

MODUL - WEB PROGRAMMING 1

Oleh :
Priyanto Tamami, S.Kom.

Daftar Isi

1	Dasar-Dasar Pemrograman PHP	1
1.1	Tujuan	1
1.2	Pengantar	1
1.3	Praktek	2
1.3.1	Instalasi XAMPP	2
1.3.2	Instalasi Git dan Akun Github	9
1.3.3	Instalasi Visual Studio Code	24
1.3.4	Registrasi 000webhost.com	31
1.3.5	Aplikasi Selamat Datang	34
1.3.6	Unggah ke Github	40
1.4	Kesimpulan	42
1.5	Tugas	42
2	Operator, Variabel, dan Tipe Data	43
2.1	Tujuan	43
2.2	Pengantar	43
2.3	Praktek	43
2.3.1	Variabel	43
2.3.2	Tipe Data	48
2.3.3	Operator	50
2.4	Kesimpulan	61
2.5	Tugas	61
3	Array	63
3.1	Tujuan	63
3.2	Pengantar	63
3.3	Praktek	63
3.4	Kesimpulan	65
3.5	Tugas	65
4	Struktur Kontrol	67
4.1	Tujuan	67
4.2	Pengantar	67
4.3	Praktek	67
4.3.1	Seleksi / Percabangan if	67
4.3.2	Seleksi / Percabangan switch	70

4.3.3	Perulangan <code>while</code>	71
4.3.4	Perulangan <code>do-while</code>	72
4.3.5	Perulangan <code>for</code>	73
4.3.6	Perulangan <code>foreach</code>	74
4.3.7	Kontrol <code>break</code>	76
4.3.8	Kontrol <code>continue</code>	82
4.3.9	Kontrol <code>include</code> dan <code>require</code>	86
4.3.10	Kontrol <code>goto</code>	89
4.4	Kesimpulan	90
4.5	Tugas	90
5	Fungsi, Modularisasi, dan Pemrosesan <i>File</i>	93
5.1	Tujuan	93
5.2	Pengantar	93
5.3	Praktek	93
5.3.1	Fungsi	93
5.3.2	Kelas dan Objek	96
5.3.3	Pemrosesan <i>File</i>	101
5.4	Kesimpulan	107
5.5	Tugas	107
6	Konektivitas PHP dan Basis Data	109
6.1	Tujuan	109
6.2	Pengantar	109
6.3	Praktek	109
6.3.1	Memulai Koneksi	109
6.3.2	Masukkan Data	112
6.3.3	Ambil Data	116
6.3.4	Perbarui Data	119
6.3.5	Delete Data	124
6.4	Kesimpulan	127
6.5	Tugas	127
7	Session	129
7.1	Tujuan	129
7.2	Pengantar	129
7.3	Praktek	129
7.3.1	Memulai Session dan Membuat Variabel <i>Session</i>	129
7.3.2	Mengakses Variabel <i>Session</i>	130
7.3.3	Merubah Variabel <i>Session</i>	132
7.3.4	Menutup <i>Session</i>	133
7.4	Kesimpulan	134
7.5	Tugas	135

8 Penggunaan Form	137
8.1 Tujuan	137
8.2 Pengantar	137
8.3 Praktek	137
8.3.1 Dengan Metode <i>GET</i>	137
8.3.2 Dengan Metode <i>POST</i>	139
8.4 Kesimpulan	141
8.5 Tugas	141
9 Operasi Aritmatika	143
9.1 Tujuan	143
9.2 Pengantar	143
9.3 Praktek	143
9.3.1 Fungsi <i>abs()</i>	143
9.3.2 Fungsi <i>bindec()</i>	144
9.3.3 Fungsi <i>ceil()</i>	145
9.3.4 Fungsi <i>decbin()</i>	146
9.3.5 Fungsi <i>dechex()</i>	147
9.3.6 Fungsi <i>decoct()</i>	148
9.3.7 Fungsi <i>deg2rad()</i>	149
9.3.8 Fungsi <i>floor()</i>	149
9.3.9 Fungsi <i>fmod()</i>	150
9.3.10 Fungsi <i>getrandmax()</i>	151
9.3.11 Fungsi <i>hexdec()</i>	151
9.3.12 Fungsi <i>is_finite()</i>	152
9.3.13 Fungsi <i>is_infinite()</i>	153
9.3.14 Fungsi <i>max()</i>	153
9.3.15 Fungsi <i>min()</i>	154
9.3.16 Fungsi <i>octdec()</i>	155
9.3.17 Fungsi <i>pow()</i>	155
9.3.18 Fungsi <i>rad2deg()</i>	156
9.3.19 Fungsi <i>rand()</i>	157
9.3.20 Fungsi <i>round()</i>	157
9.4 Kesimpulan	158
9.5 Tugas	158
10 Operasi String	159
10.1 Tujuan	159
10.2 Pengantar	159
10.3 Praktek	159
10.3.1 Fungsi <i>bin2hex()</i>	159
10.3.2 Fungsi <i>chop()</i>	160
10.3.3 Fungsi <i>chr()</i>	161
10.3.4 Fungsi <i>chunk_split()</i>	161
10.3.5 Fungsi <i>count_chars()</i>	162

10.3.6 Fungsi <code>crypt()</code>	163
10.3.7 Fungsi <code>explode()</code>	164
10.3.8 Fungsi <code>implode()</code>	165
10.3.9 Fungsi <code>lcfirst()</code>	166
10.3.10 Fungsi <code>ltrim()</code>	166
10.3.11 Fungsi <code>md5()</code>	167
10.3.12 Fungsi <code>number_format()</code>	168
10.3.13 Fungsi <code>rtrim()</code>	168
10.3.14 Fungsi <code>sha1()</code>	169
10.3.15 Fungsi <code>str_pad()</code>	170
10.3.16 Fungsi <code>str_split()</code>	171
10.3.17 Fungsi <code>str_word_count()</code>	172
10.3.18 Fungsi <code>strlen()</code>	172
10.3.19 Fungsi <code>strtolower()</code>	173
10.3.20 Fungsi <code>strtoupper()</code>	174
10.3.21 Fungsi <code>trim()</code>	174
10.3.22 Fungsi <code>ucfirst()</code>	175
10.3.23 Fungsi <code>ucwords()</code>	176
10.3.24 Fungsi <code>wordwrap()</code>	176
10.4 Kesimpulan	177
10.5 Tugas	177
11 HTML, XHTML, dan DHTML	179
11.1 Tujuan	179
11.2 Pengantar	179
11.3 Praktek	179
11.3.1 HTML	179
11.3.2 XHTML	181
11.3.3 DHTML	183
11.4 Kesimpulan	184
11.5 Tugas	185
12 CSS	187
12.1 Tujuan	187
12.2 Pengantar	187
12.3 Praktek	187
12.4 Kesimpulan	190
12.5 Tugas	190
13 Production	191
13.1 Tujuan	191
13.2 Pengantar	191
13.3 Praktek	191
13.4 Kesimpulan	194
13.5 Tugas	194

Bab 1

Dasar-Dasar Pemrograman PHP

1.1 Tujuan

Pada Bab ini diharapkan mahasiswa mengenal bentuk *syntax* dasar dari bahasa pemrograman PHP dan lingkungan kerja yang mendukungnya.

1.2 Pengantar

PHP adalah bahasa pemrograman yang dijalankan di sisi *server* dalam bentuk *scripting*, artinya bahasa pemrograman ini tidak perlu di *compile* terlebih dahulu untuk dapat dijalankan, kita cukup menyiapkan *interpreter*-nya saja.

PHP biasanya digunakan untuk membangun sebuah aplikasi Web yang dinamis, dimana halaman dapat melakukan respon terhadap *request* yang dilakukan oleh pengguna.

PHP pun telah digunakan secara luas dan menjadi alternatif gratis dibandingkan menggunakan bahasa sejenis seperti ASP milik Microsoft.

Untuk memulai melakukan praktik bahasa pemrograman menggunakan PHP, maka kita perlu mempersiapkan perangkat pendukung. Cara yang mungkin paling mudah adalah kita menggunakan aplikasi paket yang di dalamnya sudah terdapat *web server* yang mendukung PHP serta basis data yang akan digunakan, aplikasi yang mungkin dapat kita gunakan adalah :

1. LAMP, yang sebetulnya adalah singkatan dari Linux, Apache, MySQL, dan PHP. Tentunya aplikasi ini ditujukan untuk sistem operasi Linux, yang menggunakan Apache sebagai *web server* yang tentunya *plugin* untuk mendukung bahasa PHP sudah ada di dalamnya, dan MySQL sebagai basis datanya.
2. WAMP, adalah singkatan dari Windows, Apache, MySQL, dan PHP. Mirip seperti LAMP, hanya ini ditujukan bagi sistem operasi Windows.
3. MAMP, adalah singkatan dari Mac, Apache, MySQL, dan PHP. Untuk aplikasi ini dikhawasukan bagi sistem operasi Mac.

4. XAMPP, yang ini mendukung ketiga sistem operasi di atas dengan kelebihan mampu untuk mengolah bahasa pemrograman Perl.

Maka pilihan untuk praktik Web Programming 1 kita akan menggunakan XAMPP agar adaptasi antar sistem operasi lebih mudah. XAMPP dapat diunduh pada alamat <https://www.apachefriends.org>.

Aplikasi pendukung lain untuk melakukan kegiatan praktikum kita adalah sebagai berikut :

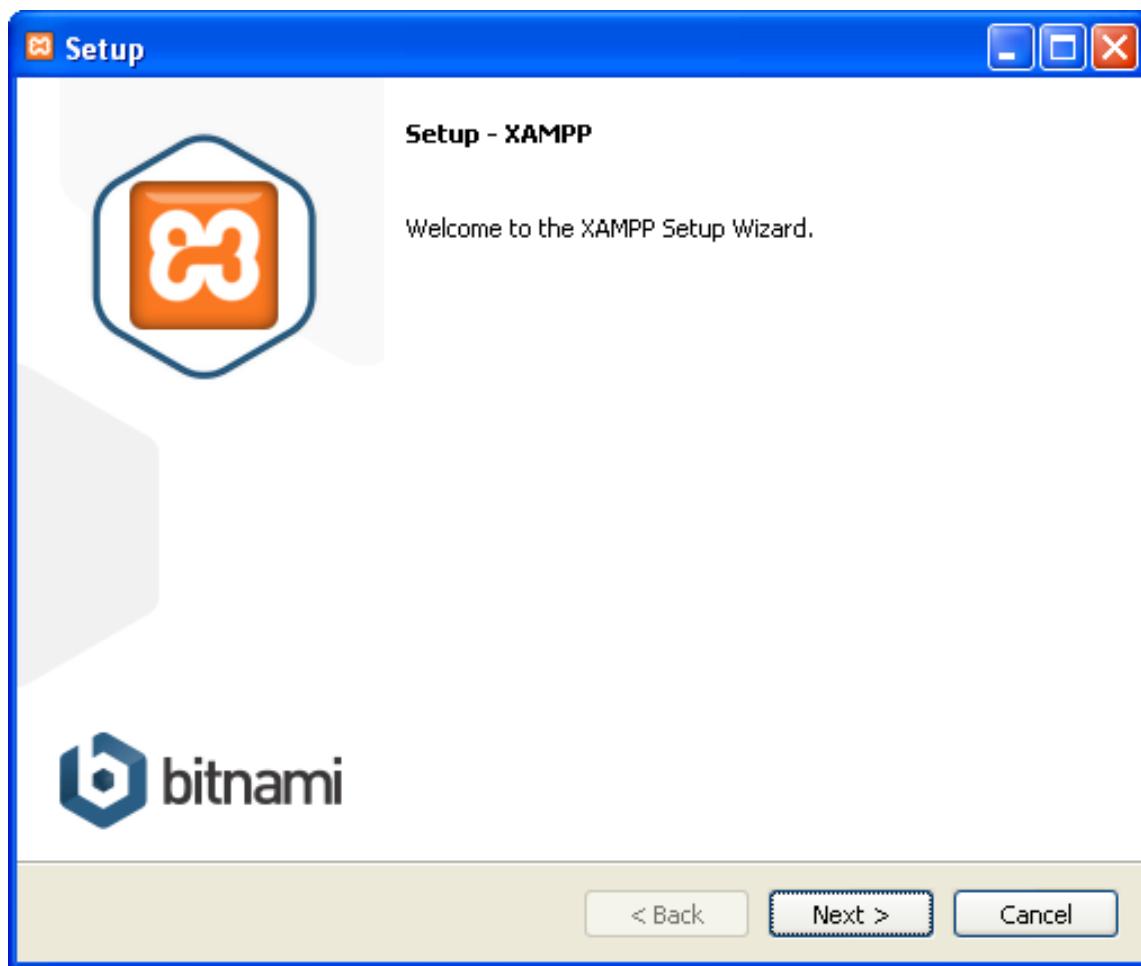
1. Git. Aplikasi ini digunakan untuk melakukan *versioning* sehingga kita lebih mudah dalam melakukan kontrol perubahan yang terjadi pada kode program yang kita bangun. Server yang kita gunakan untuk menyimpan repositori hasil *versioning* kita ada di alamat <https://github.com>. Github ini gratis. Untuk aplikasi Git dapat kita unduh di alamat <https://git-scm.com/>
2. Visual Studio Code. Aplikasi ini adalah *editor* yang akan digunakan dalam kegiatan praktikum pada mata kuliah Web Programming 1. Aplikasi ini gratis dan dapat diunduh pada alamat <https://code.visualstudio.com/> dengan dukungan instalasi untuk 3 (tiga) sistem operasi yang banyak digunakan, yaitu Linux, Windows, dan MacOS.
3. www.000webhost.com. Ini adalah layanan *hosting* gratis yang mampu menjalankan *script* PHP dengan fasilitas sistem basis data MySQL. Yang akan kita gunakan sebagai tempat aplikasi yang telah kita bangun sampai dengan akhir tatap muka mata kuliah ini.

