "Riana Putria";
4 $ipk = 3.10;
5
6 // buat query: prepare
7 $query = "SELECT * FROM mahasiswa WHERE nama = ? AND ipk = ?";
8 $stmt = mysqli_prepare($koneksi, $query);
9
10 // hubungkan data dengan prepared statements :bind
11 mysqli_stmt_bind_param($stmt, "sd", $nama, $ipk);
12 ?>

```

Argumen kedua dari fungsi **mysqli\_stmt\_bind\_param()** butuh pembahasan tersendiri. Argumen ini diisi dengan string yang merupakan inisial tipe data argumen ketiga dan seterusnya.

String “sd” dalam contoh diatas, berarti saya mempersiapkan tipe data “**string** dan **float**”. Kenapa harus “sd”? Karena data yang diinput adalah `$nama` (bertipe string) dan `$ipk` (bertipe float).

Berikut “inisial” dari jenis tipe data yang digunakan:

- **i** = variabel bertipe integer
- **d** = variabel bertipe double
- **s** = variabel bertipe string
- **b** = variabel bertipe blob (binary)

Contoh lain, perhatikan *prepared query* berikut:

```
SELECT * FROM mahasiswa WHERE nama = ? AND tempat_lahir = ? AND ipk = ?
```

Bisakah anda merancang fungsi `mysqli_stmt_bind_param()` dari query ini?

Berikut contohnya:

```
mysqli_stmt_bind_param($stmt, "ssd", $nama, $tempat_lahir, $ipk);
```

Inisial dari argumen kedua harus sama dengan jumlah data. Disini saya memiliki 3 data, maka string nya menjadi “ssd” (*string, string, float*).

## Proses Ketiga: Execute

Setelah proses **bind** selesai, langkah berikutnya adalah menjalankan query (**execute**) dengan menggunakan fungsi `mysqli_stmt_execute()`. Fungsi ini membutuhkan 1 buah argumen, yakni variabel hasil pemanggilan fungsi `mysqli_prepare()`:

```
1 <?php
2 // jalankan query: execute
3 $sukses = mysqli_stmt_execute($stmt);
4 ?>
```

Fungsi `mysqli_stmt_execute()` akan mengembalikan nilai TRUE atau FALSE bergantung apakah query sukses di jalankan atau terdapat error. Ini bisa kita jadikan patokan untuk menampilkan pesan error:

```
1 <?php
2 // cek apakah query berhasil
3 if (!$sukses){
4     die ("Query Error: ".mysqli_errno($koneksi).
5          " - ".mysqli_error($koneksi));
6 }
7 ?>
```

## Menampilkan Data Prepared Statement

Setelah proses **prepared**, **bind**, dan **execute** selesai, “kerjaan” kita belum habis. Untuk menampilkan data, membutuhkan fungsi `mysqli_stmt_get_result()`. Fungsi ini digunakan untuk mengambil data hasil query.

Hasil dari fungsi ini adalah variabel *resources* yang sama seperti hasil fungsi `mysqli_query()`. Berikut contohnya:

```
1 <?php
2 // ambil hasil query
3 $hasil_query = mysqli_stmt_get_result($stmt);
4
5 //tampilkan data
6 while($data = mysqli_fetch_row($hasil_query)) {
7     echo "$data[0] $data[1] $data[2] $data[3] $data[4] $data[5] $data[6]";
8 }
9 ?>
```

Berikut kode program lengkap cara menggunakan *prepared statement* untuk menampilkan data dari tabel mahasiswa:

```
1 <?php
2 // buat koneksi dengan database mysql
3 $koneksi = mysqli_connect("localhost", "root", "", "universitas");
4
5 //periksa koneksi, tampilkan pesan kesalahan jika gagal
6 if (!$koneksi){
7     die ("Koneksi dengan database gagal: ".mysqli_connect_errno().
8          " - ".mysqli_connect_error());
9 }
10
11 // siapkan data
12 $nama = "Riana Putria";
13 $ipk = 3.10;
14
15 // buat query: prepare
16 $query = "SELECT * FROM mahasiswa WHERE nama = ? AND ipk = ?";
17 $stmt = mysqli_prepare($koneksi, $query);
18
19 // hubungkan data dengan prepared statements :bind
20 mysqli_stmt_bind_param($stmt, "sd", $nama, $ipk);
21
22 // jalankan query: execute
23 $sukses = mysqli_stmt_execute($stmt);
24
25 // cek apakah query berhasil
26 if (!$sukses){
27     die ("Query Error: ".mysqli_errno($koneksi).
28          " - ".mysqli_error($koneksi));
29 }
30
31 // ambil hasil query
32 $hasil_query = mysqli_stmt_get_result($stmt);
```

```

34 //tampilkan data
35 while($data = mysqli_fetch_row($hasil_query))
36 {
37     echo "$data[0] $data[1] $data[2] $data[3] $data[4] $data[5] $data[6]";
38 }
39
40 // bebaskan memory
41 mysqli_stmt_free_result($stmt);
42
43 // tutup statement
44 mysqli_stmt_close($stmt);
45
46 // tutup koneksi dengan database mysql
47 mysqli_close($koneksi);
48 ?>

```

Fungsi yang belum saya bahas adalah:

**mysqli\_stmt\_free\_result()**: ini sama dengan fungsi **mysqli\_free\_result()**, hanya saja khusus untuk prepared statement. Fungsinya untuk membebaskan memory yang terpakai untuk menampung data hasil query **SELECT**.

**mysqli\_stmt\_close()**: digunakan untuk menutup sesi *prepared statement*. Fungsi ini sifatnya opsional dan boleh tidak dituliskan, karena PHP otomatis menghapus koneksi begitu halaman selesai diproses.

## Menginput Data Menggunakan Prepared Statement

Kode program sebelumnya digunakan untuk menampilkan data. Bagaimana dengan menginput data? Caranya hampir sama. Berikut kode program lengkap dimana saya menginput 1 data tambahan ke dalam tabel mahasiswa:

```

1 <?php
2 // buat koneksi dengan database mysql
3 $koneksi = mysqli_connect("localhost", "root", "", "universitas");
4
5 //periksa koneksi, tampilkan pesan kesalahan jika gagal
6 if(!$koneksi){
7     die ("Koneksi dengan database gagal: ".mysqli_connect_errno().
8          " - ".mysqli_connect_error());
9 }
10
11 // siapkan data
12 $nim = "14009037";
13 $nama = "Luna Priscila";
14 $tmp_lhr = "Jakarta";

```

```
15 $tgl_lhr = "1996-04-22";
16 $fakultas = "Sastra";
17 $jurusan = "Bahasa Inggris";
18 $ipk = 3.9;
19
20 // buat query: prepare
21 $query = "INSERT INTO mahasiswa VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
22 $stmt = mysqli_prepare($koneksi, $query);
23
24 // hubungkan data dengan prepared statements :bind
25 mysqli_stmt_bind_param($stmt, "ssssssi",
26 $nim, $nama, $tmp_lhr, $tgl_lhr, $fakultas, $jurusan, $ipk );
27
28 // jalankan query: execute
29 $sukses = mysqli_stmt_execute($stmt);
30
31 // cek apakah query berhasil
32 if (!$sukses) {
33     die('Query Error : '.mysqli_errno($koneksi). ' - '
34     .mysqli_error($koneksi));
35 }
36 else {
37     echo "Penambahan ".mysqli_stmt_affected_rows($stmt)
38     ." data berhasil<br />";
39 }
40
41 // tutup statement
42 mysqli_stmt_close($stmt);
43
44 // tutup koneksi dengan database mysql
45 mysqli_close($koneksi);
46 ?>
```

Saya tidak akan menjelaskan lagi kode program yang ada karena sangat mirip dengan kode program kita sebelumnya. Silahkan anda pelajari secara perlahan (itung-itung bahan latihan).

## 23.25 Mysql Extension

Diawal bab ini saya telah menjelaskan bahwa **mysql extension** sudah berstatus *deprecated*, bahkan di PHP 7 sudah dihapus sama sekali.

Walaupun begitu, masih banyak kode program yang menggunakan **mysql extension**, terutama kode program yang dibuat di tahun 2014 kebawah. Ini juga termasuk tutorial dan buku PHP yang belum update. Tidak jarang saya ditanya tentang cara menkonversi *mysql extension* menjadi *mysqli extension*.

Secara umum, fungsi-fungsi yang digunakan oleh kedua extension ini sangat mirip. Perbedaannya hanya pada nama dan urutan argumen. Jika pada *mysqli extension* variabel `$koneksi` ditempatkan di argumen pertama, maka di dalam *mysql extension* variabel ini menjadi argumen terakhir dan bersifat opsional.

Sebagai contoh, untuk menjalankan query MySQL, dengan **mysqli** kita menulis sebagai berikut:

```
$hasil_query = mysqli_query ($koneksi, $query);
```

Sedangkan di *mysql extension*, urutannya dibalik:

```
$hasil_query = mysqli_query ($query,$koneksi);
```

Dan biasanya di dalam *mysql extension*, penulisan variabel `$koneksi` ini boleh dibaikan:

```
$hasil_query = mysqli_query ($query);
```

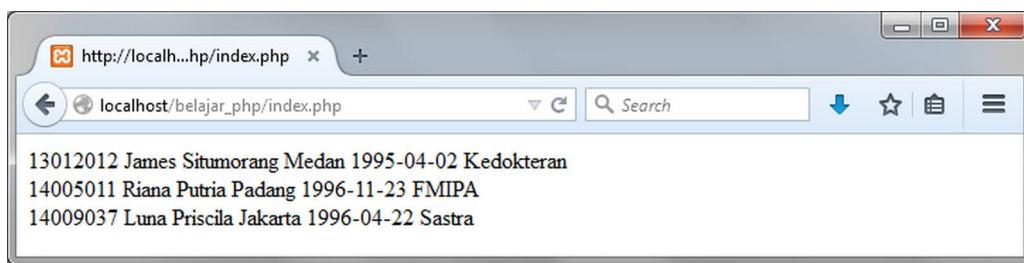
Berikut contoh kode program lengkap cara membuat koneksi, memilih database, menjalankan query dan menampilkan data menggunakan *mysql extension*. Jika anda menggunakan PHP 7 (XAMPP 7), kode program ini tidak akan bisa dijalankan:

```
1 <?php
2 //buat koneksi dengan MySQL
3 $koneksi = mysql_connect('localhost','root','');
4
5 //periksa koneksi, tampilkan pesan kesalahan jika gagal
6 if(!$koneksi){
7     die ("Koneksi dengan database gagal: ".mysql_connect_errno().
8          " - ".mysql_connect_error());
9 }
10
11 //pilih dan gunakan database universitas
12 $hasil_query = mysql_query('USE universitas',$koneksi);
13
14 //jalankan query
15 $query = "SELECT * FROM mahasiswa";
16 $hasil_query = mysql_query($query);
17
18 //tampilkan data
19 while ($row = mysql_fetch_array($hasil_query, MYSQL_NUM))
20 {
21     echo "$row[0] $row[1] $row[2] $row[3] $row[4]";
22     echo "<br />";
23 }
24
```