1.3 Praktek

1.3.1 Instalasi XAMPP

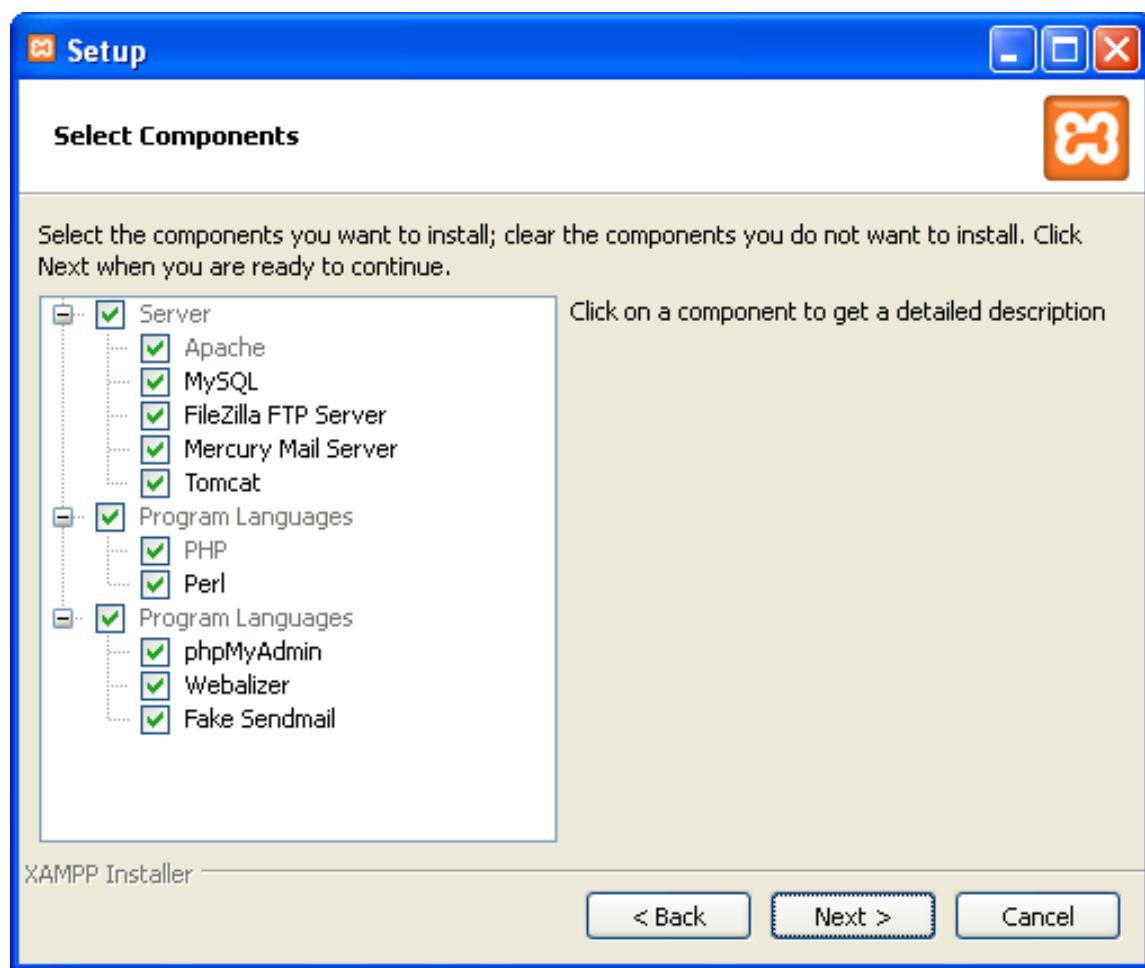
Langkah-langkah untuk instalasi XAMPP adalah sebagai berikut :

1. Tentunya mengunduh *installer* aplikasi ini dari alamat <https://www.apachefriends.org>, pilihlah yang model *installer* agar lebih mudah proses instalasinya.
2. Awal proses instalasi, kita akan bertemu dengan tampilan seperti pada gambar 1.1 berikut ini :



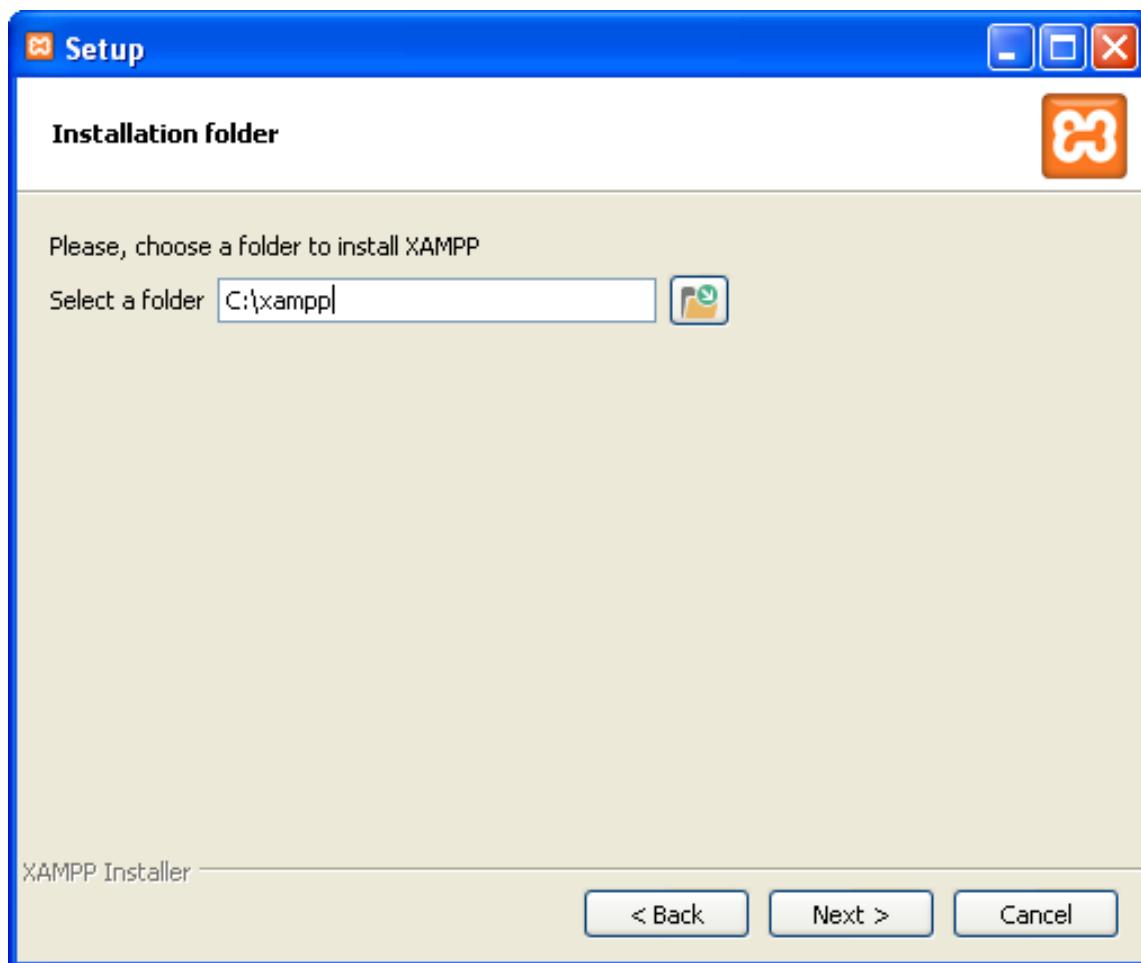
Gambar 1.1: Awal Instalasi XAMPP

3. Setelah itu kita diminta untuk memilih komponen yang akan ikut diinstall / dipasangkan seperti pada gambar 1.2 berikut :



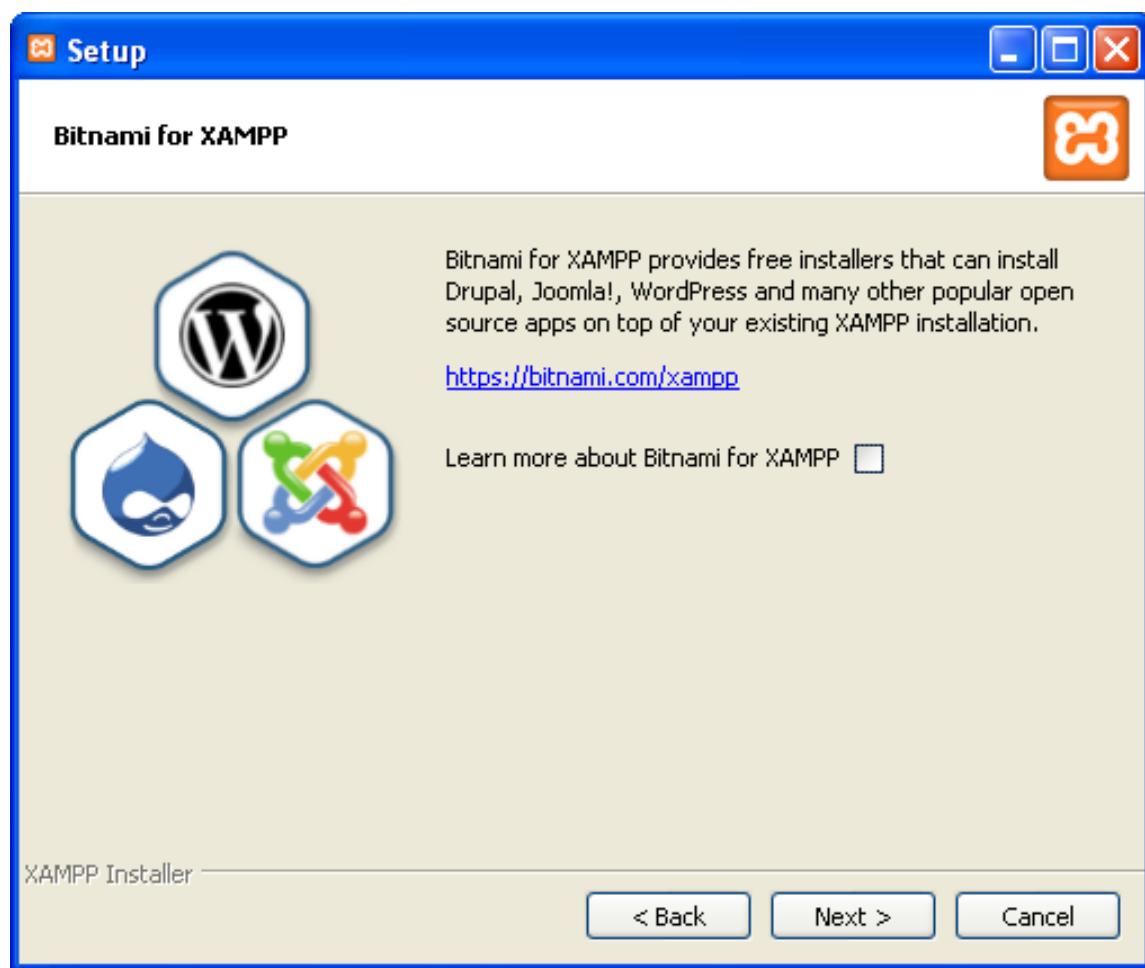
Gambar 1.2: Pemilihan Komponen

4. Selanjutnya adalah penetapan *folder* instalasi, akan ditaruh dimana hasil instalasi dari XAMPP ini. Tampilannya seperti pada gambar 1.26 berikut ini :



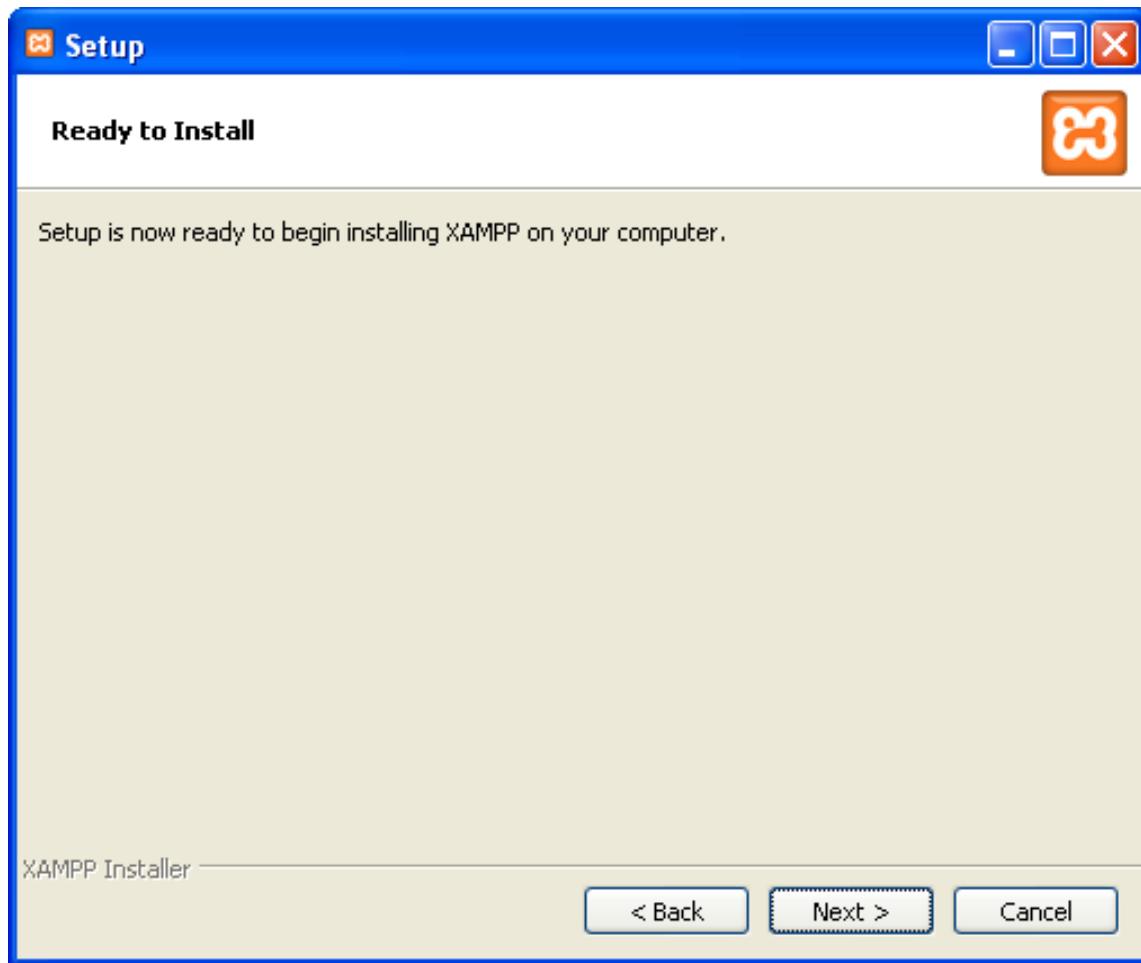
Gambar 1.3: Pemilihan *Folder* Instalasi

5. Langkah berikutnya adalah informasi mengenai Bitnami, Bitnami ini adalah aplikasi gratis yang mampu melakukan *instalasi* aplikasi lain semacam Drupal atau Joomla di atas XAMPP, tampilannya seperti pada gambar 1.4 berikut :



Gambar 1.4: Informasi Bitnami

6. Setelah menekan tombol **Next**, maka akan muncul informasi bahwa aplikasi siap diinstall seperti pada gambar 1.5 berikut :



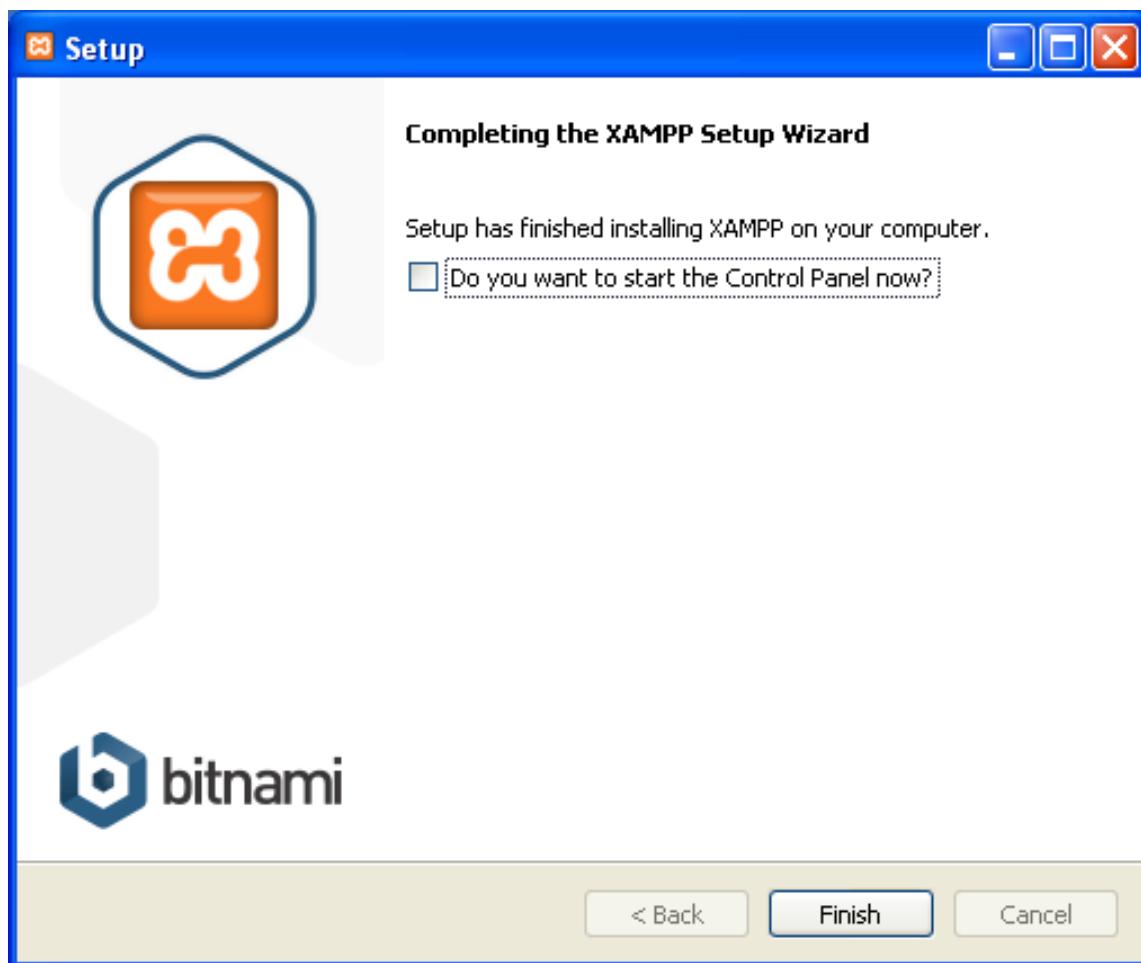
Gambar 1.5: Instalasi Sudah Siap

7. Setelah itu proses instalasi akan berjalan seperti pada gambar 1.6 berikut :



Gambar 1.6: Instalasi berjalan

8. Setelah proses instalasi selesai, maka akan muncul informasi seperti pada gambar 1.7 berikut :



Gambar 1.7: Instalasi Selesai

Sampai sini, proses instalasi XAMPP telah berhasil dan siap untuk digunakan.

1.3.2 Instalasi Git dan Akun Github

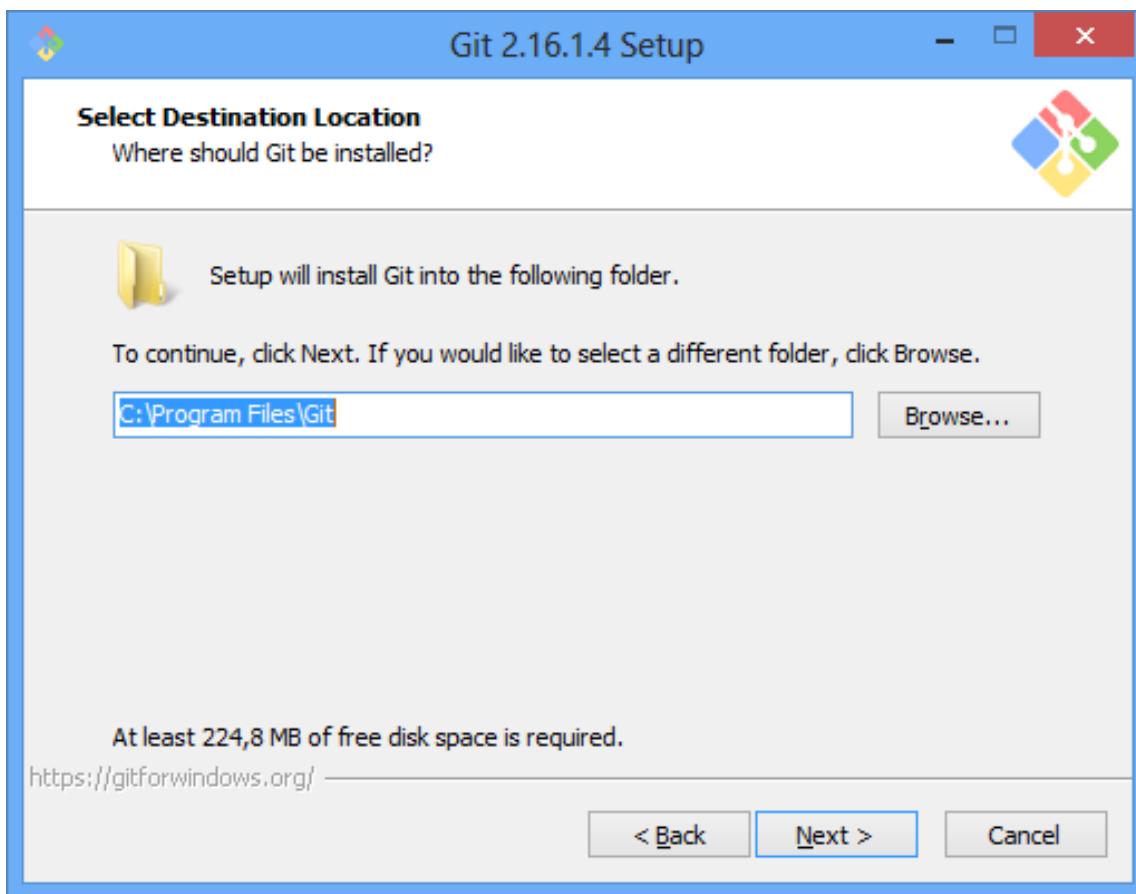
Proses instalasi dari Git adalah sebagai berikut :

1. Awalnya akan disajikan informasi mengenai lisensi seperti pada gambar 1.8 berikut :



Gambar 1.8: Informasi Licensi

2. Berikutnya akan ditampilkan lokasi dimana aplikasi Git akan diinstall, tampilannya adalah seperti pada gambar 1.9 berikut :



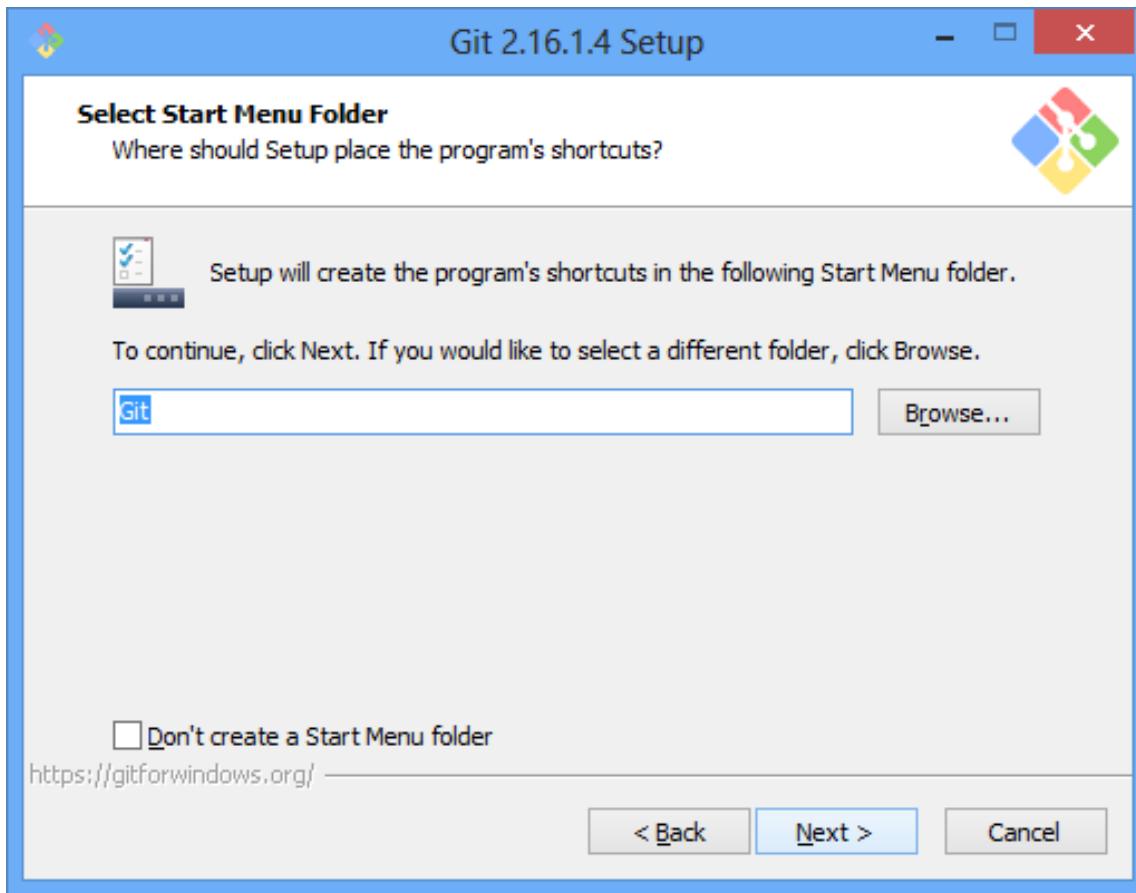
Gambar 1.9: Lokasi Tempat Instalasi Git

3. Berikutnya adalah pemilihan komponen yang akan diinstall di komputer, tampilannya adalah seperti pada gambar 1.10 berikut :