```

25 //periksa query, tampilkan pesan kesalahan jika gagal
26 if(!$hasil_query){
27     die ("Query gagal dijalankan: ".mysql_errno($koneksi).
28          " - ".mysql_error($koneksi));
29 }
30
31 // bebaskan memory
32 mysql_free_result($hasil_query);
33
34 // tutup koneksi dengan database mysql
35 mysql_close($koneksi);
36 ?>

```



Gambar: Hasil tabel mahasiswa dari mysqli extension

Saya tidak akan membahas fungsi yang digunakan pada mysql extension ini. Saya yakin anda sudah bisa membaca maksud dari setiap fungsi ini. Jika butuh penjelasan yang lebih detail, anda bisa membaca tutorialnya di [duniaIlkom](#)<sup>6</sup>.

---

Dalam bab ini kita telah membahas berbagai hal terkait cara menghubungkan PHP dengan MySQL. Menurut saya, materi ini sudah mencover 90% penggunaan dasar fungsi-fungsi **mysqli extension**. Namun tentu saja masih ada fungsi-fungsi mysqli lain yang belum saya bahas. Anda bisa mempelajarinya dari PHP Manual.

Seperti yang juga sering saya tulis dalam pembahasan ini. Kemampuan dan pemahaman anda tentang MySQL dan cara menulis querinya sangat berperan. Misalkan saya belum membahas cara membuat proses pencarian di tabel mahasiswa. Untuk hal ini sebenarnya tinggal mengganti query MySQL menggunakan perintah **LIKE**.

Atau bagaimana jika ingin menampilkan 10 data mahasiswa yang memiliki IPK tertinggi? Inipun hanya tinggal membuat query SQL yang sesuai (**LIMIT** dan **ORDER BY**).

Dalam bab berikutnya saya akan membahas sebuah study case khusus yang sekaligus menutup buku PHP Uncover. Saya akan merancang sebuah aplikasi **CRUD** sederhana, dimana kita akan mengakses data di tabel mahasiswa secara interaktif, termasuk menggabungkannya dengan form HTML.

---

<sup>6</sup><http://www.duniaIlkom.com/tutorial-php-cara-membuat-koneksi-php-ke-database-mysql/>

# 24. Aplikasi CRUD – Sistem Informasi Kampusku

Bab ini merupakan bab terakhir dari buku **PHP Uncover**. Disini saya akan mencoba membuat sebuah aplikasi sederhana dengan menggabungkan berbagai konsep pemrograman PHP yang telah kita pelajari sepanjang buku ini.

Aplikasi yang akan saya rancang biasa disebut dengan **CRUD**, singkatan dari **Create, Read, Update** dan **Delete**. Istilah ini merujuk kepada apa yang akan diproses oleh aplikasi ini. Saya akan menampilkan data dari database (*READ*), membuat data (*CREATE*), mengedit data (*UPDATE*), dan menghapus data (*DELETE*).

Selain itu juga terdapat konsep **session** untuk membuat fitur *login* dan *logout*. Termasuk proses validasi form dan berbagai teknik lainnya seperti *password hashing*.

Karena cukup kompleks, kode program yang diperlukan juga lumayan panjang. Saya hanya akan membahas dan menampilkan kode program yang penting-penting saja. Ini supaya buku **PHP Uncover** ini tidak terlalu tebal.

Anda bisa persilahkan untuk membaca dan membuka sendiri file ini di [belajar\\_php.zip](#), folder **bab\_24\_crud**. Terdapat 10 file PHP dan 1 file CSS. Di dalam setiap file saya sudah menyertakan banyak komentar terkait fungsi dari kode program tersebut. Jika anda sudah memahami materi dari bab-bab sebelumnya, saya yakin tidak akan terlalu sulit membaca kode-kode ini.

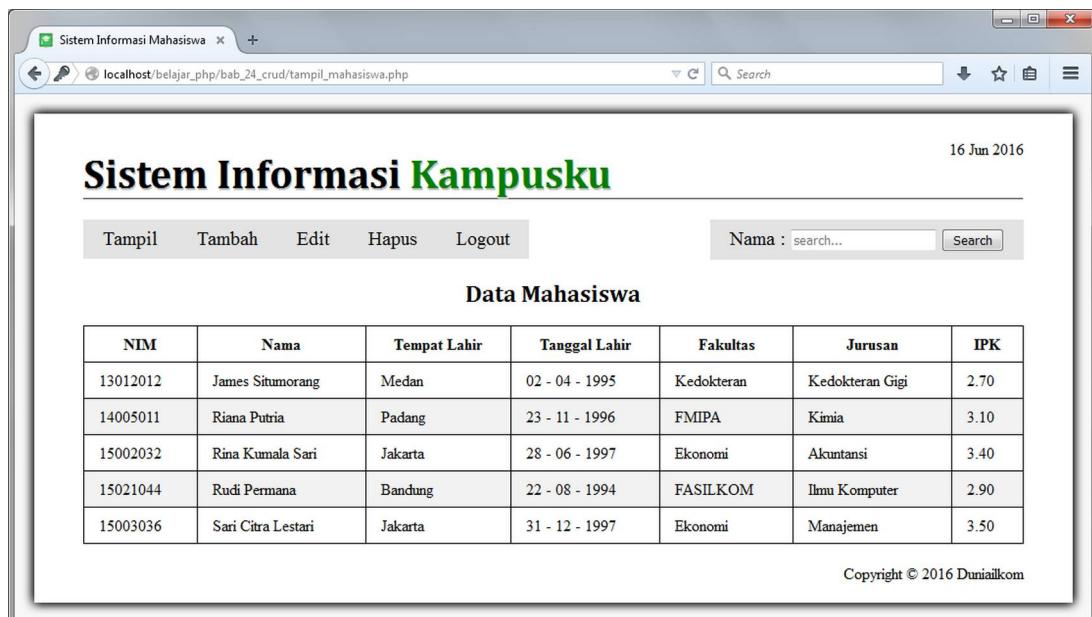
## 24.1 Tampilan Aplikasi Sistem Informasi Kampusku

Aplikasi yang akan saya rancang sebenarnya cukup sederhana, hanya terdiri dari 5 halaman tampilan: *halaman login, menampilkan data, menambah data, mengubah data, dan menghapus data*. Agar terkesan lebih menarik, saya menamakannya sebagai “**Sistem Informasi Kampusku**”.

Di dalam **Sistem Informasi Kampusku**, saya mengolah database *mahasiswa* yang sebenarnya sudah kita praktekkan dalam bab tentang koneksi PHP MySQL. Berikut tampilan kelima halaman tersebut:



Gambar: Halaman Login



Gambar: Halaman utama untuk menampilkan seluruh data mahasiswa

**Sistem Informasi Kampusku**

16 Jun 2016

Tampil    Tambah    Edit    Hapus    Logout

Nama :  Search

**Tambah Data Mahasiswa**

Mahasiswa Baru

NIM :  (Contoh: 12345678) (8 digit angka)

Nama :

Tempat Lahir :

Tanggal Lahir :  Januari  1996

Fakultas :  Kedokteran

Jurusan :

IPK :  (Contoh: 2.75) (angka desimal dipisah dengan karakter titik ".")

**Tambah Data**

Gambar: Form untuk menambah data mahasiswa baru

**Sistem Informasi Kampusku**

16 Jun 2016

Tampil    Tambah    Edit    Hapus    Logout

Nama :  Search

**Edit Data Mahasiswa**

NIM	Nama	Tempat Lahir	Tanggal Lahir	Fakultas	Jurusan	IPK	
13012012	James Situmorang	Medan	1995-04-02	Kedokteran	Kedokteran Gigi	2.70	<b>Edit</b>
14005011	Riana Putria	Padang	1996-11-23	FMIPA	Kimia	3.10	<b>Edit</b>
15002032	Rina Kumala Sari	Jakarta	1997-06-28	Ekonomi	Akuntansi	3.40	<b>Edit</b>
15021044	Rudi Permana	Bandung	1994-08-22	FASILKOM	Ilmu Komputer	2.90	<b>Edit</b>
15003036	Sari Citra Lestari	Jakarta	1997-12-31	Ekonomi	Manajemen	3.50	<b>Edit</b>

Gambar: Halaman untuk memilih data mahasiswa yang ingin di-edit

**Edit Data Mahasiswa**

Mahasiswa Baru

NIM :	14005011	(tidak bisa diubah di menu edit)	
Nama :	Riana Putria		
Tempat Lahir :	Padang		
Tanggal Lahir :	01	Nopember	1990
Fakultas :	FMIPA		
Jurusan :	Kimia		
IPK :	3.10	(angka desimal dipisah dengan karakter titik ".")	

**Update Data**

Gambar: Form untuk edit data mahasiswa

NIM	Nama	Tempat Lahir	Tanggal Lahir	Fakultas	Jurusan	IPK	Hapus
13012012	James Situmorang	Medan	1995-04-02	Kedokteran	Kedokteran Gigi	2.70	**Hapus**
14005011	Riana Putria	Padang	1996-11-23	FMIPA	Kimia	3.10	**Hapus**
15002032	Rina Kumala Sari	Jakarta	1997-06-28	Ekonomi	Akuntansi	3.40	**Hapus**
15021044	Rudi Permana	Bandung	1994-08-22	FASILKOM	Ilmu Komputer	2.90	**Hapus**
15003036	Sari Citra Lestari	Jakarta	1997-12-31	Ekonomi	Manajemen	3.50	**Hapus**

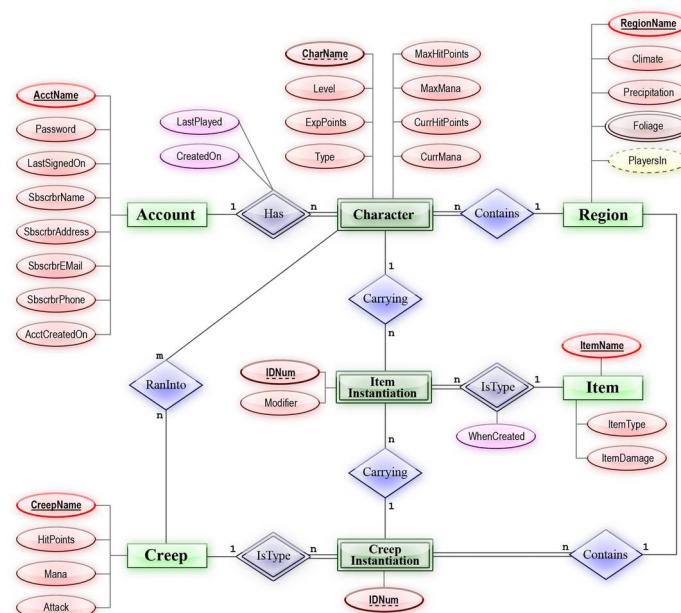
Gambar: Halaman untuk memilih data mahasiswa yang ingin dihapus

Untuk membuat aplikasi seperti ini saya memecahnya kedalam 11 file. Kita akan membahasnya satu per satu.

## 24.2 Membuat dan Mengisi Tabel Mahasiswa (generate.php)

Hal pertama yang mesti dilakukan dalam setiap perancangan program adalah membuat struktur database. Untuk aplikasi yang kompleks, kadang kita perlu membuat diagram ERD (*Entity Relationship Diagram*).

ERD digunakan untuk merancang hubungan antar tiap-tiap tabel di dalam database. Bisa saja anda butuh 10 tabel yang saling berhubungan satu sama lain. Di dalam ERD inilah alur datanya akan terlihat dan menjadi panduan kita dalam merancang kode program. Jika anda tertarik mempelajari ERD ini, bisa mencari buku tentang teori database.



Gambar: Contoh diagram ERD, sumber: wikipedia

Dalam aplikasi **Sistem Informasi Kampusku**, saya hanya menggunakan 2 buah tabel: **tabel mahasiswa**, dan **tabel admin**.

**Tabel mahasiswa** adalah tabel utama yang berisi data mahasiswa. Tabel ini sama persis dengan yang saya gunakan pada pembahasan bab sebelumnya. Tabel mahasiswa terdiri dari 7 kolom: *nim, nama, tempat\_lahir, tanggal\_lahir, fakultas, jurusan, dan ipk*.

**Tabel admin** adalah tabel ‘bantu’ yang digunakan untuk menyimpan *username* dan *password* dari user yang berhak mengakses aplikasi. Tabel ini memiliki 2 kolom: *username* dan *password*. Pada saat login, PHP akan mengecek apakah *username* dan *password* yang diinput sesuai dengan isi tabel ini atau tidak. Jika iya, berikan hak akses. Jika tidak, tampilkan pesan kesalahan.

Kedua tabel ini (**mahasiswa** dan **admin**) bisa saja kita buat manual dari *cmd windows* atau menggunakan *PhpMyadmin*. Akan tetapi saya akan membuat sebuah script untuk menggenerate kedua tabel ini berserta isinya. Script ini saya simpan ke dalam file *generate.php*.

File *generate.php* berisi kode program yang akan menjalankan 5 langkah berikut:

- Membuat database “kampusku” (jika belum ada).

- Membuat tabel mahasiswa.
- Mengisi data tabel mahasiswa.
- Membuat tabel admin.
- Mengisi data tabel admin.

Anda bisa membuka file `belajar_php/bab_24_crud/generate.php` untuk melihat kode program lengkapnya.

Mari kita bahas. Pada kode program paling atas, saya membuat koneksi PHP MySQL menggunakan fungsi `mysqli_connect()`:

```

1  <?php
2  // buat koneksi dengan database mysql
3  $dbhost = "localhost";
4  $dbuser = "root";
5  $dbpass = "";
6  $link = mysqli_connect($dbhost,$dbuser,$dbpass);
7
8  //periksa koneksi, tampilkan pesan kesalahan jika gagal
9  if(!$link){
10     die ("Koneksi dengan database gagal: ".mysqli_connect_errno().
11          " - ".mysqli_connect_error());
12 }
13 ?>

```

Bisakah anda lihat perbedaan dari fungsi `mysqli_connect()` yang saya gunakan dengan yang saya bahas pada bab sebelumnya? Kali ini saya menggunakan variabel `$link` untuk menampung hasil koneksi. Variabel inilah yang akan kita pakai sepanjang kode program.

Selain itu, perhatikan cara pemanggilan fungsi `mysqli_connect()`. Saya hanya menggunakan 3 argumen: alamat server MySQL (*localhost*), username (*root*), dan password (*tanpa password*). Saya belum memilih jenis database yang digunakan (yang biasanya ada di argumen ke empat). Ini karena saya akan membuat database baru dengan kode program selanjutnya:

```

1  <?php
2  //buat database kampusku jika belum ada
3  $query = "CREATE DATABASE IF NOT EXISTS kampusku";
4  $result = mysqli_query($link, $query);
5
6  if(!$result){
7      die ("Query Error: ".mysqli_errno($link).
8           " - ".mysqli_error($link));
9  }
10 else {
11     echo "Database <b>'kampusku'</b> berhasil dibuat... <br>";
12 }
13 ?>

```

Perintah diatas saya gunakan untuk membuat database “kampusku”. Query yang digunakan adalah: **CREATE DATABASE IF NOT EXISTS kampusku**. Maksudnya, buat database kampusku hanya apabila tidak ada (*if not exists*). Query ini saya jalankan dengan fungsi **mysqli\_query()**.

Apabila query tersebut berhasil dijalankan (dan tidak error), tampilkan pesan ke web browser: *Database 'kampusku' berhasil dibuat...*

Setelah database tersedia, berikutnya kita akan pilih database ini dengan perintah berikut:

```

1 <?php
2     //pilih database kampusku
3     $result = mysqli_select_db($link, "kampusku");
4
5     if (!$result){
6         die ("Query Error: ".mysqli_errno($link).
7             " - ".mysqli_error($link));
8     }
9     else {
10        echo "Database 'kampusku' berhasil dipilih... <br>";
11    }
12 ?>

```

Saya menggunakan fungsi yang belum pernah kita bahas sampai saat ini: **mysqli\_select\_db()**. Sesuai dengan namanya, fungsi **mysqli\_select\_db()** digunakan untuk memilih database yang akan digunakan. Argumen pertama fungsi ini berupa variabel koneksi **\$link**, dan argumen kedua adalah nama database yang ingin dipilih (bertipe string). Teknik seperti ini juga bisa anda gunakan untuk menukar database sepanjang kode program PHP.

Dalam bab sebelumnya kita tidak memerlukan fungsi ini karena database sudah dipilih langsung dari pemanggilan fungsi **mysqli\_connect()**. Yakni sebagai argumen dari fungsi **mysqli\_connect()**.

Setelah koneksi dibuat dan database dipilih, kita akan membuat serta mengisi tabel **mahasiswa**. Kode programnya sendiri tidak ada hal yang baru:

```

1 <?php
2     // cek apakah tabel mahasiswa sudah ada. jika ada, hapus tabel
3     $query = "DROP TABLE IF EXISTS mahasiswa";
4     $hasil_query = mysqli_query($link, $query);
5
6     if (!$hasil_query){
7         die ("Query Error: ".mysqli_errno($link).
8             " - ".mysqli_error($link));
9     }
10    else {
11        echo "Tabel <b>'mahasiswa'</b> berhasil dihapus... <br>";
12    }

```

```

13
14 // buat query untuk CREATE tabel mahasiswa
15 $query = "CREATE TABLE mahasiswa (nim CHAR(8), nama VARCHAR(100), ";
16 $query .= "tempat_lahir VARCHAR(50), tanggal_lahir DATE, ";
17 $query .= "fakultas VARCHAR(50), jurusan VARCHAR(50), ";
18 $query .= "ipk DECIMAL(3,2), PRIMARY KEY (nim))";
19
20 $hasil_query = mysqli_query($link, $query);
21
22 if (!$hasil_query){
23     die ("Query Error: ".mysqli_errno($link).
24          " - ".mysqli_error($link));
25 }
26 else {
27     echo "Tabel <b>'mahasiswa'</b> berhasil dibuat... <br>";
28 }
29
30 // buat query untuk INSERT data ke tabel mahasiswa
31 $query = "INSERT INTO mahasiswa VALUES ";
32 $query .= "('14005011', 'Riana Putria', 'Padang', '1996-11-23', ";
33 $query .= "'FMIPA', 'Kimia', 3.1), ";
34 $query .= "('15021044', 'Rudi Permana', 'Bandung', '1994-08-22', ";
35 $query .= "'FASILKOM', 'Ilmu Komputer', 2.9), ";
36 $query .= "('15003036', 'Sari Citra Lestari', 'Jakarta', '1997-12-31', ";
37 $query .= "'Ekonomi', 'Manajemen', 3.5), ";
38 $query .= "('15002032', 'Rina Kumala Sari', 'Jakarta', '1997-06-28', ";
39 $query .= "'Ekonomi', 'Akuntansi', 3.4), ";
40 $query .= "('13012012', 'James Situmorang', 'Medan', '1995-04-02', ";
41 $query .= "'Kedokteran', 'Kedokteran Gigi', 2.7)";
42
43 $hasil_query = mysqli_query($link, $query);
44
45 if (!$hasil_query){
46     die ("Query Error: ".mysqli_errno($link).
47          " - ".mysqli_error($link));
48 }
49 else {
50     echo "Tabel <b>'mahasiswa'</b> berhasil diisi... <br>";
51 }
52 ?>

```

Terdapat 3 buah perintah query yang saya gunakan: hapus database mahasiswa jika sebelumnya sudah ada (**DROP TABLE IF EXISTS mahasiswa**), buat database (**CREATE TABLE mahasiswa**) dan tambahkan data (**INSERT INTO mahasiswa**).

Tabel mahasiswa ini memiliki 7 kolom yang sama persis seperti yang saya gunakan pada bab tentang koneksi PHP MySQL.

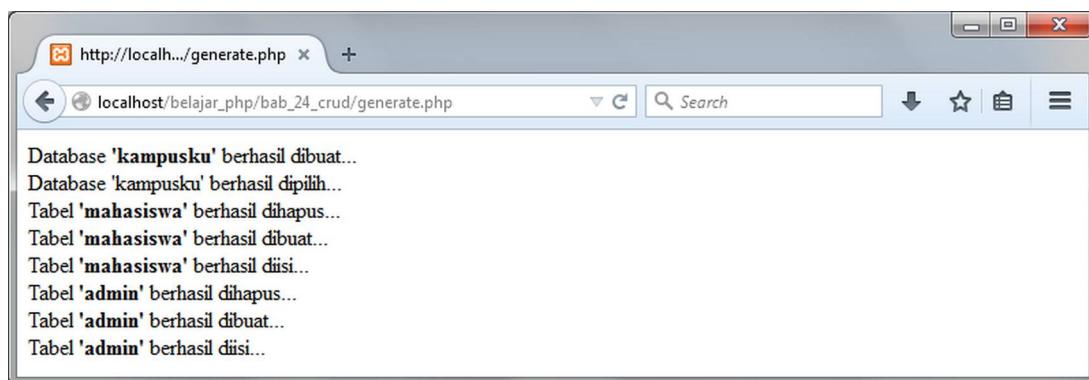
Langkah terakhir adalah membuat dan mengisi tabel **admin**. Berikut kode programnya:

```
1 <?php
2 // cek apakah tabel admin sudah ada. jika ada, hapus tabel
3 $query = "DROP TABLE IF EXISTS admin";
4 $hasil_query = mysqli_query($link, $query);
5
6 if (!$hasil_query){
7     die ("Query Error: ".mysqli_errno($link).
8          " - ".mysqli_error($link));
9 }
10 else {
11     echo "Tabel <b>'admin'</b> berhasil dihapus... <br>";
12 }
13
14 // buat query untuk CREATE tabel admin
15 $query = "CREATE TABLE admin (username VARCHAR(50), password CHAR(40))";
16 $hasil_query = mysqli_query($link, $query);
17
18 if (!$hasil_query){
19     die ("Query Error: ".mysqli_errno($link).
20          " - ".mysqli_error($link));
21 }
22 else {
23     echo "Tabel <b>'admin'</b> berhasil dibuat... <br>";
24 }
25
26 // buat username dan password untuk admin
27 $username = "admin123";
28 $password = sha1("rahasia");
29
30 // buat query untuk INSERT data ke tabel admin
31 $query = "INSERT INTO admin VALUES ('$username', '$password')";
32
33 $hasil_query = mysqli_query($link, $query);
34
35 if (!$hasil_query){
36     die ("Query Error: ".mysqli_errno($link).
37          " - ".mysqli_error($link));
38 }
39 else {
40     echo "Tabel <b>'admin'</b> berhasil diisi... <br>";
41 }
42
43 ?>
```

Sama seperti ketika membuat tabel mahasiswa, saya juga menjalankan 3 query: hapus tabel admin jika sudah ada (**DROP TABLE IF EXISTS admin**), buat tabel admin (**CREATE TABLE admin**), dan isi data untuk tabel admin (**INSERT INTO admin**).

Tabel admin saya rancang dengan 2 kolom: username dengan VARCHAR(50), dan password dengan VARCHAR(40). Jika anda perhatikan, saya menginput data ke dalam tabel admin dengan username: **admin123** dan password: **sha1("rahasia")**. Apa kegunaan dari fungsi **sha1()** ini? Saya akan bahas sesaat lagi.

Jika semua proses pembuatan tabel ini tidak ada masalah, anda akan melihat tampilan sebagai berikut:



Gambar: Database kampusku, tabel mahasiswa dan tabel admin berhasil di buat

Artinya, *database kampusku*, *tabel mahasiswa* dan *tabel admin* sudah berhasil di buat. Teknik seperti ini bisa anda terapkan untuk membuat website yang akan dijalankan di tempat lain. Kita tidak perlu membuat tabel secara manual, tapi cukup akses halaman ini, dan semua tabel sudah langsung terisi.

Andapun bebas ingin mengutak atik tabel yang ada. Jika nanti terjadi kesalahan, cukup akses kembali halaman *generate.php*, dan database akan kembali “di-reset” ke kondisi awal.

## 24.3 Mengenal Fungsi Hashing PHP

Salah satu konsep yang amat sangat penting yang perlu kita terapkan dalam setiap pembuatan aplikasi adalah: **jangan menyimpan password “apa adanya” di dalam database**.

Untuk tabel **admin**, saya berencara membuat username: **admin123**, dengan password: **rahasia**. Sebenarnya, saya bisa saja menginput kedua nilai ini langsung ke dalam tabel.

Tapi bayangkan jika website ini sudah “live” dan terdapat puluhan atau ratusan user yang sudah mendaftar. Saya sebagai pembuat aplikasi bisa dengan jelas melihat password apa saja yang digunakan oleh setiap user. Namun bisa dikatakan hal ini tidak terlalu bermasalah. Toh saya tidak akan menyalahgunakan password-password ini.

Tapi bagaimana seandainya situs saya di bobol dan isi data tabel ini di curi? Sang “*hacker*” yang membobol akan mendapatkan data yang sangat berharga: **password dari setiap user**. Yang membuatnya semakin parah, sebagian besar dari kita menggunakan password yang sama untuk setiap website, mulai dari email, facebook, twitter, instagram, dll.

Anda bisa bayangkan apa jadinya jika data ini jatuh ke tangan yang salah. Bisa-bisa user yang dirugikan akan menuntut situs saya karena dari sinilah password tersebut “bocor”.

Salah satu solusi menghindari hal ini adalah dengan tidak menyimpan password dalam bentuk aslinya. Teknik yang umum dipakai dikenal dengan teknik **hashing**.

Di dalam ilmu komputer, ada sebuah bidang ilmu yang sangat menarik, yakni **criptografi**. *Kriptografi* membahas bagaimana cara menyembunyikan sebuah pesan rahasia agar tidak bisa diketahui oleh pihak lain (pihak musuh).

Contoh kriptografi sederhana adalah dengan menukar sebuah huruf dengan huruf lain. Perhatikan ‘kalimat’ berikut:

```
tfeboh cfmbkbs qiq
```

Bisakah anda menebak artinya?

Kalimat diatas saya buat dengan menggeser 1 huruf abjad menjadi huruf berikutnya. Misalnya a menjadi b, b menjadi c, c menjadi b, dst. Dengan membalikkan proses ini, t menjadi s, f menjadi e, e menjadi d, dst. Kalimat diatas menjadi:

```
sedang belajar php
```

### *Inilah kriptografi!*

Untuk menerjemahkan sebuah pesan, kita butuh 2 hal: teks yang sudah di kodekan (yang sudah di enkripsi), dan cara mengkodekannya (sering disebut sebagai **key**). Pada contoh diatas, teks enkripsi-nya adalah: `tfeboh cfmbkbs qiq`. Sedangkan key-nya yakni menggeser 1 huruf abjad menjadi huruf berikutnya. pesan dan key ini harus diketahui agar pesan asli bisa dibaca.

Teknik kriptografi diatas disebut dengan enkripsi 2 arah (*two way encryption*). Dimana pesan yang sudah dienkripsi, bisa dikembalikan ke kondisi awal selama kita mengetahui key yang digunakan.

Di sisi lain, **hashing** adalah metode enkripsi satu arah (*one way encryption*). Disini, pesan yang sudah dienkripsi tidak bisa di kembalikan ke kondisi awal. Jadi, apa gunanya?

Ketika seseorang menemukan teks yang disimpan dari hasil hashing, dia tidak akan bisa mengembalikan teks tersebut ke kondisi awal, karena memang tidak ada key-nya. Tapi sebuah teks yang sama, akan menghasilkan nilai hashing yang sama pula.

Mari kita langsung masuk ke dalam praktek. PHP menyediakan berbagai fungsi terkait kriptografi dan hashing. Dua buah fungsi hashing yang masih sering digunakan adalah **md5()** dan **sha1()**. Keduanya menggunakan metode khusus untuk menghasilkan teks yang tidak mungkin di kembalikan ke dalam bentuk aslinya (sesuai dengan prinsip *hashing*).

Perbedaan dari kedua fungsi ini terletak dari cara atau algoritma *hashing* yang digunakan. Algoritma hashing ini dibuat oleh kalangan akademisi dengan penelitian dan rumus matematika yang cukup njelmet. Untungnya, kita tidak perlu mempelajari hal ini. Kita cukup memberikan teks yang akan dicari nilai hasingnya ke dalam fungsi **md5()** dan **sha1()** sebagai argumen.

Untuk contoh praktek, silahkan jalankan kode program berikut:

```

1 <?php
2   $password = "rahasia";
3   echo md5($password);
4   // ac43724f16e9241d990427ab7c8f4228
5 ?>

```

Karakter “acak” inilah yang merupakan *hasil hashing*. Fungsi **md5()** menghasilkan string sepanjang 32 digit yang terlihat acak. Berapapun panjang teks asli, hasilnya tetap 32 digit. Silahkan anda mencoba menukar string “rahasia” menjadi teks lain. String ini sebenarnya bukanlah acak, tapi menggunakan sebuah algoritma perhitungan khusus.

Selanjutnya, hasil hashing inilah yang kita simpan ke dalam database, bukan karakter password asli “rahasia”.

Dengan demikian, seseorang yang kebetulan bisa membobol sistem dan melihat tabel **admin**, tidak akan bisa menebak apa password asli dari string ac43724f16e9241d990427ab7c8f4228. Karena memang tidak ada cara mengembalikan hasil hashing ini ke karakter awal (setidaknya secara teori).

Tapi, bagaimana cara membuat program untuk mengecek apakah password yang diinput sesuai atau tidak?

Caranya sangat mudah. Kita tinggal membandingkan hasil yang ada di database dengan hasil fungsi **md5()** dari inputan user. Seperti contoh berikut:

```

1 <?php
2   // misalkan data ini berasal dari form
3   $_POST["password"] = "rahasia";
4   $password = $_POST["password"];
5
6   // ambil data password yang ada di database
7   $pass_database="ac43724f16e9241d990427ab7c8f4228";
8
9   if (md5($password) == $pass_database) {
10     echo "Password sesuai";
11   }
12   else {
13     echo "Password anda salah!";
14   }
15 ?>

```

Beginilah cara kita men-cek apakah inputan password user sesuai atau tidak. Khusus untuk sistem informasi kampusku, pengecekan ini akan dilakukan nanti di halaman **login.php**.

## Pilih Hashing dengan **md5()** atau **sha1()** ?

Fungsi hashing **md5()** dulunya sangat populer di gunakan. Akan tetapi jika anda mengikuti forum-forum terkait security dan keamanan, algoritma hashing md5 dianggap tidak lagi memadai.

Ini juga di jelaskan di [PHP Manual](#)<sup>1</sup>: *It is not recommended to use this function to secure passwords, due to the fast nature of this hashing algorithm.*

Semakin cepatnya kemampuan komputer modern, memungkinkan seseorang membobol keamanan algoritma **md5** (yang sebelumnya dianggap mustahil). Juga terdapat [penelitian](#)<sup>2</sup> yang menyebutkan beberapa kelemahan dari algoritma hashing **md5**.

Sebagai alternatif, anda bisa menggunakan fungsi **sha1()**. Cara penggunaan fungsi **sha1()** sama persis seperti **md5()**. Bedanya, jika **md5()** menghasilkan string sepanjang 32 karakter, **sha1()** menghasilkan string dengan panjang 40 karakter.

Untuk penjelasan lebih lanjut tentang perbedaan algoritma **md5** dengan **sha1**, anda bisa baca di [crypto.stackexchange.com](#)<sup>3</sup>.

## Algoritma hashing **md5()** dan **sha1()** pun sebenarnya belum cukup

Sekali lagi, seiring dengan berkembangnya teknologi komputasi, algoritma hashing **md5()** dan **sha1()** pun bisa dianggap belum cukup aman. Berbagai cara terus dikembangkan untuk membobol sistem ini.

Sebagai contoh, silahkan anda search kode berikut di google:

19a66b38bb79d4d31b58c41865ccd2c2

String diatas adalah hasil hashing dari sebuah password dengan algoritma md5. Bagaimana hasilnya?

---

<sup>1</sup><http://php.net/manual/en/function.md5.php>

<sup>2</sup><http://stackoverflow.com/questions/401656/secure-hash-and-salt-for-php-passwords>

<sup>3</sup><http://crypto.stackexchange.com/questions/18612/how-is-sha1-different-from-md5>

Google search results for the query '19a66b38bb79d4d31b58c41865ccd2c2'. The results page shows a link to 'Previous 10 Pages - Requested MD5 Hash queue' on www.md5this.com, which contains the plain text 'belajar'.

Gambar: Password asli ketahuan di google

Yup, anda bisa melihat bahwa karakter asli dari string tersebut adalah “belajar”. Inilah kode program yang sebelumnya saya jalankan:

```

1 <?php
2   $password = "belajar";
3   echo md5($password);
4   // 19a66b38bb79d4d31b58c41865ccd2c2
5 ?>

```

Bagaimana dengan algoritma **sha1** ? Silahkan lakukan hal yang sama dengan string berikut:

96e79218965eb72c92a549dd5a330112

What is the Russian word for uncover? - Word Hippo  
[www.wordhippo.com/.../russian-word-for-256c170dd63cf899...](https://www.wordhippo.com/.../russian-word-for-256c170dd63cf899...) Terjemahkan laman ini  
 How do you say 'uncover' in Russian? Here's a list of words you may be looking for.

Vorherige - HashDecryption.com - Passwords Recovery Online  
<https://hashdecryption.com/s/sha1/2254> Terjemahkan laman ini  
 ... socianize, 256b0837592ee4703d893c9f2e77fd709c05b, unsounding.  
 256c170dd63cf8990d6ffde869440d03e74de07c, uncover.

uncover - word spelling in wordbook - Starting page  
[abc-wordbook.com/word/uncover](http://abc-wordbook.com/word/uncover) Terjemahkan laman ini  
 The MD5 hash is c9b57e3c7f1ab075ac1b660f3969a106 and the SHA1 checksum is  
 256c170dd63cf8990d6ffde869440d03e74de07c. The dialable telephone ...

uncover - informations about word  
[en.wordlist.eu/word/uncover](http://en.wordlist.eu/word/uncover) Terjemahkan laman ini  
 Proper varieties of 'uncover', definitions and crossword clues. Scrabble score, anagrams and similar words. What is the uncover?

Gambar: Algoritma sha1 juga bobol...

Berikut kode asli yang saya jalankan:

```

1  <?php
2      $password = "uncover";
3      echo sha1($password);
4      // 19a66b38bb79d4d31b58c41865cccd2c2
5  ?>

```

Kenapa hal ini bisa terjadi? Sebenarnya bukan kelemahan dari algoritma md5 maupun sha1, tapi lebih disebabkan cara penggunaan teknik hashing secara umum.

Setiap fungsi hashing selalu menghasilkan string dengan panjang karakter yang sama. Dengan demikian saya bisa membuat sebuah tabel yang berisi “kata” password, dan nilai hashingnya, seperti berikut ini:

password	md5 hashing
admin	21232f297a57a5a743894a0e4a801fc3
123456	e10adc3949ba59abbe56e057f20f883e
qwerty	d8578edf8458ce06fbc5bb76a58c5ca4
rahasia	ac43724f16e9241d990427ab7c8f4228
111111	96e79218965eb72c92a549dd5a330112

Tabel diatas di kenal juga sebagai **rainbow tabel**, yakni tabel yang berisi hasil dari setiap fungsi hashing. Dengan kemampuan processor yang makin tinggi, 1 juta baris tabel bisa digenerate dengan cukup cepat. Saya tinggal mencocokkan apakah karakter hashing ini ada di tabel atau tidak.

## Jadi, apakah `md5()` dan `sha1()` masih layak digunakan?

Jika anda ingin membuat situs “live” yang akan dikunjungi oleh banyak orang dan menyimpan data sensitif seperti nomor kartu kredit, algoritma `md5()` dan `sha1()` ini tidak lagi mencukupi.

Salah satu solusi untuk menghindari masalah ini adalah dengan memaksa user menggunakan password yang tidak umum dan bukan berasal dari kata yang ada di kamus. Password seperti “*rahasia*”, “*belajar*”, “*kampus*”, dan “*harian*” sangat mungkin ada di dalam rainbow tabel. Tetapi password seperti “*4nd1kh@*” cukup sulit di bobol (mungkin belum ada di rainbow tabel).

Alternatif lain dengan melakukan *hashing* beberapa kali:

```

1 <?php
2   $password = "belajar";
3   echo md5(sha1(md5($password)));
4   // 033fcf42a737e201019715aed846cb0e
5 ?>
```

Sekarang, string `033fcf42a737e201019715aed846cb0e` tidak ada di google (atau mungkin lebih tepatnya: belum ada).

Pembahasan tentang keamanan web seperti ini bisa menjadi sangat panjang dan cukup rumit. Buku PHP Uncover adalah buku pengantar pemrograman PHP untuk pemula, jadi saya tidak akan membahas hal ini lebih jauh lagi. Jika anda berminat, bisa browsing atau cari buku yang fokus membahas tentang keamanan web.

Dalam *sistem informasi kampusku*, saya tetap menggunakan fungsi `sha1()` untuk menyimpan password. Jika anda mempelajari PHP sebagai hobi atau untuk tugas akademik seperti skripsi, teknik hashing seperti ini saya rasa sudah cukup layak dipakai.



Sebenarnya, PHP versi 5.5 keatas sudah menyediakan alternatif metode hashing yang lebih baik melalui fungsi `password_hash()`<sup>4</sup>.

Tapi karena keterbatasan tempat dan saya tidak mau membuat anda lebih pusing lagi membaca teori kriptografi seperti *salt*, *algoritma blowfish*, dll, saya tidak akan membahas disini. Materi ini akan saya siapkan di buku PHP lanjutan nanti.

## 24.4 Membuat File Koneksi Global (`connection.php`)

Setelah membahas sekilas tentang teori hashing, mari kita kembali ke sistem informasi kampusku.

File selanjutnya yang akan saya bahas adalah `connection.php`. File ini merupakan file khusus untuk membuat koneksi ke database MySQL. Berikut kode program dari `connection.php`:

---

<sup>4</sup><http://php.net/manual/en/function.password-hash.php>

```

1 <?php
2 // buat koneksi dengan database mysql
3 $dbhost = "localhost";
4 $dbuser = "root";
5 $dbpass = "";
6 $dbname = "kampusku";
7 $link = mysqli_connect($dbhost,$dbuser,$dbpass,$dbname);
8
9 //periksa koneksi, tampilkan pesan kesalahan jika gagal
10 if(!$link){
11     die ("Koneksi dengan database gagal: ".mysqli_connect_errno().
12          " - ".mysqli_connect_error());
13 }
14 ?>

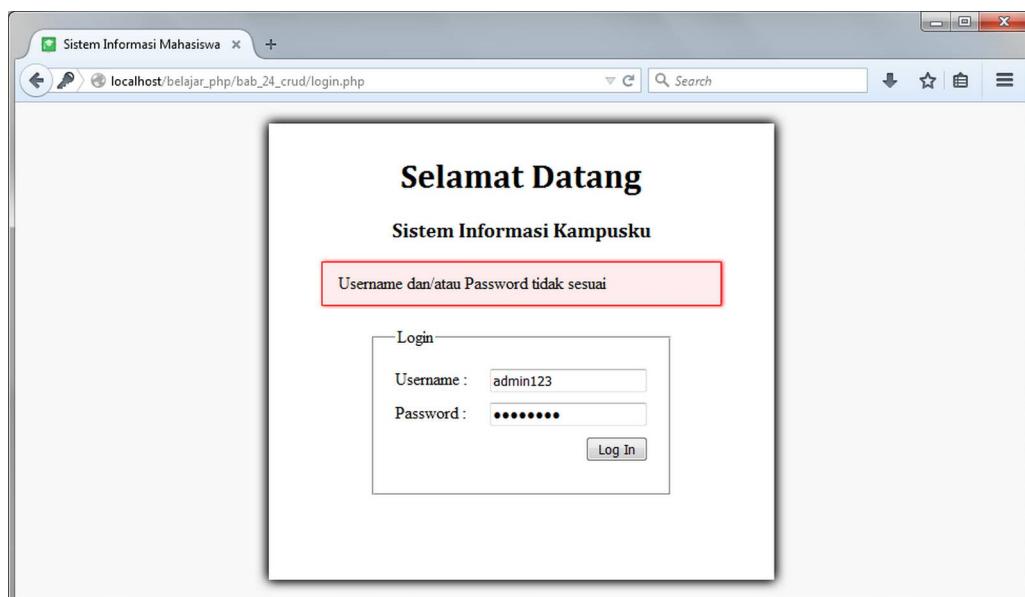
```

File ini terdiri dari sebuah fungsi `mysqli_connect()`. Perhatikan kali ini saya sudah langsung memilih database **kampusku**. Nantinya, file ini akan dipanggil menggunakan fungsi `include()` dari halaman-halaman lain.

Dengan memisahkan koneksi PHP MySQL menjadi file tersendiri, kode program menjadi lebih fleksibel. Jika nanti saya ingin mengganti nama database atau mengganti nama user MySQL, cukup mengubahnya dari halaman `connection.php` ini.

## 24.5 Halaman Login (login.php)

Agar bisa mengakses **Sistem Informasi Kampusku**, setiap user harus mengisi form login terlebih dahulu. Ini dijalankan oleh file `login.php`. Berikut tampilannya:



Gambar: Pesan kesalahan tampil karena username dan password tidak cocok

Kode program untuk `login.php` ini 90% sama dengan case study pada bab *Cookie dan Session*. Bedanya, kali ini pengecekan `username` dan `password` diambil dari database. Berikut kode program yang melakukan proses tersebut:

```

1 <?php
2 // buat koneksi ke mysql dari file connection.php
3 include("connection.php");
4
5 // filter dengan mysqli_real_escape_string
6 $username = mysqli_real_escape_string($link,$username);
7 $password = mysqli_real_escape_string($link,$password);
8
9 // generate hashing
10 $password_sha1 = sha1($password);
11
12 // cek apakah username dan password ada di tabel admin
13 $query = "SELECT * FROM admin WHERE username = '$username'
14           AND password = '$password_sha1'";
15 $result = mysqli_query($link,$query);
16
17 if(mysqli_num_rows($result) == 0 ) {
18     // data tidak ditemukan, buat pesan error
19     $pesan_error .= "Username dan/atau Password tidak sesuai";
20 }
21 ?>

```

Perlu saya ingatkan kembali bahwa kode program diatas hanyalah sebagian kecil dari seluruh kode program di halaman `login.php`. Silahkan anda buka file ini untuk lebih jelasnya.

Di awal kode program, saya menulis `include("connection.php")`. Perintah ini digunakan untuk memanggil file `connection.php`. Dengan pemanggilan ini, koneksi PHP MySQL sudah langsung dibuat dan variabel `$link` akan berisi hasil dari pemanggilan fungsi `mysqli_connect()`, yakni sebagai variabel *handle*.

Ketika form login di submit, nilai `$_POST["username"]` akan disimpan ke dalam variabel `$username`, sedangkan nilai `$_POST["password"]` akan disimpan ke dalam variabel `$password`.

Seperti yang telah kita bahas sebelumnya, kolom password di tabel admin saya simpan dalam bentuk *hashing sha1()*. Oleh karena itu saya harus menggenerate kembali nilai *hashing sha1* dari form yang diinput user:

```

<?php
$password_sha1 = sha1($password);
?>

```

Agar data bisa dianggap sesuai, nilai dari `$password_sha1` ini harus sama dengan nilai kolom password yang tersimpan di dalam tabel admin. Untuk memeriksa apakah `username` dan `password` cocok, cukup dengan 1 query:

```
SELECT * FROM admin WHERE username = '$username' AND
password = '$password_sha1';
```

Apabila data \$username dan \$password\_sha1 cocok dan ada di dalam database, fungsi `mysql_num_rows()` akan menghasilkan 1, jika data tidak ditemukan atau salah satu dari username dan password tidak cocok, fungsi `mysql_num_rows()` akan menghasilkan nilai 0.

Jika didapat hasil `mysql_num_rows() == 0` (yang artinya username dan password tidak cocok), load kembali halaman dan tampilkan pesan kesalahan.

Namun apabila username dan password lolos validasi, jalankan kode program berikut:

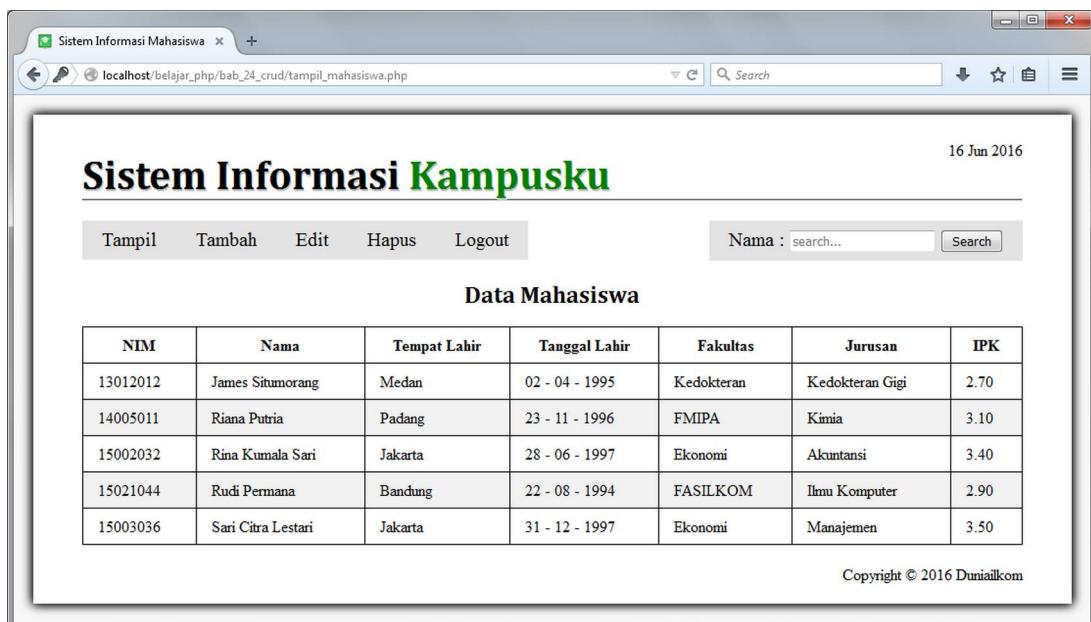
```
1 <?php
2 // jika lolos validasi, set session
3 if ($pesan_error === "") {
4     session_start();
5     $_SESSION["nama"] = $username;
6     header("Location: tampil_mahasiswa.php");
7 }
8 ?>
```

Disini saya membuat *session nama* dengan nilai \$username. Setelah itu, user langsung di redirect ke halaman `tampil_mahasiswa.php`.

## 24.6 Menampilkan Data Mahasiswa (`tampil_mahasiswa.php`)

Sesuai dengan namanya, halaman `tampil_mahasiswa.php` digunakan untuk menampilkan isi seluruh tabel mahasiswa. Kode program yang digunakan juga relatif sederhana. Saya tinggal menjalankan query `SELECT * FROM mahasiswa`, kemudian menggunakan perulangan `while` untuk men-generate tabel HTML. Ini sudah kita pelajari dalam bab sebelumnya.

Berikut tampilan dari halaman `tampil_mahasiswa.php`:



Gambar: Menampilkan seluruh data tabel mahasiswa

Mengenai aspek design seperti judul, menu navigasi, dan tampilan tabel saya atur menggunakan HTML dan CSS. Seluruh kode CSS saya tempatkan ke dalam file external: `style.css`. Jika anda sudah pernah belajar tentang CSS, silahkan mempelajari kode yang ada.

Saya juga menggunakan *favicon.png* untuk mempercantik icon halaman. Semuanya ditempatkan di dalam tag `<head>`:

```
1 <head>
2   <meta charset="UTF-8">
3   <title>Sistem Informasi Mahasiswa</title>
4   <link href="style.css" rel="stylesheet" >
5   <link rel="icon" href="favicon.png" type="image/png" >
6 </head>
```

Sekarang, mari masuk ke kode PHP. Pada baris pertama, saya memeriksa kehadiran *session nama*. Apabila tidak ada, langsung redirect ke halaman *login.php*:

```
1 <?php
2     session_start();
3     if (!isset($_SESSION["nama"])) {
4         header("Location: login.php");
5     }
6 ?>
```

Kode program ini saya tempatkan di bagian awal file-file lain. Artinya, jika user belum login tapi mencoba mengakses langung halaman ini, ia akan di redirect ke halaman login.

Jika anda perhatikan, di sisi kanan menu navigasi terdapat sebuah form pencarian. Form ini digunakan untuk mencari nama mahasiswa. Berikut struktur HTML dari form tersebut:

```

1 <form id="search" action="tampil_mahasiswa.php" method="get">
2   <p>
3     <label for="nim">Nama : </label>
4     <input type="text" name="nama" id="nama" placeholder="search..." >
5     <input type="submit" name="submit" value="Search">
6   </p>
7 </form>

```

Idenya adalah, ketika seseorang mengetik nama dan menekan tombol *search*, sebuah variabel `$_GET["nama"]` akan dikirim ke halaman `tampil_mahasiswa.php`. Variabel ini nantinya akan ditangkap oleh PHP yang berarti form pencarian telah di klik.

Hal ini di deteksi dari kehadiran variabel `$_GET["nama"]`. Jika ada, artinya form telah di **submit**. Jika tidak ada, berarti halaman ini diakses secara normal (tampilkan seluruh data mahasiswa).

Berikut gambaran alur logika halaman `tampil_mahasiswa.php`:

```

IF (session nama tidak ditemukan) {
  redirect ke login.php
}
IF (form disubmit dari form pencarian) {
  Jalankan query SELECT...LIKE // tampilkan isi tabel sesuai pencarian
}
ELSE {
  Jalankan query SELECT * FROM mahasiswa // tampilkan seluruh tabel
}

```

Berikut implementasinya ke dalam kode PHP:

```

1 <?php
2   // periksa apakah user sudah login, cek kehadiran session name
3   // jika tidak ada, redirect ke login.php
4   session_start();
5   if (!isset($_SESSION["nama"])) {
6     header("Location: login.php");
7   }
8
9   // buka koneksi dengan MySQL
10  include("connection.php");
11
12  // ambil pesan jika ada
13  if (isset($_GET["pesan"])) {
14    $pesan = $_GET["pesan"];
15  }
16
17  // cek apakah form telah di submit

```

```
18 // berasal dari form pencarian, siapkan query
19 if (isset($_GET["submit"])) {
20
21     // ambil nilai nama
22     $nama = htmlentities(strip_tags(trim($_GET["nama"])));
23
24     // filter untuk $nama untuk mencegah sql injection
25     $nama = mysqli_real_escape_string($link,$nama);
26
27     // buat query pencarian
28     $query = "SELECT * FROM mahasiswa WHERE nama LIKE '%$nama%' ";
29     $query .= "ORDER BY nama ASC";
30
31     // buat pesan
32     $pesan = "Hasil pencarian untuk nama <b>\\"$nama\"</b> :";
33 }
34 else {
35     // bukan dari form pencarian
36     // siapkan query untuk menampilkan seluruh data dari tabel mahasiswa
37     $query = "SELECT * FROM mahasiswa ORDER BY nama ASC";
38 }
39 ?>
```

Silahkan anda pelajari sejenak kode program diatas. Selain “menangkap” variabel `$_GET["nama"]`, saya juga mengambil nilai dari `$_GET["pesan"]`. Pesan digunakan untuk menampilkan berbagai keterangan tambahan, baik dari halaman ini maupun dari halaman lain.

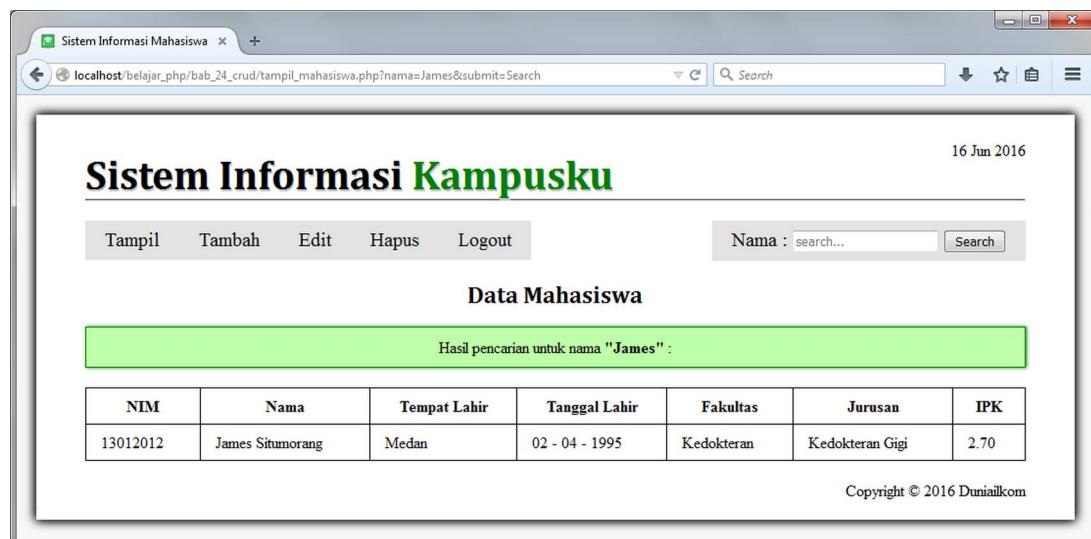
Hasil akhir dari kode program diatas adalah variabel `$query` akan berisi 1 diantara 2 kondisi:

Jika berasal dari form pencarian, variabel `$query` akan berisi string: “`SELECT * FROM mahasiswa WHERE nama LIKE '%$nama%' ORDER BY nama ASC`”.

Jika bukan dari form pencarian, variabel `$query` akan berisi string “`SELECT * FROM mahasiswa ORDER BY nama ASC`”.

Selanjutnya, saya tinggal menjalankan fungsi `mysqli_query($link, $query)` dan melakukan proses looping untuk membuat tabel HTML.

Agar lebih mudah dipahami, silahkan anda coba langsung dengan mengetik nama di form pencarian lalu klik submit, Berikut contoh tampilannya:



Gambar: Hasil proses pencarian

Saya memang tidak mengharapkan anda untuk bisa langsung mengerti kode program yang saya gunakan di halaman ini. Silahkan pelajari secara perlahan (dan mungkin berulang-ulang) agar lebih paham.

## 24.7 Menambah Data Baru (tambah\_mahasiswa.php)

Jika anda men-klik menu “Tambah” di bagian navigasi, akan tampil halaman tambah\_mahasiswa.php. Halaman ini berisi sebuah form yang digunakan untuk menambah data mahasiswa baru.

Gambar: Form untuk menambah data mahasiswa baru

Dibandingkan dengan `tampil_mahasiswa.php`, kode program untuk `tambah_mahasiswa.php` lebih panjang, karena saya perlu melakukan proses validasi form.

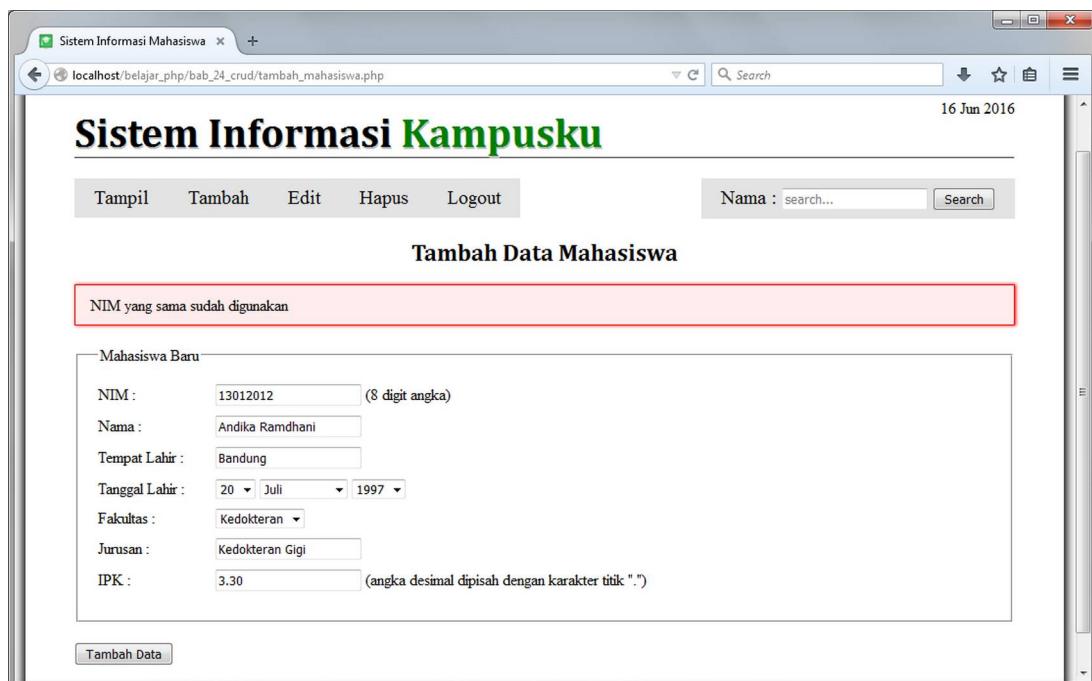
Sebagai contoh, saya harus memastikan panjang dari inputan **nim** persis 8 angka, apakah **nama** sudah diisi, apakah nilai **ipk** memang berupa angka atau bukan, dll. Sebagian besar dari teknik validasi ini telah kita bahas pada bab tentang *Pemrosesan Form*. Oleh karena itu saya tidak akan membahasnya lagi.

Khusus untuk proses validasi yang datanya akan diinput ke dalam database, terdapat 1 hal penting yang harus selalu diperhatikan. Bagaimana jika seseorang menginput nomor nim dengan data yang sudah ada di dalam tabel?

Yup, MySQL akan mengeluarkan pesan error. Ini karena kolom nim sudah saya set sebagai *primary key*. Artinya, tidak boleh ada nomor nim ganda. Bagaimana cara membuat validasi seperti ini? Berikut teknik yang saya gunakan:

```
1 <?php
2 // cek ke database, apakah sudah ada nomor NIM yang sama
3 // filter data $nim
4 $nim = mysqli_real_escape_string($link,$nim);
5 $query = "SELECT * FROM mahasiswa WHERE nim='$nim'";
6 $hasil_query = mysqli_query($link, $query);
7
8 // cek jumlah record (baris), jika ada, $nim tidak bisa diproses
9 $jumlah_data = mysqli_num_rows($hasil_query);
10 if ($jumlah_data >= 1 ) {
11     $pesan_error .= "NIM yang sama sudah digunakan <br>";
12 }
13 ?>
```

Caranya, pada saat proses validasi berlangsung, jalankan query `SELECT * FROM mahasiswa WHERE nim='\$nim'`. Jika hasil `mysqli_num_rows == 1`, artinya nomor nim tersebut sudah ada di dalam tabel. Karena itu tampilkan pesan error:



Gambar: NIM yang sama sudah ada di dalam tabel mahasiswa, silahkan input nomor lain

Hal seperti ini perlu dilakukan agar user yang sedang menginput data terinformasi dan bisa mencari nomor nim baru.

Untuk proses validasi lain, tidak akan saya bahas karena sudah kita pelajari dalam bab tentang pemrosesan form. Silahkan anda pelajari dari file tambah\_mahasiswa.php.

Jika seluruh proses validasi berhasil dilewati (artinya tidak terdapat error), query **INSERT** sudah bisa dijalankan. Berikut kode program yang saya gunakan:

```

1 <?php
2 // jika tidak ada error, input ke database
3 if ($pesan_error === "") {
4
5     // filter semua data
6     $nim = mysqli_real_escape_string($link,$nim);
7     $nama = mysqli_real_escape_string($link,$nama );
8     $tempat_lahir = mysqli_real_escape_string($link,$tempat_lahir);
9     $fakultas = mysqli_real_escape_string($link,$fakultas);
10    $jurusan = mysqli_real_escape_string($link,$jurusan);
11    $tgl = mysqli_real_escape_string($link,$tgl);
12    $bln = mysqli_real_escape_string($link,$bln);
13    $thn = mysqli_real_escape_string($link,$thn);
14    $ipk = (float) $ipk;
15
16    //gabungkan format tanggal agar sesuai dengan date MySQL
17    $tgl_lhr = $thn."-".$bln."-".$tgl;
18
19    //buat dan jalankan query INSERT

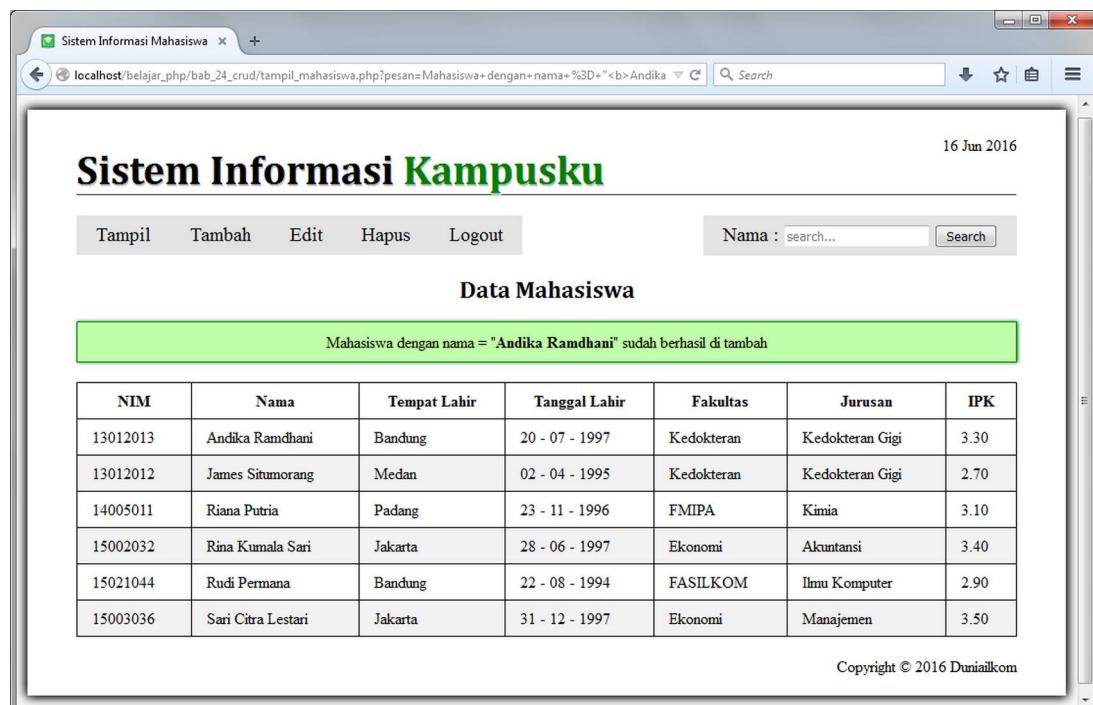
```

```
20     $query = "INSERT INTO mahasiswa VALUES ";
21     $query .= "('$nim', '$nama', '$tempat_lahir', ";
22     $query .= "'$tgl_lahir', '$fakultas', '$jurusan', $ipk)";
23
24     $result = mysqli_query($link, $query);
25
26     //periksa hasil query
27     if($result) {
28         // INSERT berhasil, redirect ke tampil_mahasiswa.php + pesan
29         $pesan = "Mahasiswa dengan nama = <b>$nama</b>\"
30                 sudah berhasil di tambah";
31         $pesan = urlencode($pesan);
32         header("Location: tampil_mahasiswa.php?pesan={$pesan}");
33     }
34     else {
35         die ("Query gagal dijalankan: ".mysqli_errno($link).
36             " - ".mysqli_error($link));
37     }
38 }
39 ?>
```

Tidak ada hal yang baru disini. Setelah menfilter seluruh inputan menggunakan fungsi `mysqli_real_escape_string()`, saya tinggal merangkai query `INSERT`.

Khusus untuk kolom `tanggal_lahir`, saya harus menggabungkannya agar sesuai dengan format MySQL, yakni `yyyy-mm-dd`.

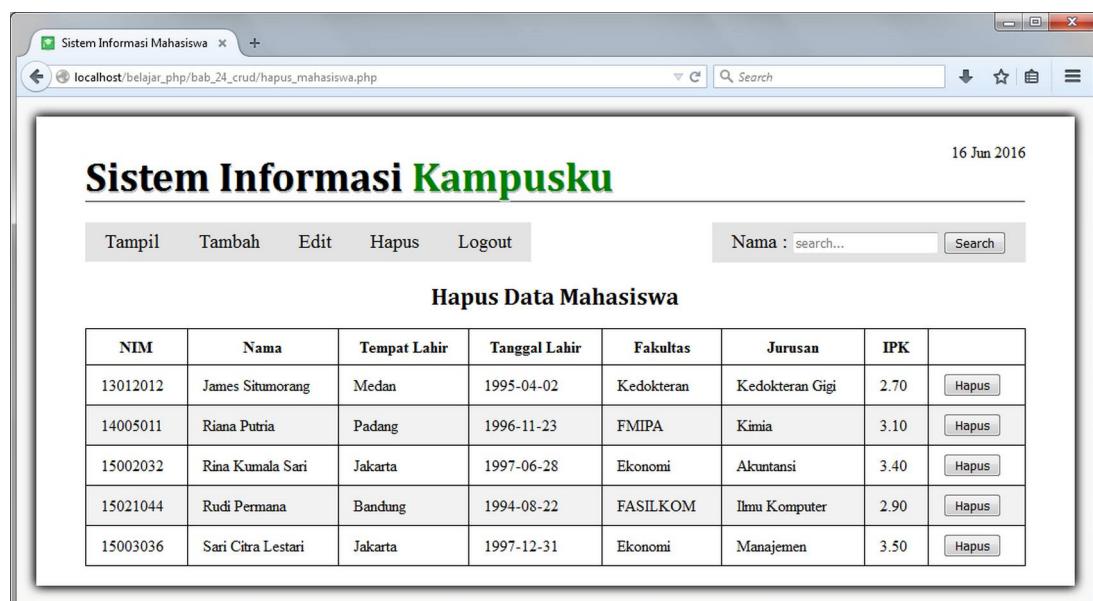
Setelah query berhasil dijalankan, user akan di redirect ke halaman `tampil_mahasiswa.php`. Selain itu, saya juga mengirim pesan bahwa: *Mahasiswa dengan nama = <b>\$nama</b> sudah berhasil di tambah*. Pesan ini akan “ditangkap” dan ditampilkan di halaman `tampil_mahasiswa.php`. Silahkan anda coba sendiri agar lebih jelas:



Gambar: Data mahasiswa baru berhasil ditambahkan

## 24.8 Menghapus Data Mahasiswa (hapus\_mahasiswa.php)

Halaman berikutnya adalah `hapus_mahasiswa.php`. Halaman ini bisa diakses dari menu **Hapus**. Berikut tampilannya:



Gambar: Halaman untuk menghapus data mahasiswa

Tampilan halaman `hapus_mahasiswa.php` mirip dengan halaman `tampil_mahasiswa.php`. Disini seluruh data mahasiswa ditampilkan dalam bentuk tabel. Bedanya, pada kolom paling kanan terdapat tombol **hapus**. Pada saat tombol **hapus** di-klik, halaman akan *reload* sebentar dan data tersebut akan dihapus dari database.

Disini saya membuat setiap baris memiliki 1 form. Tombol “**hapus**” di sisi kanan merupakan tombol *submit* dari form tersebut. Agar lebih jelas, berikut potongan source code HTML dari tabel ini:

```

1  <tr>
2      <td>13012013</td>
3      <td>Andika Ramdhani</td>
4      <td>Bandung</td>
5      <td>1997-07-20</td>
6      <td>Kedokteran</td>
7      <td>Kedokteran Gigi</td>
8      <td>3.30</td>
9      <td>
10         <form action="hapus_mahasiswa.php" method="post" >
11             <input type="hidden" name="nim" value="13012013" >
12             <input type="submit" name="submit" value="Hapus" >
13         </form>
14     </td>
15 </tr>
16
17 <tr>
18     <td>13012012</td>
19     <td>James Situmorang</td>
20     <td>Medan</td>
21     <td>1995-04-02</td>
22     <td>Kedokteran</td>
23     <td>Kedokteran Gigi</td>
24     <td>2.70</td>
25     <td>
26         <form action="hapus_mahasiswa.php" method="post" >
27             <input type="hidden" name="nim" value="13012012" >
28             <input type="submit" name="submit" value="Hapus" >
29         </form>
30     </td>
31 </tr>

```

Perhatikan bagaimana saya menambahkan tag `<form>` ke dalam tabel. Setiap baris tabel akan memiliki struktur HTML seperti ini.

Untuk setiap form, saya menggunakan 2 buah tag input, yakni tag `<input type="hidden">` dan tag `<input type="submit">`. Apabila sudah pernah mempelajari form HTML, saya yakin sudah

paham bagaimana cara kerja kedua tag ini. Saya menggunakan tag `<input type="hidden">` karena data nim ini tidak perlu ditampilkan, tapi harus dikirim bersama dengan form.

Di dalam form saya menggunakan atribut `action="hapus_mahasiswa.php"` dan `method="post"`. Artinya, nilai form akan dikirim ke halaman yang sama (`hapus_mahasiswa.php`) dengan method POST.

Data yang dikirim akan berbentuk `$_POST["submit"]` dan `$_POST["nim"]`. Kedua nilai inilah yang menjadi patokan bahwa tombol “hapus” telah diklik dan query **DELETE** bisa dijalankan. Nantinya, saya harus “menangkap” nilai ini untuk memproses penghapusan data.

Sebagai informasi tambahan, sebenarnya ada cara yang lebih mudah dan tanpa form. Saya bisa mengubah tombol *hapus* menjadi sebuah link dengan query string, seperti contoh berikut:

```
<a href="hapus_mahasiswa.php?nim=13012012">Hapus</a>
<a href="hapus_mahasiswa.php?nim=15002032">Hapus</a>
```

Disini, data yang dikirim bisa “ditangkap” menggunakan `$_GET["nim"]`. Tapi karena proses penghapusan data cukup beresiko, saya memutuskan untuk menggunakan method POST dan dikirim dari form.

Kembali ke kode program kita, bagaimana cara menggenerate tabel seperti ini? Berikut potongan kode program yang saya gunakan:

```
1 <?php
2 // buat query untuk menampilkan seluruh data tabel mahasiswa
3 $query = "SELECT * FROM mahasiswa ORDER BY nama ASC";
4 $result = mysqli_query($link, $query);
5
6 //buat perulangan untuk element tabel dari data mahasiswa
7 while($data = mysqli_fetch_assoc($result))
8 {
9     echo "<tr>";
10    echo "<td>$data[nim]</td>";
11    echo "<td>$data[nama]</td>";
12    echo "<td>$data[tempat_lahir]</td>";
13    echo "<td>$data[tanggal_lahir]</td>";
14    echo "<td>$data[fakultas]</td>";
15    echo "<td>$data[jurusan]</td>";
16    echo "<td>$data[ipk]</td>";
17    echo "<td>";
18 ?>
19    <form action="hapus_mahasiswa.php" method="post" >
20        <input type="hidden" name="nim" value="<?php echo "$data[nim]"; ?>" >
21        <input type="submit" name="submit" value="Hapus" >
22    </form>
23 <?php
24     echo "</td>";
```

```

25     echo "</tr>";
26 }
27 ?>

```

Disini, nilai atribut **value** untuk form saya ambil dari database. Dengan demikian, setiap baris tabel memiliki tombol **hapus** yang ketika di klik akan mengirim nilai `$_POST["nim"]`.

Selanjutnya, bagaimana cara memproses query **DELETE**? Berikut kode program yang saya tempatkan dibagian atas halaman `hapus_mahasiswa.php` (sebelum kode HTML):