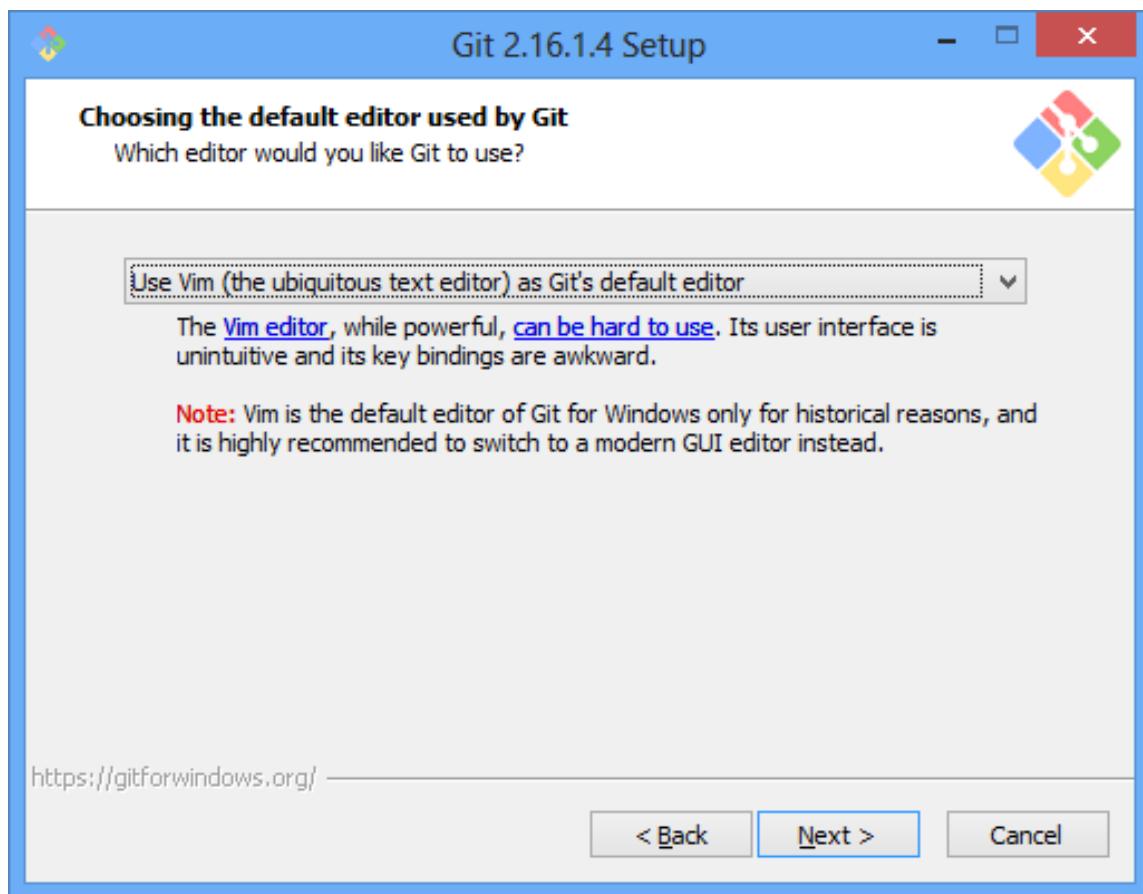
Gambar 1.10: Pemilihan Komponen

4. Selanjutnya adalah lokasi pada **start menu** dimana Git akan ditempatkan, tampilannya adalah seperti pada gambar 1.11 berikut :



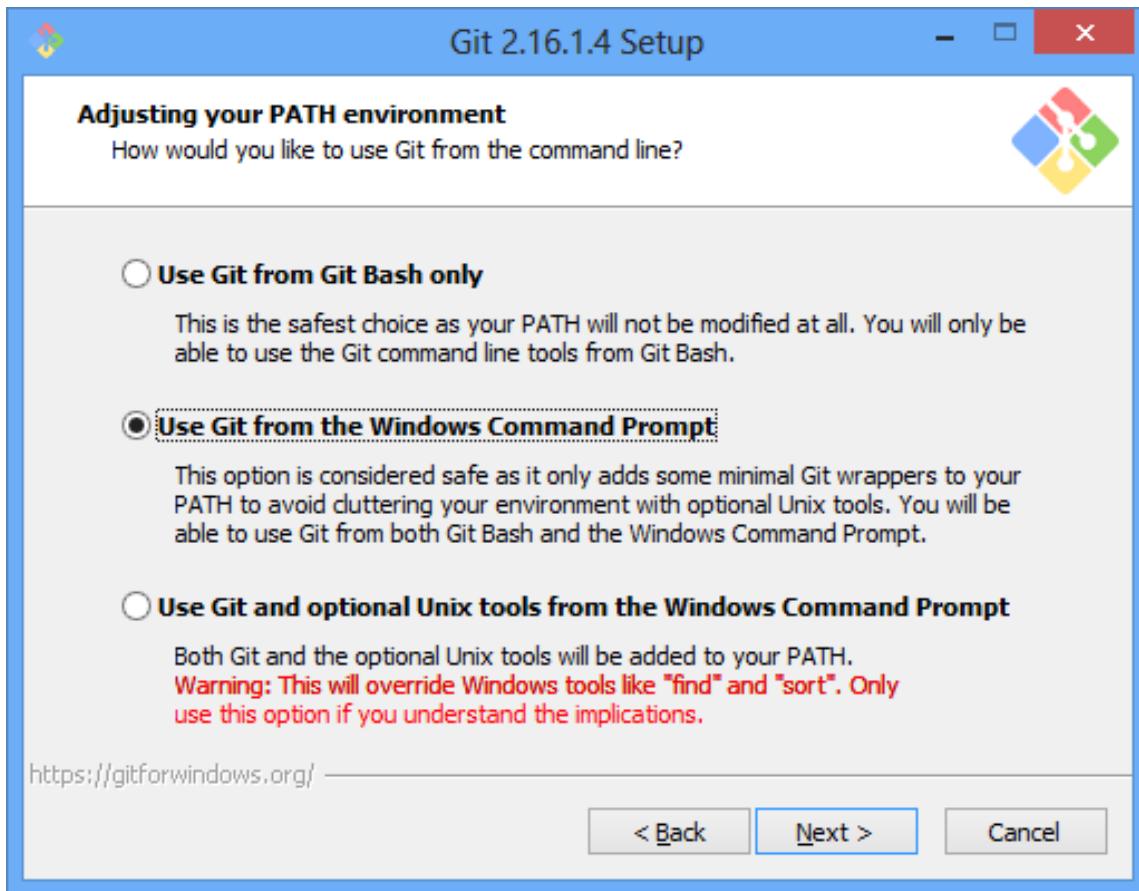
Gambar 1.11: Lokasi di Start Menu

5. Setelah itu kita diminta untuk memilih *editor* yang digunakan oleh Git seperti pada gambar 1.12 berikut :



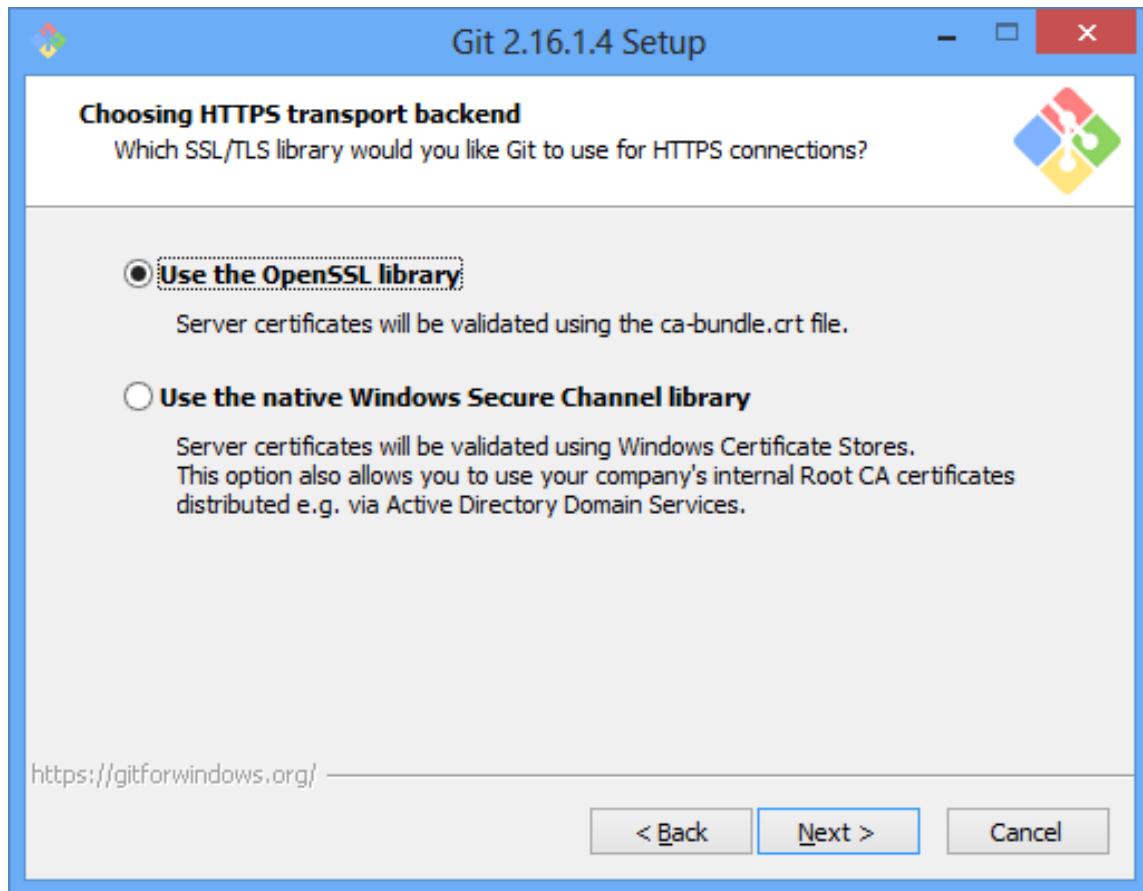
Gambar 1.12: Pemilihan *Editor*

6. Langkah berikutnya adalah mengatur *environment variable* PATH untuk penggunaan Git melalui *console* atau *command prompt*, tampilannya adalah seperti pada gambar 1.13 berikut :



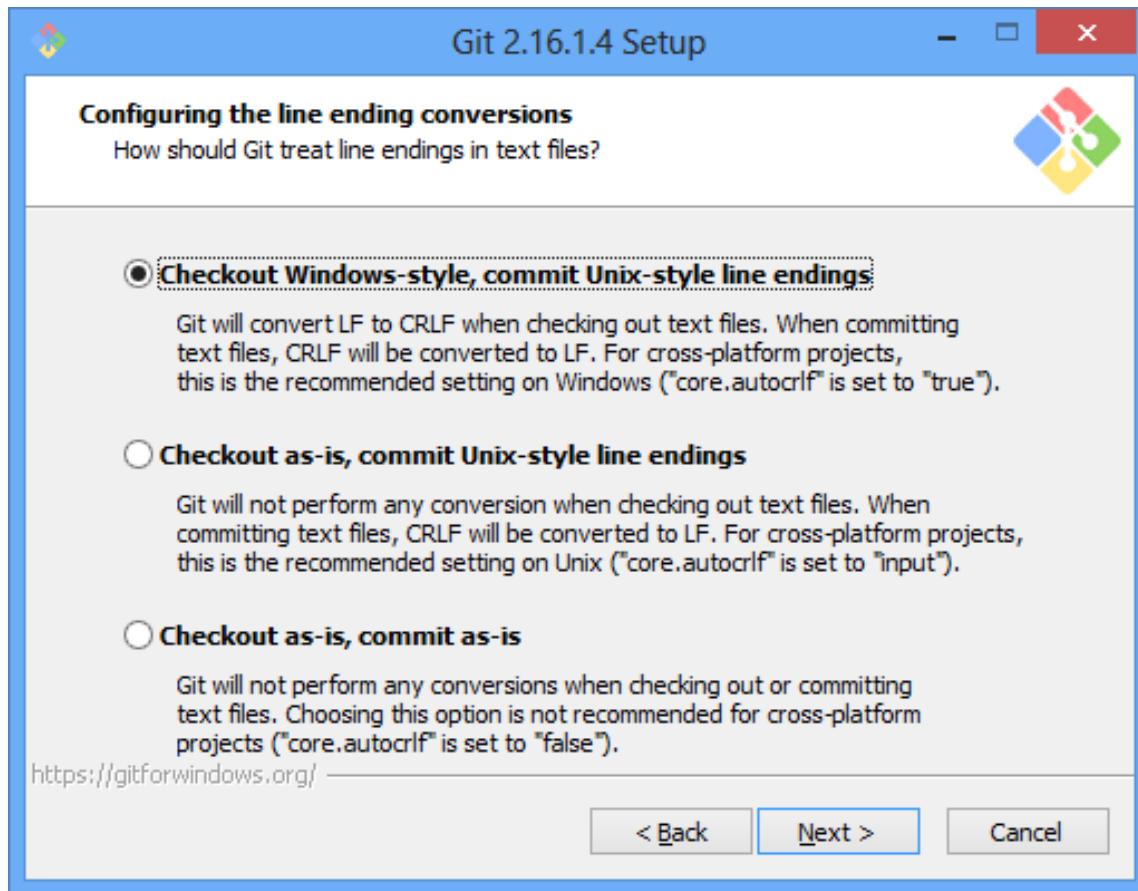
Gambar 1.13: Pengaturan PATH

7. Selanjutnya adalah memilih pustaka SSL yang akan digunakan untuk koneksi HTTPS, tampilannya adalah seperti pada gambar 1.14 berikut :



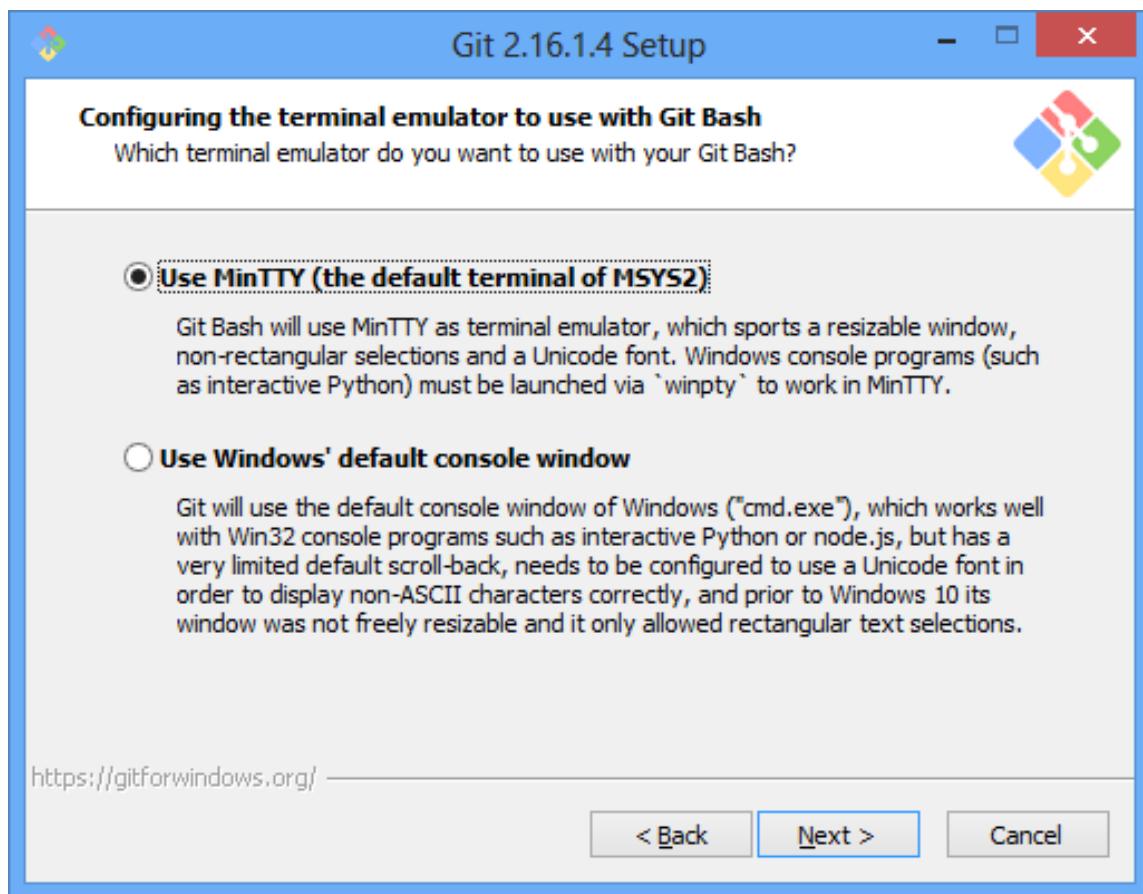
Gambar 1.14: Pemilihan Pustaka SSL

8. Berikutnya adalah memilih bagaimana cara Git memperlakukan *line endings*, tampilannya adalah seperti pada gambar 1.15 berikut :



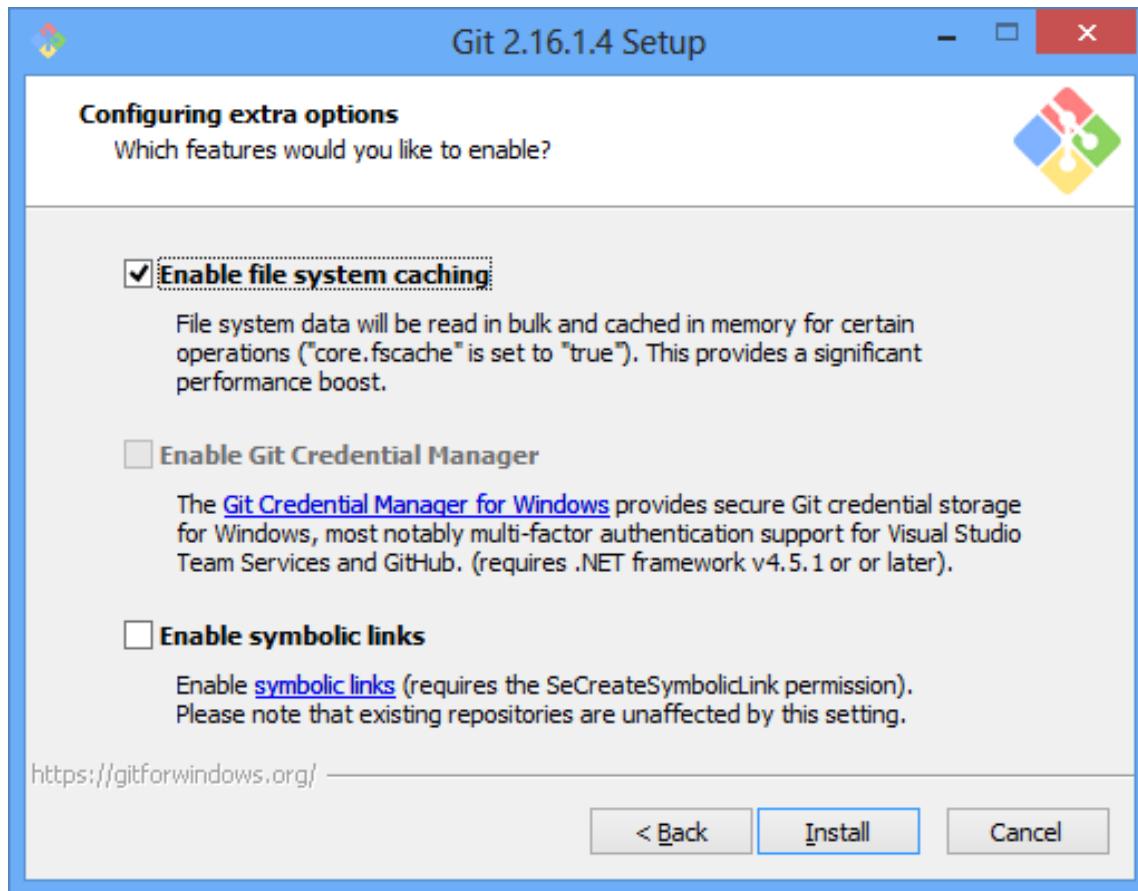
Gambar 1.15: Pemilihan Konversi *Line Ending*

9. Selanjutnya adalah pemilihan emulator terminal yang akan digunakan oleh Git, tampilannya seperti pada gambar 1.16 berikut :



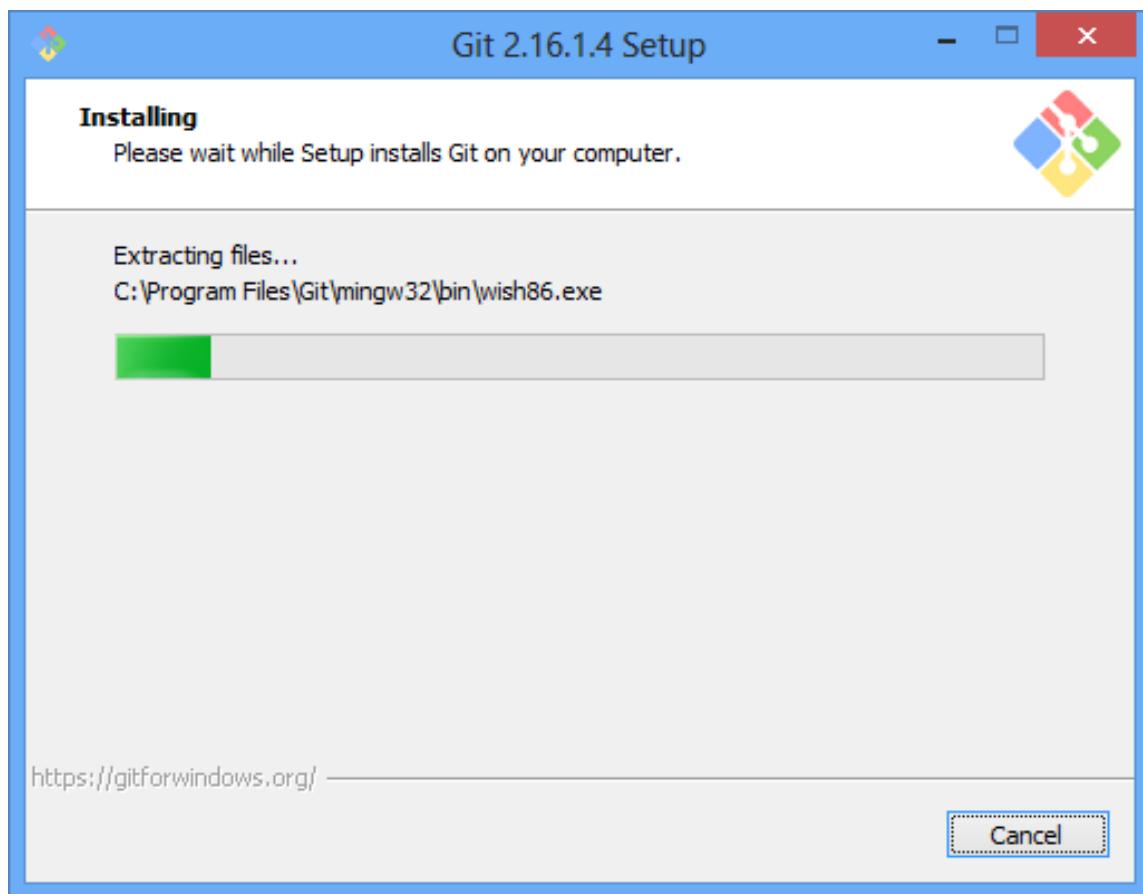
Gambar 1.16: Pemilihan Emulator Terminal

10. Langkah berikutnya adalah memilih fasilitas yang akan diaktifkan, tampilannya akan terlihat seperti pada gambar 1.17 berikut :



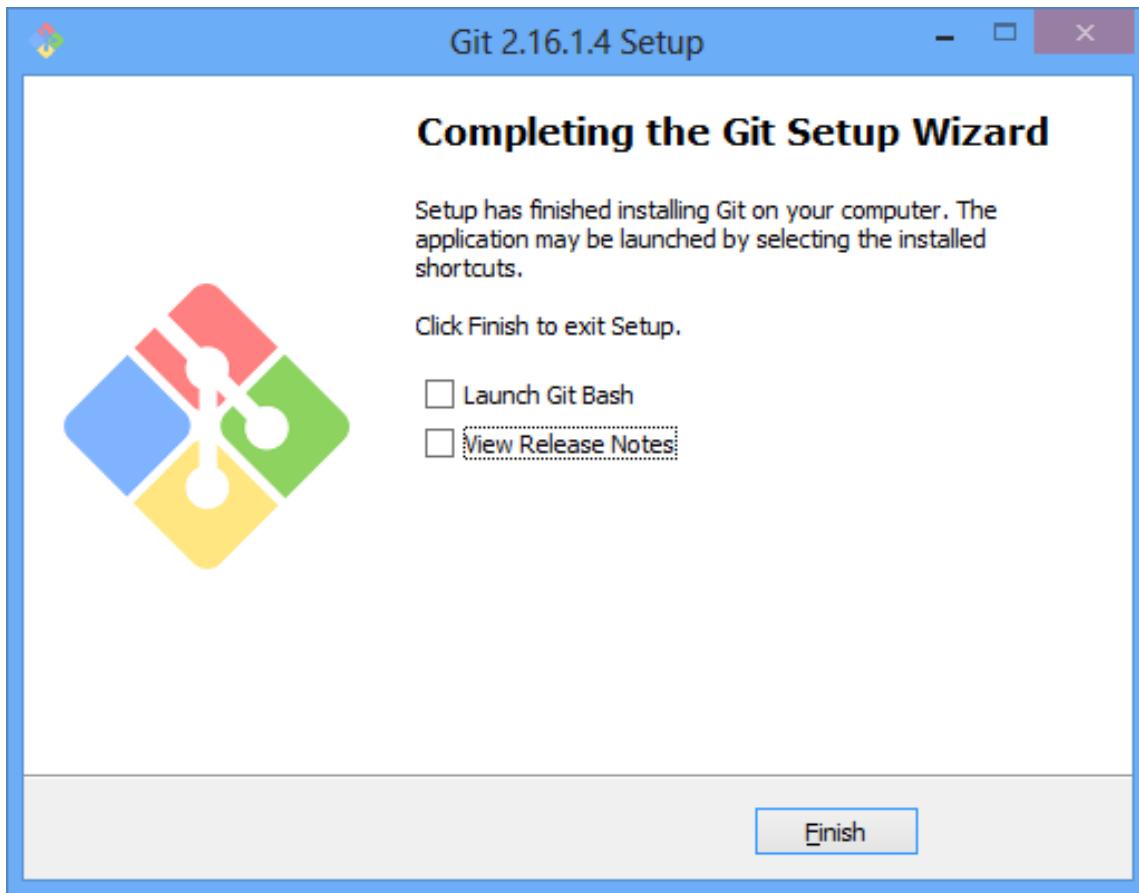
Gambar 1.17: Pemilihan Fasilitas Tambahan

11. Setelah menekan tombol **Install**, maka proses instalasi akan berjalan seperti pada gambar 1.18 berikut :



Gambar 1.18: Proses Instalasi

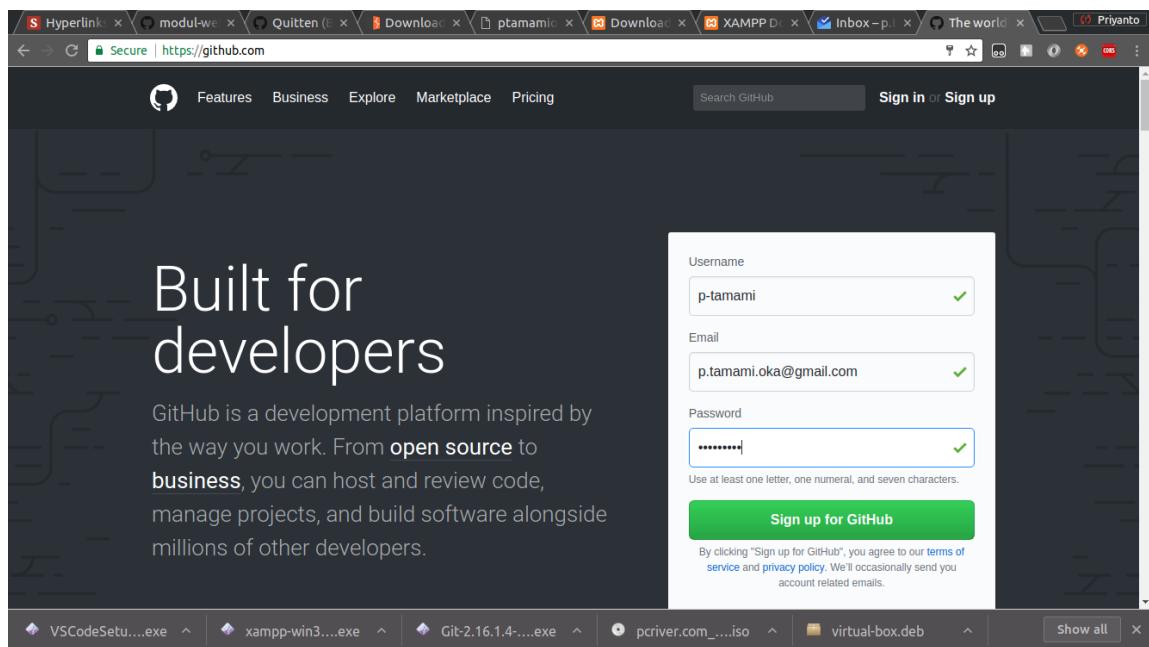
12. Setelah proses instalasi selesai, maka akan muncul jendela seperti pada gambar 1.19 berikut :