```

1  <?php
2  // periksa apakah user sudah login, cek kehadiran session name
3  // jika tidak ada, redirect ke login.php
4  session_start();
5  if (!isset($_SESSION["nama"])) {
6      header("Location: login.php");
7  }
8
9  // buka koneksi dengan MySQL
10 include("connection.php");
11
12 // cek apakah form telah di submit (untuk menghapus data)
13 if (isset($_POST["submit"])) {
14     // form telah disubmit, proses data
15
16     // ambil nilai nim
17     $nim = htmlentities(strip_tags(trim($_POST["nim"])));
18     // filter data
19     $nim = mysqli_real_escape_string($link,$nim);
20
21     //jalankan query DELETE
22     $query = "DELETE FROM mahasiswa WHERE nim='$nim' ";
23     $hasil_query = mysqli_query($link, $query);
24
25     //periksa query, tampilkan pesan kesalahan jika gagal
26     if($hasil_query) {
27         // DELETE berhasil, redirect ke tampil_mahasiswa.php + pesan
28         $pesan = "Mahasiswa dengan nim = \"<b>$nim</b>\""
29                     sudah berhasil di hapus";
30         $pesan = urlencode($pesan);
31         header("Location: tampil_mahasiswa.php?pesan={$pesan}");
32     }
33     else {
34         die ("Query gagal dijalankan: ".mysqli_errno($link).
35             " - ".mysqli_error($link));
36     }

```

```
37     }
38 ?>
```

Sama seperti halaman lain, di baris paling awal saya memeriksa kehadiran **session nama**. Jika tidak ada, halaman akan di redirect ke `login.php` dan user harus login terlebih dahulu.

Apabila variabel **session nama** ditemukan, proses halaman dengan membuka koneksi ke MySQL menggunakan `include("connection.php")`.

Selanjutnya saya membuat percabangan (kondisi IF), apakah halaman ini berasal dari tombol **hapus**, atau diakses dari menu navigasi? Ini bisa dideteksi dari kehadiran variabel `$_POST["submit"]`. Apabila variabel ini ditemukan, artinya halaman dibuka oleh tombol form “**hapus**”. Jika tidak ada, berarti halaman di akses dari menu.

Jika variabel `$_POST["submit"]` tidak ditemukan, lompati kode program PHP ini, dan masuk ke HTML untuk menampilkan tabel mahasiswa dengan tombol **hapus**.

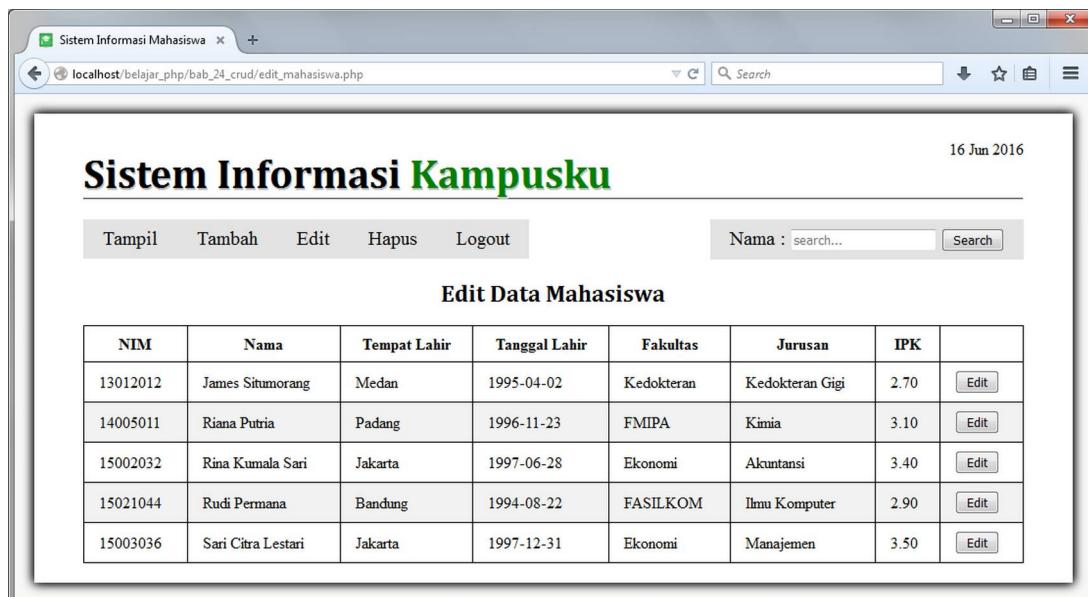
Tapi jika terdeteksi halaman ini di akses dari tombol **hapus**, kita bisa mulai proses penghapusan data. Ambil nilai `$_POST["nim"]`, pindahkan ke variabel `$nim`, lalu jalankan query berikut:

```
DELETE FROM mahasiswa WHERE nim='$nim';
```

Jika query berjalan lancar dan tidak ada masalah, redirect ke halaman `tampil_mahasiswa` dengan pesan: *Mahasiswa dengan nim = "<b>\$nim</b>" sudah berhasil di hapus*. Proses penghapusan data sukses dijalankan.

## 24.9 Edit Data Mahasiswa (edit\_mahasiswa.php)

Aspek terakhir dari CRUD yang belum kita bahas adalah **Update**, yakni bagaimana cara mengedit data mahasiswa. Halaman ini bisa diakses dari menu **Edit**.



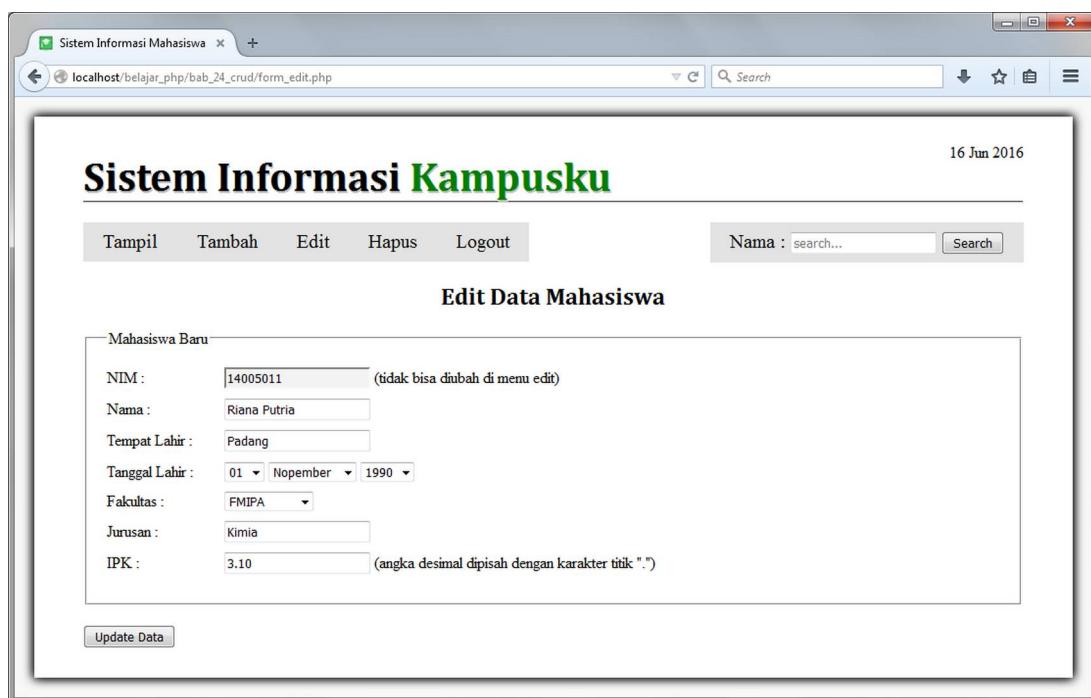
Gambar: Halaman edit mahasiswa

Dari seluruh CRUD, proses **update** bisa dibilang yang paling kompleks, karena kita harus melalukan beberapa langkah: ambil data dari database, tampilkan ke dalam form, jalankan proses validasi, dan update data.

Oleh karena itu pula saya memecah alur update kedalam 2 buah halaman: `edit_mahasiswa.php` dan `form_edit.php`.

Kode program untuk halaman `edit_mahasiswa.php` sangat mirip dengan `hapus_mahasiswa.php`, dimana saya menampilkan tabel dengan sebuah tombol **Edit**. Bedanya, ketika tombol **Edit** ini diklik, pemrosesan akan dilakukan di halaman `form_edit.php`.

Karena sangat mirip, saya tidak akan membahas lagi kode program `edit_mahasiswa.php`. Silahkan klik salah satu tombol **Edit** dan berikut tampilannya:



The screenshot shows a web browser window for 'Sistem Informasi Mahasiswa'. The URL is `localhost/belajar_php/bab_24_crud/form_edit.php`. The page title is 'Sistem Informasi Kampusku'. The date '16 Jun 2016' is in the top right. A navigation bar at the top includes 'Tampil', 'Tambah', 'Edit', 'Hapus', and 'Logout'. A search bar is on the right. The main content is titled 'Edit Data Mahasiswa'. A form titled 'Mahasiswa Baru' contains the following data:

NIM :	14005011	(tidak bisa diubah di menu edit)	
Nama :	Riana Putria		
Tempat Lahir :	Padang		
Tanggal Lahir :	01	Nopember	1990
Fakultas :	FMIPA		
Jurusan :	Kimia		
IPK :	3.10	(angka desimal dipisah dengan karakter titik ".")	

At the bottom of the form is a 'Update Data' button.

Gambar: Form untuk edit data mahasiswa

Seperti yang terlihat, form ini sangat mirip dengan halaman `tambah_mahasiswa.php`. Dan, memang kode programnya saya ambil dari halaman tersebut, termasuk seluruh proses validasi form. Di halaman ini, setiap object form langsung berisi data yang diambil dari database.

Alur rancangan kode program untuk halaman `edit_mahasiswa.php` adalah sebagai berikut:

```

IF (session nama tidak ditemukan) {
    redirect ke login.php
}
IF (form disubmit dari tombol Edit) {
    Tampilkan form dengan data yang diambil dari database
}
ELSE {
    Proses validasi data
    IF (tidak lolos validasi) {
        Tampilkan kembali form dengan pesan kesalahan
    }
}
IF (lolos validasi dan form disubmit dari halaman ini) {
    Proses query UPDATE, kemudian redirect ke tampil_mahasiswa.php
}

```

Terlihat cukup kompleks, tapi jika anda sudah paham kode program untuk halaman tambah\_mahasiswa.php, halaman ini hanyalah versi modifikasinya.

Pertama, adalah men-generate isi form dari database. Berikut kode program yang saya gunakan:

```

1  <?php
2      if ($_POST["submit"]=="Edit") {
3          //nilai form berasal dari halaman edit_mahasiswa.php
4
5          // ambil nilai nim
6          $nim = htmlentities(strip_tags(trim($_POST["nim"])));
7          // filter data
8          $nim = mysqli_real_escape_string($link,$nim);
9
10         // ambil semua data dari database untuk menjadi nilai awal form
11         $query = "SELECT * FROM mahasiswa WHERE nim='$nim'";
12         $result = mysqli_query($link, $query);
13
14         if(!$result){
15             die ("Query Error: ".mysqli_errno($link).
16                  " - ".mysqli_error($link));
17         }
18
19         // tidak perlu pakai perulangan while, karena hanya ada 1 record
20         $data = mysqli_fetch_assoc($result);
21
22         $nama = $data["nama"];
23         $tempat_lahir = $data["tempat_lahir"];
24         $fakultas = $data["fakultas"];
25         $jurusan = $data["jurusan"];

```

```

26     $ipk = $data["ipk"];
27
28     // untuk tanggal harus dipecah
29     $tg1 = substr($data["tanggal_lahir"],0,4);
30     $b1n = substr($data["tanggal_lahir"],5,2);
31     $thn = substr($data["tanggal_lahir"],8,2);
32
33     // bebaskan memory
34     mysqli_free_result($result);
35 }
36 ?>

```

Disini saya mengambil terlebih dahulu nilai `$_POST["nim"]` yang berasal dari halaman `edit_mahasiswa.php`. Nilai inilah yang akan digunakan untuk mengambil data dari tabel melalui query:

```
SELECT * FROM mahasiswa WHERE nim='$nim';
```

Selanjutnya nilai kolom dari hasil query akan diinput ke dalam setiap variabel seperti `$nama`, `$tempat_lahir`, `$fakultas`, dll. Khusus untuk kolom `tanggal_lahir`, saya harus memecah format kolom date MySQL, agar bisa diambil mana bagian tanggal, bulan dan tahun. Masing-masingnya diinput ke variabel `$tg1`, `$b1n` dan `$thn`.

Hasil akhir dari kode program diatas, nilai setiap kolom diinput ke variabel yang sesuai yakni: `$nim`, `$nama`, `$tempat_lahir`, `$fakultas`, `$jurusan`, `$ipk`, `$tg1`, `$b1n` dan `$thn`. Variabel ini akan menjadi nilai dari atribut **value** pada form.

Di dalam form, inputan `nim` saya tambahkan atribut **readonly**:

```

1 <p>
2   <label for="nim">NIM : </label>
3   <input type="text" name="nim" id="nim" value="<?php echo $nim ?>" readonly>
4   (tidak bisa diubah di menu edit)
5 </p>

```

Untuk keperluan update data, nilai `nim` inilah yang menjadi patokan utama dan seharunya memang tidak bisa diedit.

Setelah form diisi (atau diperbarui), nilai dari setiap form akan di validasi kembali. Karena bisa saja ada data yang tidak sesuai. Sekali lagi, proses validasi ini saya ambil dari halaman `tambah_mahasiswa.php`.

Jika proses validasi berhasil dilalui, saatnya membuat query **Update**:

```

1  <?php
2  // jika tidak ada error, input ke database
3  if (($pesan_error === "") AND ($_POST["submit"]=="Update Data")) {
4
5      // buka koneksi dengan MySQL
6      include("connection.php");
7
8      // filter semua data
9      $nim = mysqli_real_escape_string($link,$nim);
10     $nama = mysqli_real_escape_string($link,$nama );
11     $tempat_lahir = mysqli_real_escape_string($link,$tempat_lahir);
12     $fakultas = mysqli_real_escape_string($link,$fakultas);
13     $jurusan = mysqli_real_escape_string($link,$jurusan);
14     $tgl = mysqli_real_escape_string($link,$tgl);
15     $bln = mysqli_real_escape_string($link,$bln);
16     $thn = mysqli_real_escape_string($link,$thn);
17     $ipk = (float) $ipk;
18
19     //gabungkan format tanggal agar sesuai dengan date MySQL
20     $tgl_lhr = $thn."-".$bln."-".$tgl;
21
22     //buat dan jalankan query UPDATE
23     $query = "UPDATE mahasiswa SET ";
24     $query .= "nama = '$nama', tempat_lahir = '$tempat_lahir', ";
25     $query .= "tanggal_lahir = '$tgl_lhr', fakultas='$fakultas', ";
26     $query .= "jurusan = '$jurusan', ipk=$ipk ";
27     $query .= "WHERE nim = '$nim'";
28
29     $result = mysqli_query($link, $query);
30
31     //periksa hasil query
32     if($result) {
33         // INSERT berhasil, redirect ke tampil_mahasiswa.php + pesan
34         $pesan = "Mahasiswa dengan nama = \"<b>$nama</b>\""
35             " sudah berhasil di update";
36         $pesan = urlencode($pesan);
37         header("Location: tampil_mahasiswa.php?pesan={$pesan}");
38     }
39     else {
40         die ("Query gagal dijalankan: ".mysqli_errno($link).
41             " - ".mysqli_error($link));
42     }
43 }
44 ?>

```

Di awal saya mengecek apakah tidak ada pesan error (yang berarti lolos validasi) dan halaman

telah di submit. Jika semuanya TRUE, mulai dengan filter seluruh data menggunakan fungsi `mysqli_real_escape_string()`.

Setelah itu tinggal jalankan query berikut:

```
UPDATE mahasiswa SET nama = '$nama', tempat_lahir = '$tempat_lahir',
tanggal_lahir = '$tgl_lhr', fakultas = '$fakultas', jurusan = '$jurusan',
ipk = $ipk WHERE nim = '$nim';
```

Jika tidak masalah, redirect user ke halaman `tampil_mahasiswa.php` dengan pesan: “*Mahasiswa dengan nama = <b>\$nama</b> sudah berhasil di update*”.

## 24.10 Halaman Logout (logout.php)

Proses `logout` user ditangani oleh halaman `logout.php`. Kode programnya cukup sederhana:

```
1 <?php
2   session_start();
3   // hapus session
4   unset($_SESSION["nama"]);
5
6   // redirect ke halaman login.php
7   header("Location: login.php");
8 ?>
```

Yup, hanya menghapus variabel `session nama`, kemudian redirect ke `login.php`.

## 24.11 Halaman Utama (index.php)

Agar setiap halaman memiliki nama yang informatif, saya memutuskan untuk tidak menggunakan `index.php` sebagai halaman utama. Halaman ini hanya sebagai “penghubung”. Mari kita lihat kode program yang saya tulis untuk halaman ini:

```
1 <?php
2   // langsung redirect ke halaman tampil mahasiswa
3   header("Location: tampil_mahasiswa.php");
4 ?>
```

Maksudnya, jika seseorang mengakses folder `crud` secara langsung tanpa menulis nama file, seperti: `http://localhost/belajar_php/bab_24_crud/`, ia akan di redirect ke halaman `tampil_mahasiswa.php`.

Dan jika anda masih ingat, di halaman `tampil_mahasiswa.php` ini saya juga akan memeriksa apakah variabel `session nama` ada atau tidak. Jika tidak ada, user kembali di redirect ke halaman `login.php`. Walaupun terdapat 2 kali proses `redirect`, durasi pemrosesan berlangsung sangat cepat dan tidak akan terdeteksi oleh user.

Dengan teknik seperti ini, saya bisa menggunakan halaman `tampil_mahasiswa.php` sebagai halaman utama, tapi tanpa harus mengubah nama filenya menjadi `index.php`.

## 24.12 Modifikasi Sistem Informasi Kampusku

Sampai disini saya berharap anda bisa memahami alur kode program dari **sistem informasi kampusku**. Walau demikian, saya juga tidak akan heran jika anda merasa “terintimidasi” dengan banyaknya kode program yang ada.

Jika masih bingung, silahkan anda pelajari secara perlahan. Saya sendiri butuh waktu hampir seminggu untuk merancang kode program tersebut (yang jumlahnya sudah mencapai ribuan baris!). Silahkan anda luangkan waktu beberapa jam untuk mempelajari kode program yang ada.

Selanjutnya, saya juga menantang anda untuk melangkah lebih jauh lagi, yakni dengan memodifikasi kode program tersebut. Ini juga sebagai bahan latihan dan menguji kemampuan anda tentang PHP. Berikut beberapa hal yang bisa dilakukan:

### Pecah kode program menjadi file-file terpisah

Jika anda perhatikan, banyak kode program yang berulang di hampir setiap file. Sebagai contoh, bagian *header* dalam setiap halaman nyaris sama, yakni menu navigasi, form pencarian dan *footer*.

Ketika ingin menambahkan menu baru, saya harus mengedit satu persatu halaman yang ada. Akan jauh lebih efektif jika *header* dan *footer* dipecah menjadi file terpisah, lalu diinput kembali menggunakan fungsi *include()* PHP.

### Menambahkan Form Upload Gambar

Di dalam **Sistem Informasi Kampusku** saya tidak menggunakan form upload agar kodennya tidak terlalu kompleks. Silahkan jika anda ingin memodifikasinya dengan menambahkan *form upload* gambar mahasiswa.

Namun perlu saya jelaskan bahwa yang disimpan ke dalam database hanyalah nama gambar, bukan file gambarnya. File gambar cukup di upload ke folder yang telah kita siapkan, misalnya folder: **gambar**.

Untuk menampilkan gambar, buat ulang atribut **src** dari tag **<img>** menggunakan variabel PHP, seperti contoh berikut:

```

```

Variabel **\$nama\_gambar** ini nantinya diambil dari database.

### Membuat CRUD untuk tabel admin

Silahkan jika anda ingin menambah menu baru yang isinya sebuah CRUD untuk tabel **admin**. Menu navigasi untuk CRUD admin ini bisa ditempatkan di bagian footer.

Dalam bab ini kita telah mempelajari studi kasus membuat aplikasi CRUD. Walaupun sederhana, kode programnya sudah cukup kompleks.

Saya juga mempersilahkan anda menggunakan kode program ini untuk tugas, skripsi, aplikasi, dan hal-hal lain. Tulisan *copyright* di bagian footer hanyalah sebagai ‘pemanis’, silahkan jika anda ingin mengubahnya.

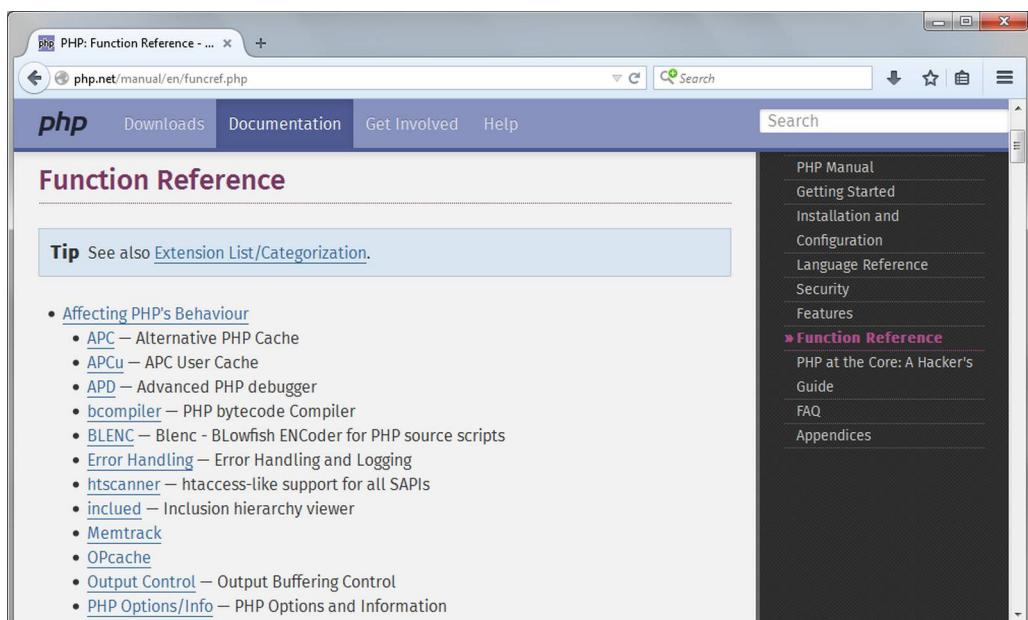
Tentu saja akan jauh lebih baik lagi jika anda bisa mengembangkan aplikasi ini dengan menambahkan berbagai fitur lain atau mengedit tampilannya menggunakan CSS.

# Penutup: PHP Uncover

Tidak terasa, sudah 640 halaman saya membahas PHP di buku **PHP Uncover** ini. Sedikit banyak semoga anda bisa mendapat gambaran tentang apa itu PHP, dan apa yang bisa dilakukan dengan PHP. Walaupun begitu, materi ini barulah ‘segelintir’ dari PHP. Masih banyak materi lanjut yang belum saya bahas.

Sebagai bahan masukan, berikut beberapa hal yang bisa anda lakukan untuk menguasai PHP dengan lebih dalam lagi:

## Pelajari PHP Manual



Gambar: Sub-menu Function Reference di PHP Manual

**PHP Manual** menyimpan berbagai ‘ilmu’ PHP yang sangat bermanfaat, terutama bagian referensi fungsi-fungsi (*Function Reference*). Menurut saya, bagian ini tidak terlalu rumit, karena sudah disertai berbagai contoh kode program.

Memahami fungsi-fungsi bawaan PHP juga sangat penting. Anda akan memiliki ‘senjata tambahan’ sebagai modal untuk memecahkan berbagai masalah yang akan dihadapi sepanjang pembuatan kode program PHP.

Fungsi-fungsi ini juga tidak mesti dihafal, tapi cukup dipahami cara penggunaannya. Jika sudah paham, dengan melihat sekilas *syntax* (aturan cara penulisannya) fungsi tersebut, anda sudah langsung ingat bagaimana cara penggunaannya.

## Perbanyak Latihan kode Program

Jika sebelumnya anda sudah pernah mempelajari HTML dan CSS, gabungkan kedua *skill* ini dengan PHP.

Banyak hal yang bisa dibuat, misalnya: aplikasi ujian online, kamus online, aplikasi kalkulator, bahkan game sederhana. Silahkan berkreasi. Mirip seperti *skill* lain, kemampuan programming akan semakin terasah dengan banyaknya latihan dan memecahkan berbagai masalah.

## Lanjut ke OOP PHP

Object Oriented Programming (OOP) adalah sebuah *paradigma* atau cara pembuatan kode program yang menggunakan konsep berbasis **object**. Biasanya OOP digunakan untuk membuat aplikasi yang cukup rumit dan butuh pengembangan terus menerus.

Saya juga berencana menulis eBook PHP lanjutan yang akan membahas OOP PHP. Tapi mungkin masih butuh waktu.

Jika berminat, silahkan anda cari referensi untuk mulai belajar OOP PHP. Di duniaIlkom juga sudah saya publish [seri tutorial OOP PHP<sup>5</sup>](#), yang walaupun masih dasar, bisa menjadi gambaran tentang pemrograman object di PHP.

## Pelajari Framework PHP



Gambar: Framework PHP, CodeIgniter

**Framework** adalah kumpulan kode program PHP yang dirancang untuk membuat aplikasi dengan cepat dan efisien. Walaupun begitu, mempelajari sebuah framework perlu skill dasar yang cukup. Tidak hanya PHP dasar, umumnya framework PHP menggunakan konsep OOP. Oleh karena itu, sebelum ke framework sebaiknya sudah paham tentang OOP PHP.

<sup>5</sup><http://www.duniaIlkom.com/tutorial-belajar-oop-php-pemrograman-berbasis-objek-php/>

Jika anda ingin berkariir sebagai programmer PHP professional, framework ini wajib dikuasai. Umumnya lowongan kerja programmer PHP mensyaratkan menguasai salah satu framework. Framework PHP yang cukup populer saat ini adalah **Code Igniter, Laravel, Symfony, dan Zend**.

## **Lanjut Mempelajari CSS, MySQL dan JavaScript**

Dalam berbagai kesempatan, saya sering menulis tentang 5 bahasa dasar web programming, yakni: **HTML, CSS, PHP, MySQL, dan JavaScript**. Kelima bahasa pemrograman inilah yang membangun mayoritas website saat ini. Kelimanya juga saling melengkapi dan tidak dapat digantikan oleh yang lain.

Saya sangat sarankan agar anda melengkapi skill yang belum dipelajari. Misalkan anda sudah paham HTML dan PHP, silahkan lanjut ke CSS. Atau bisa juga ke MySQL dan JavaScript.

Saat ini di duniaIlkom baru tersedia eBook **HTML, CSS dan PHP**. Kedepannya, saya juga akan menulis buku tentang **JavaScript** dan **MySQL**. Tapi anda juga dipersilahkan mencari buku lain sebagai alternatif (sambil menunggu materi tersebut selesai saya tulis).

---

Akhir kata, semoga materi yang ada di buku PHP Uncover ini bisa bermanfaat. Mohon maaf jika ada kata-kata yang salah.

Terimakasih atas dukungannya dengan membeli versi asli **eBook PHP Uncover**. Sampai jumpa di buku DuniaIlkom selanjutnya :)