Gambar 1.19: Instalasi Selesai

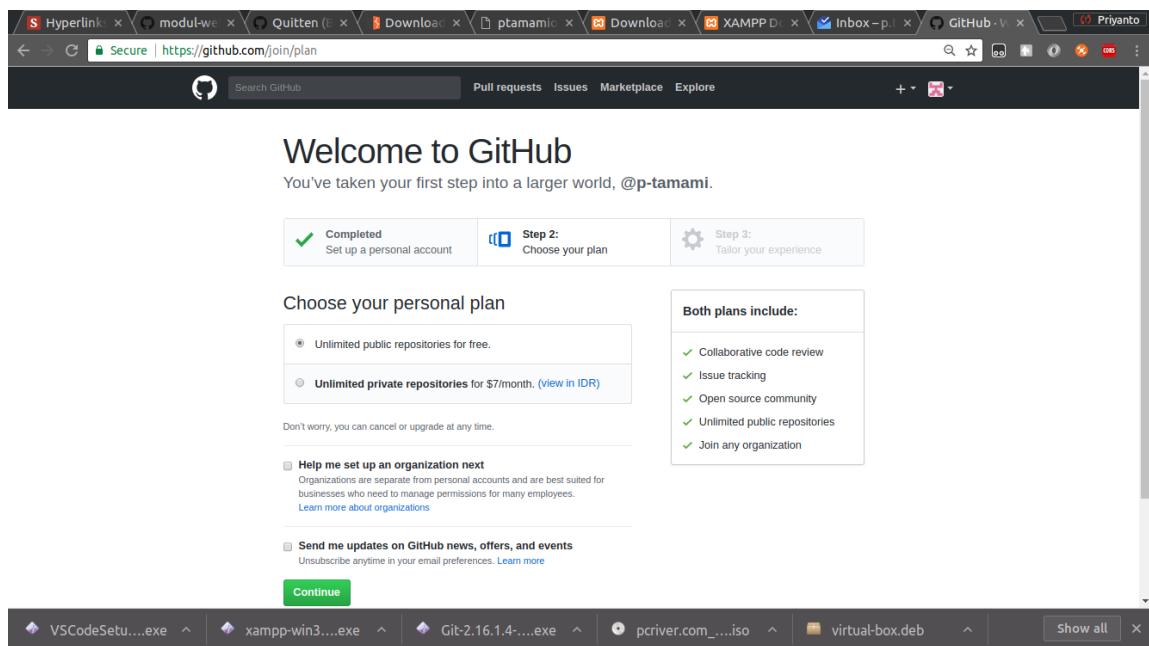
Berikutnya kita akan bahas bagaimana caranya membuat akun di Github agar kita dapat menyimpan kode yang telah kita bangun di repositori Github. Langkah-langkahnya adalah sebagai berikut :

1. Bukalah *browser* dan isikan alamatnya dengan **github.com**, sehingga nanti akan muncul tampilan seperti pada gambar 1.20 berikut :



Gambar 1.20: Halaman Depan Github

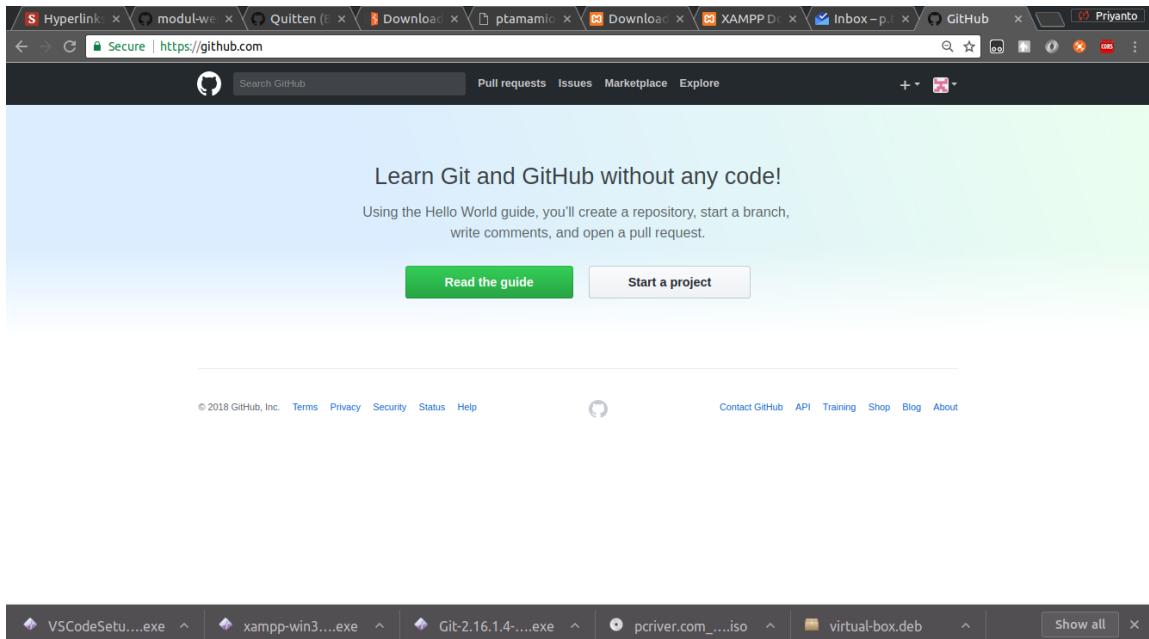
- Setelah mengisikan *username*, alamat *email*, dan *password*, kemudian menekan tombol **Sign up for Github**, maka akan tampilan halaman berikutnya seperti pada gambar 1.21 berikut :



Gambar 1.21: Halaman Selamat Datang Github

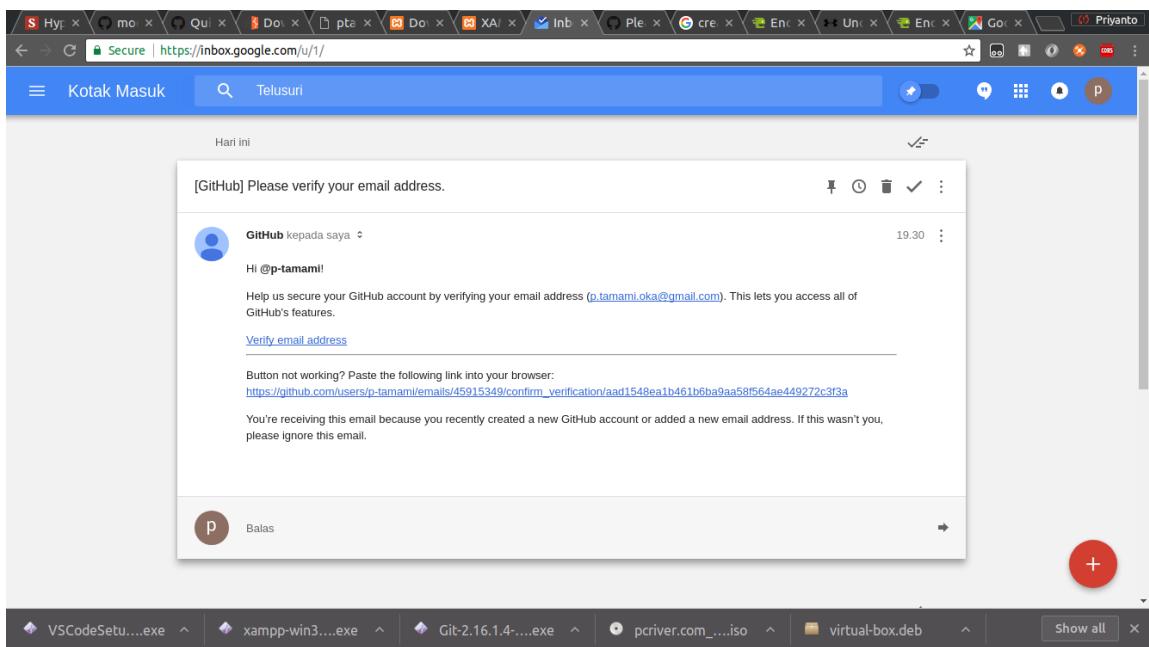
- Karena kita menggunakan versi gratis, jadi lanjutkan saja dengan menekan tombol **Continue** sehingga muncul tampilan berikutnya seperti pada gambar

1.22 :



Gambar 1.22: Halaman Pembuka Github

- Selanjutnya, untuk membuat repositori, kita diwajibkan melakukan verifikasi alamat *email*, bukalah *email* yang kita gunakan untuk mendaftar Github, lalu klik *link* yang bertuliskan *Verify email address* seperti pada gambar 1.23 berikut :

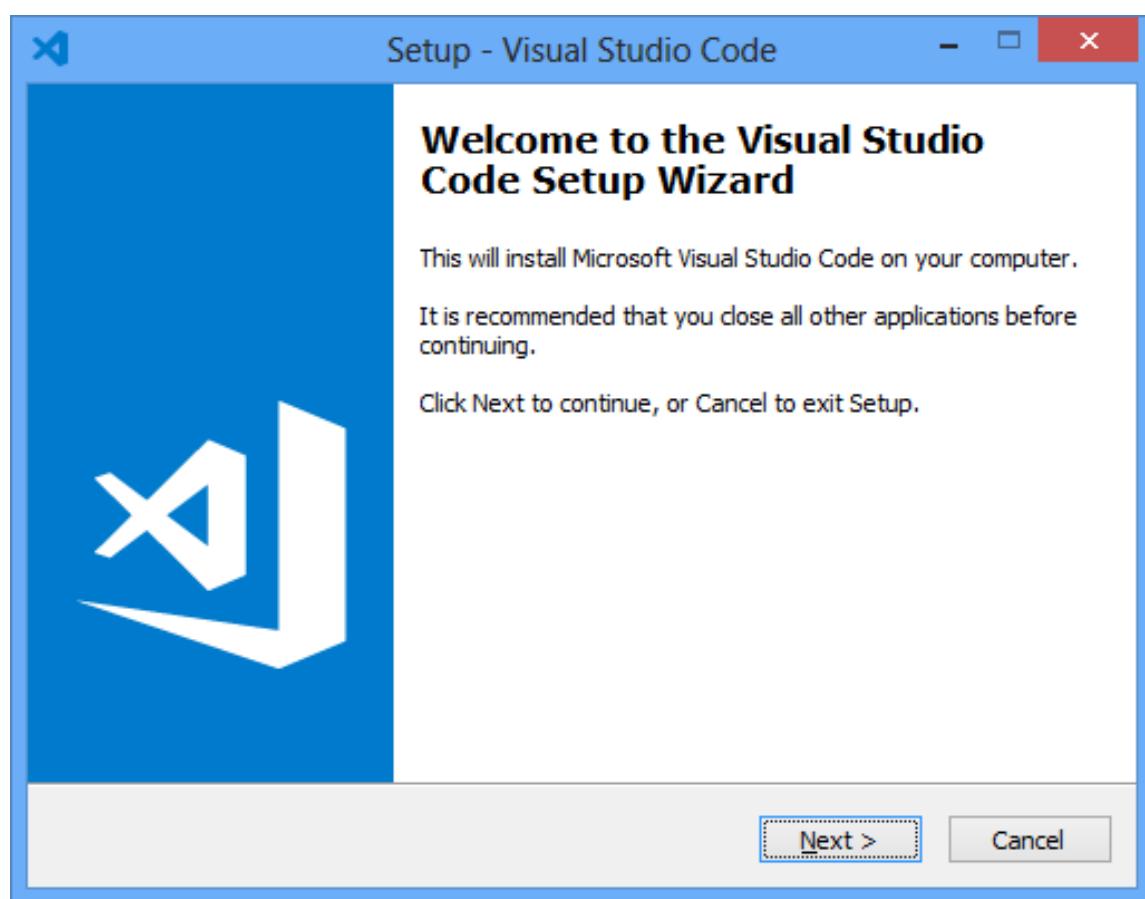
Gambar 1.23: Verifikasi Alamat *Email*

Sampai langkah ini Github sudah dapat kita gunakan untuk menyimpan kode program yang kita bangun.

1.3.3 Instalasi Visual Studio Code

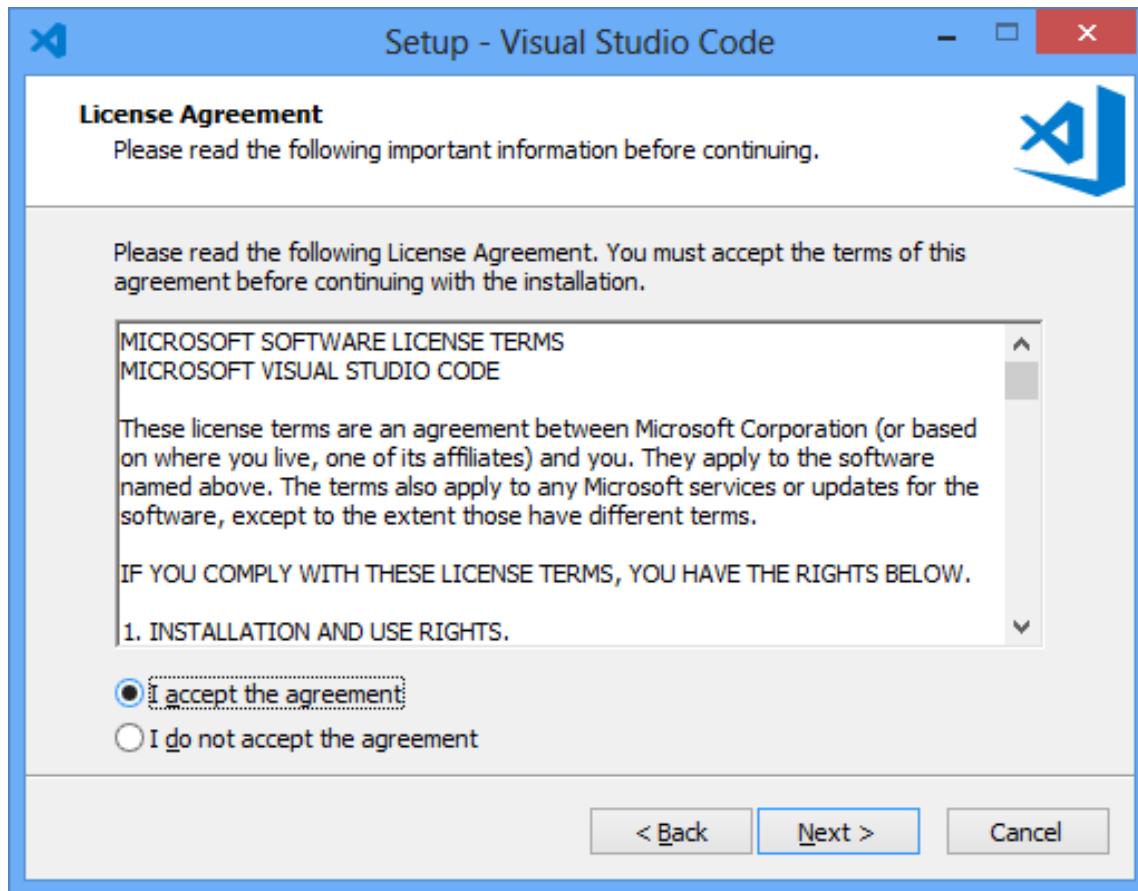
Proses instalasi Visual Studio Code adalah sebagai berikut :

1. Pertama kita akan disajikan jendela selamat datang seperti biasa, ditunjukkan oleh gambar 1.24 berikut :



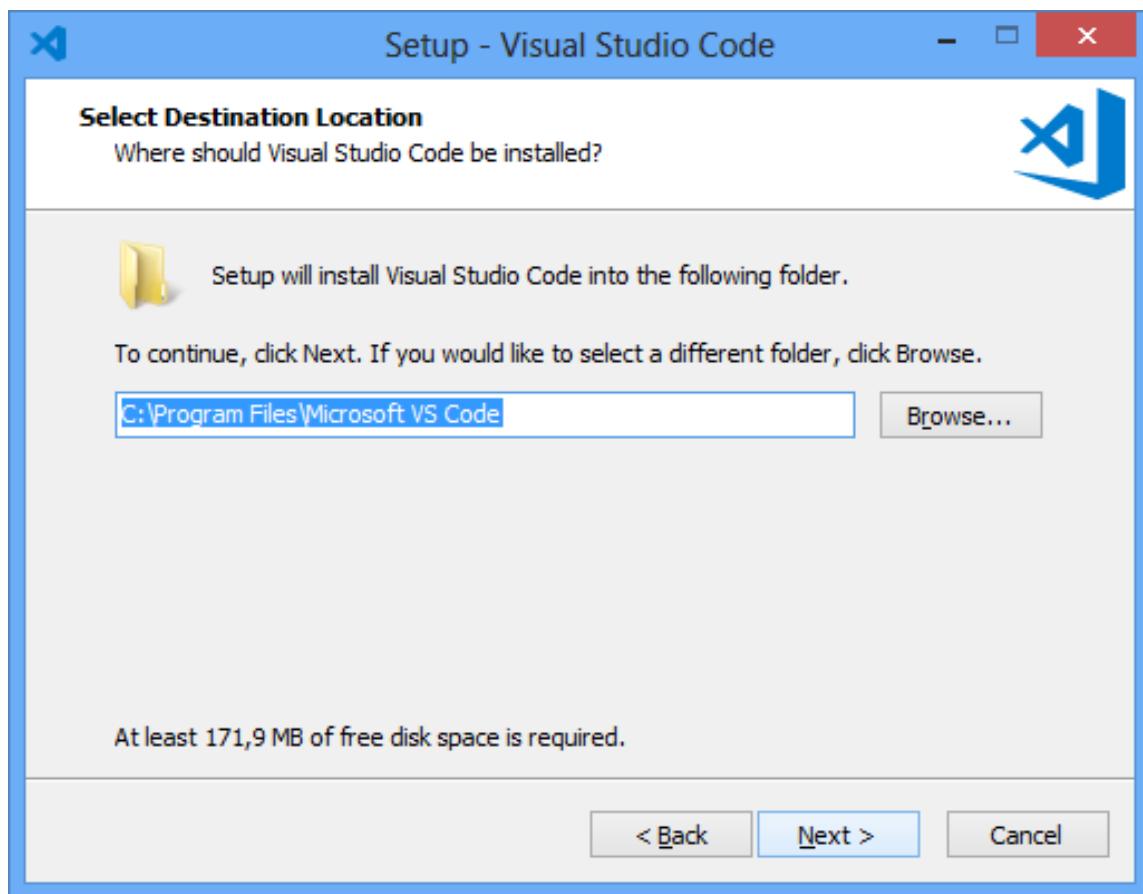
Gambar 1.24: Halaman Selamat Datang dari Visual Studio Code

2. Jendela berikutnya akan menampilkan informasi mengenai lisensi yang digunakan, tampilannya akan terlihat seperti gambar 1.25 berikut :



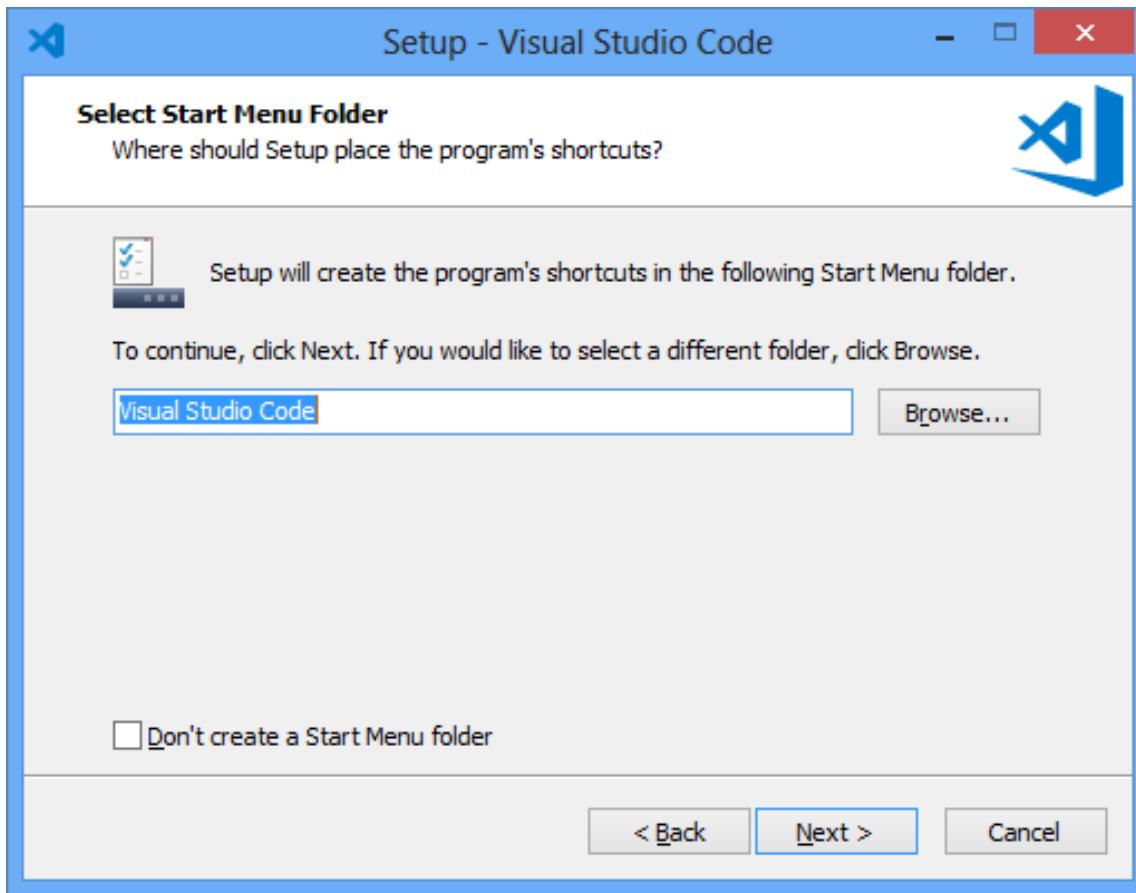
Gambar 1.25: Informasi Licensi

3. Selanjutnya adalah menentukan lokasi direktori dimana *Visual Studio Code* akan ditempatkan, tampilannya adalah seperti pada gambar 1.26 berikut ini :



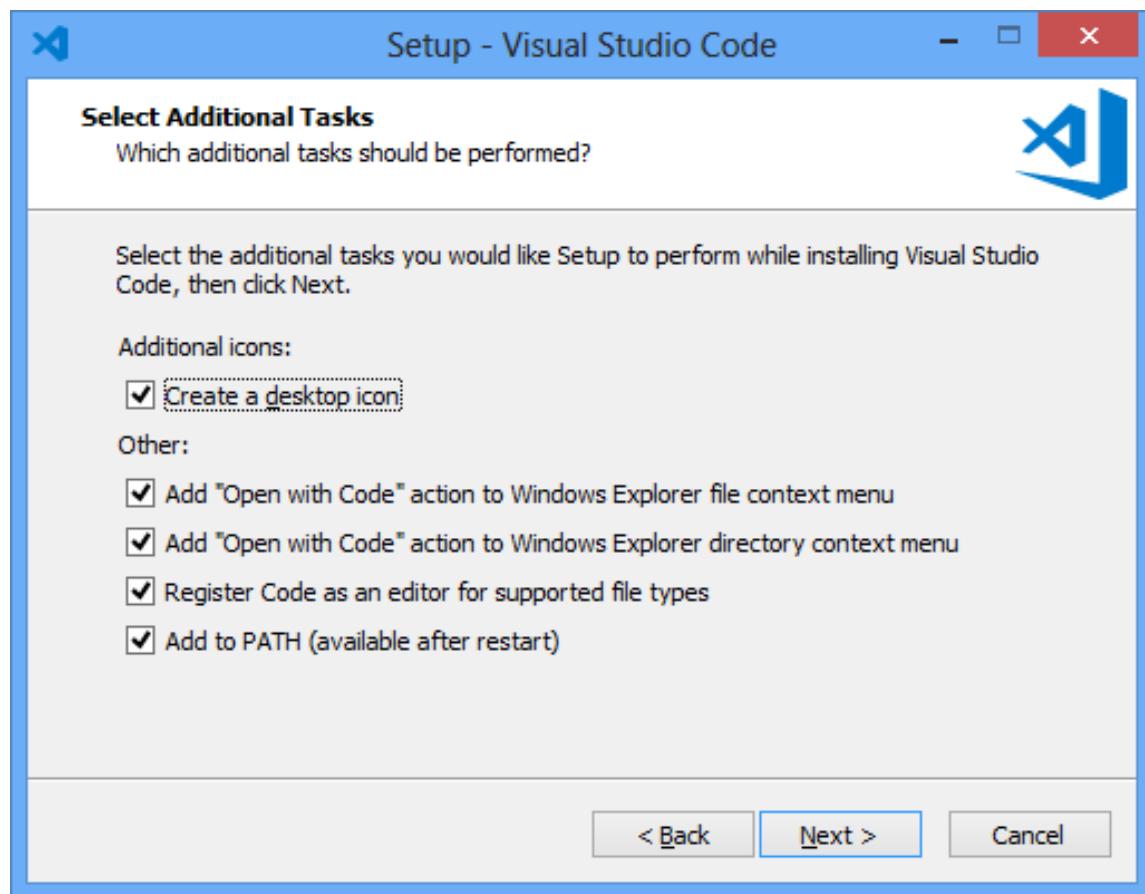
Gambar 1.26: Lokasi Direktori Instalasi

4. Berikutnya adalah menentukan lokasi direktori pada *start menu*, tampilannya akan terlihat seperti pada gambar 1.27 berikut :



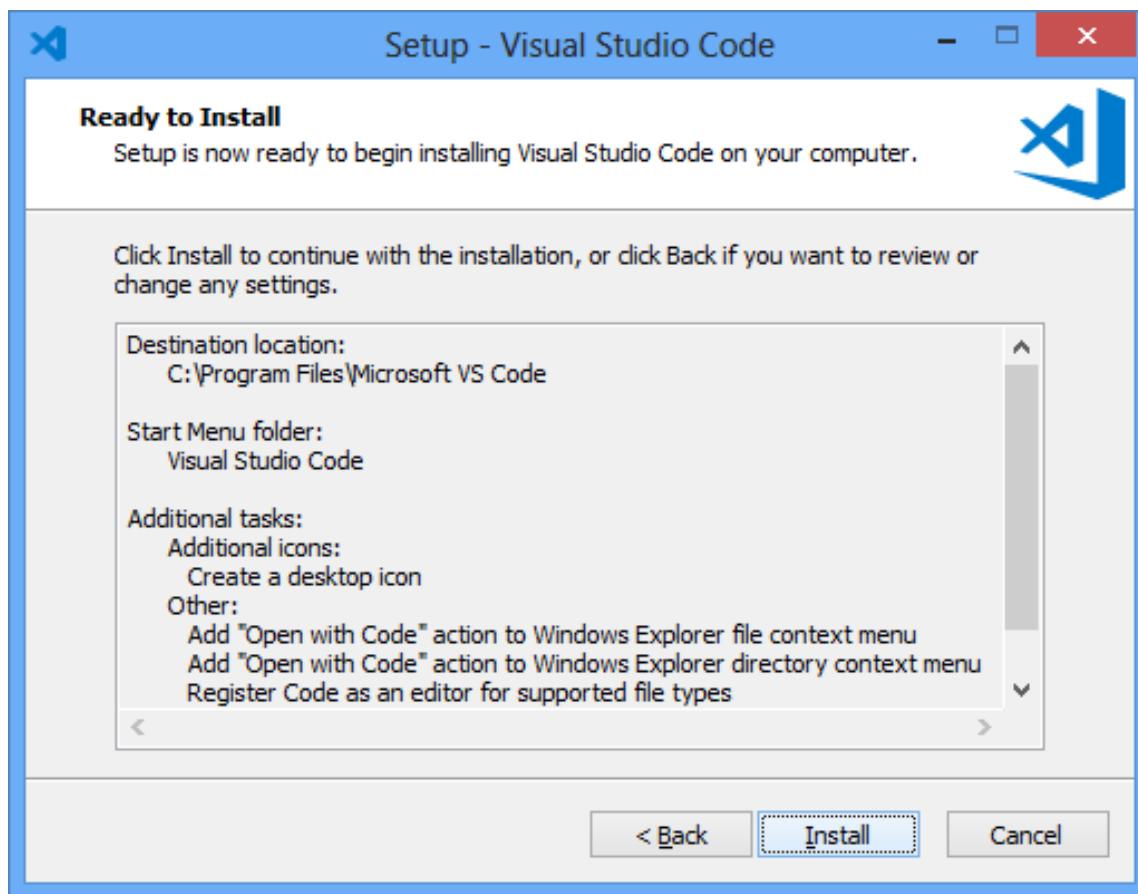
Gambar 1.27: Lokasi Pada *Start Menu*

5. Jendela berikutnya menampilkan pilihan agar pada saat instalasi dilakukan beberapa konfigurasi, tampilannya akan terlihat seperti pada gambar 1.28 berikut :



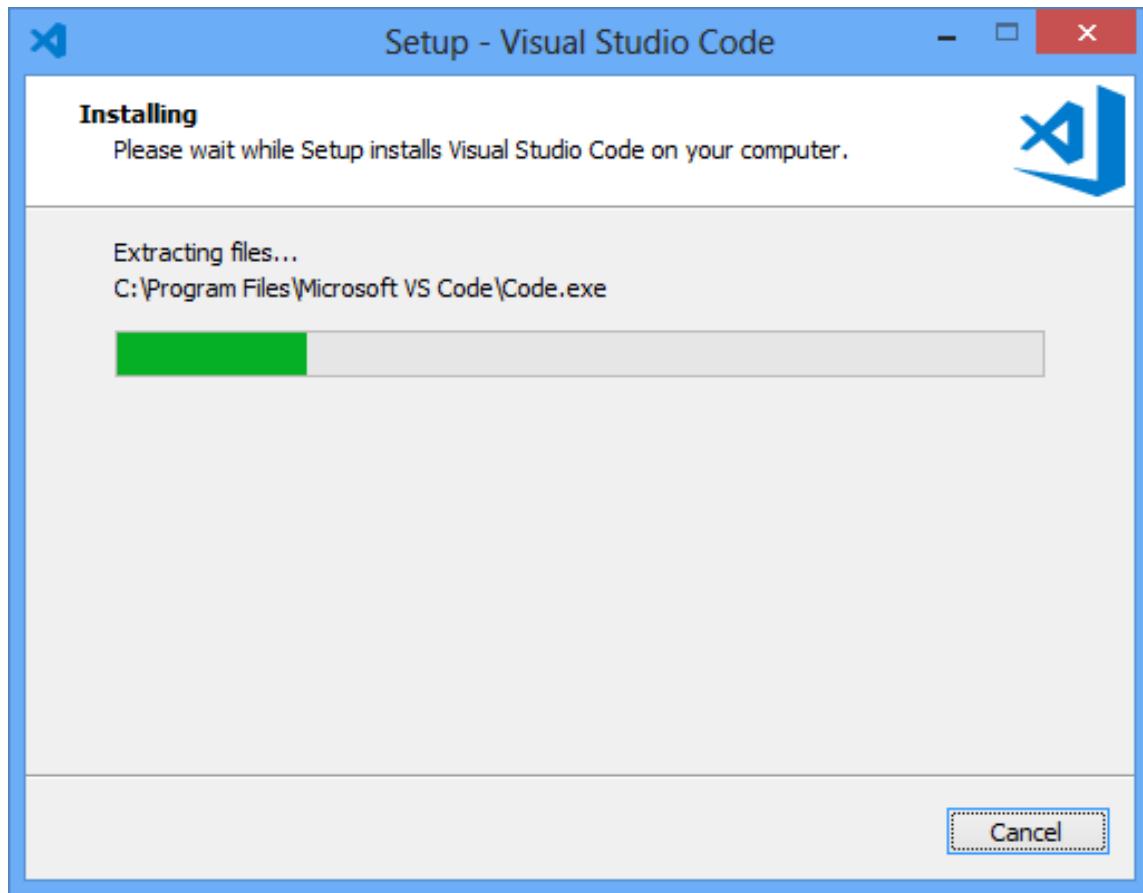
Gambar 1.28: Konfigurasi Tambahan

6. Langkah berikutnya, jendela instalasi akan menampilkan ringkasan dan proses instalasi siap dijalankan, tampilannya akan terlihat seperti gambar 1.29 berikut :



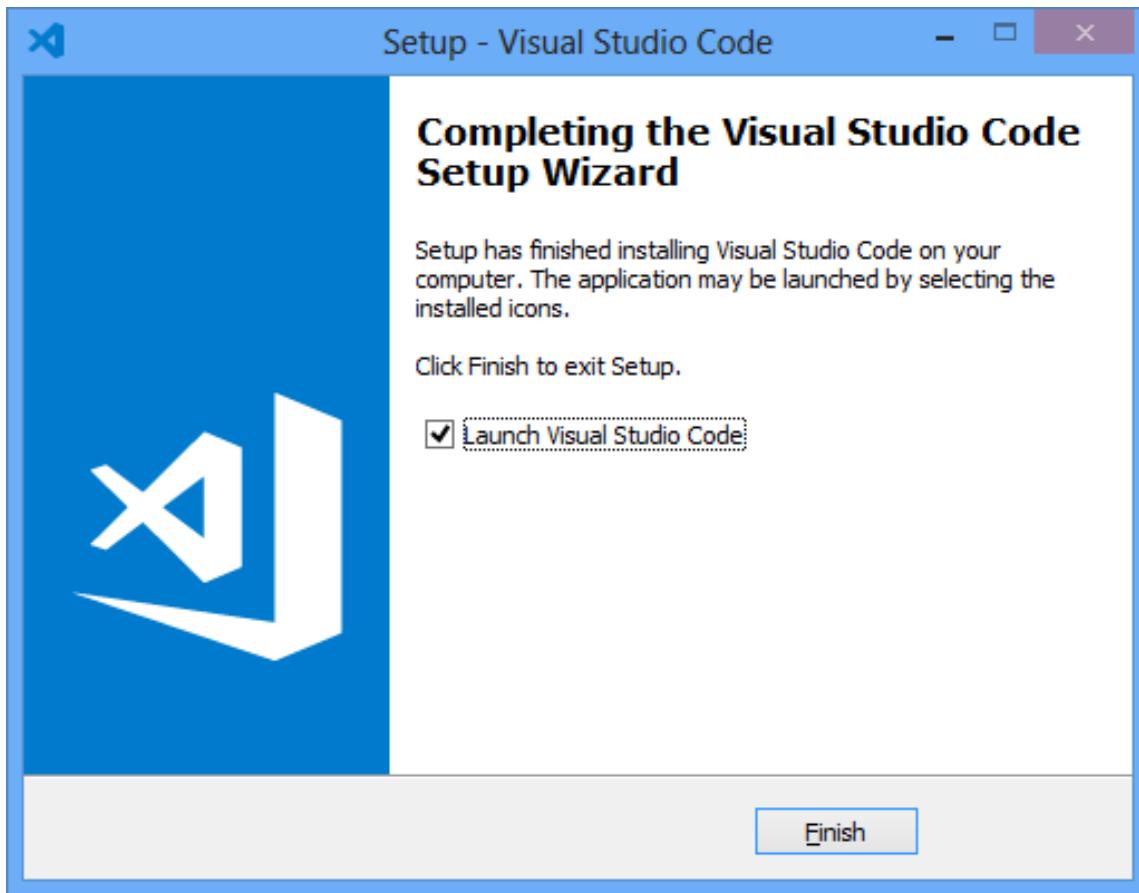
Gambar 1.29: Proses Instalasi Telah Siap

7. Jendela berikutnya akan menampilkan status proses instalasi, tampilannya akan terlihat seperti pada gambar 1.30 berikut :



Gambar 1.30: Proses Instalasi Berjalan

8. Terakhir adalah informasi bahwa proses instalasi telah selesai dan berhasil, tampilannya akan terlihat seperti pada gambar 1.31 berikut :

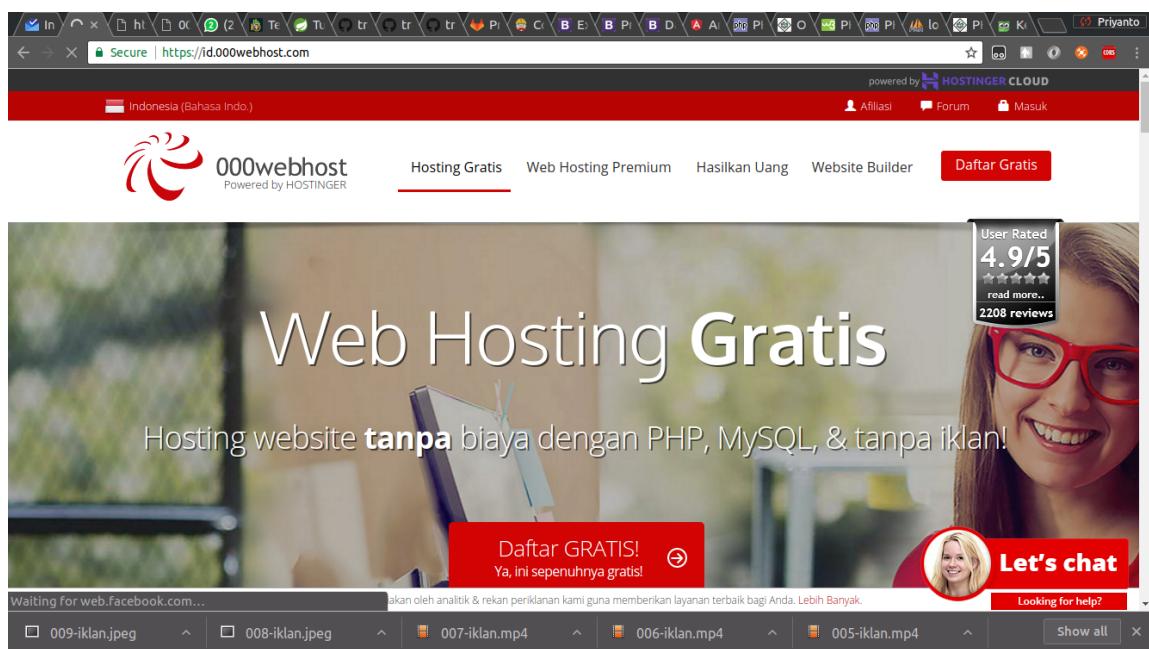


Gambar 1.31: Proses Instalasi Selesai

1.3.4 Registrasi 000webhost.com

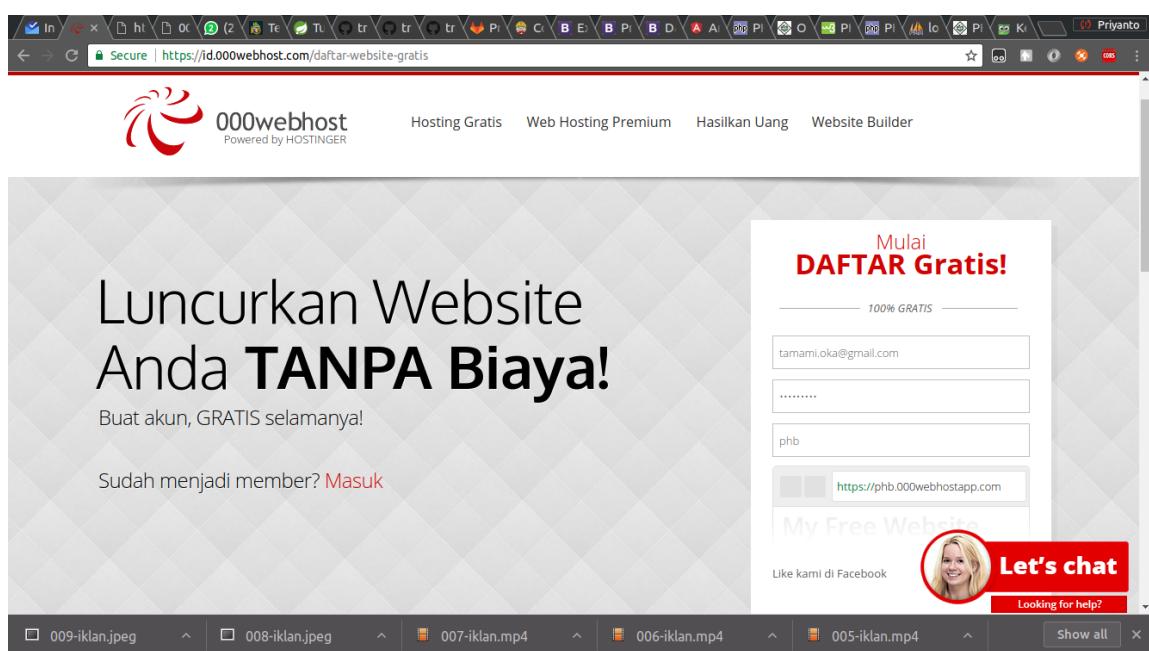
Registrasi pada website www.000webhost.com cukup mudah, langkahnya adalah sebagai berikut :

1. Mengunjungi website www.000webhost.com, apabila ingin pilihan bahasa Indonesia, kita dapat mengubahnya di bagian kiri atas, atau mengunjungi alamat id.000webhost.com, tampilan awal dari website ini seperti terlihat pada gambar 1.32 berikut :



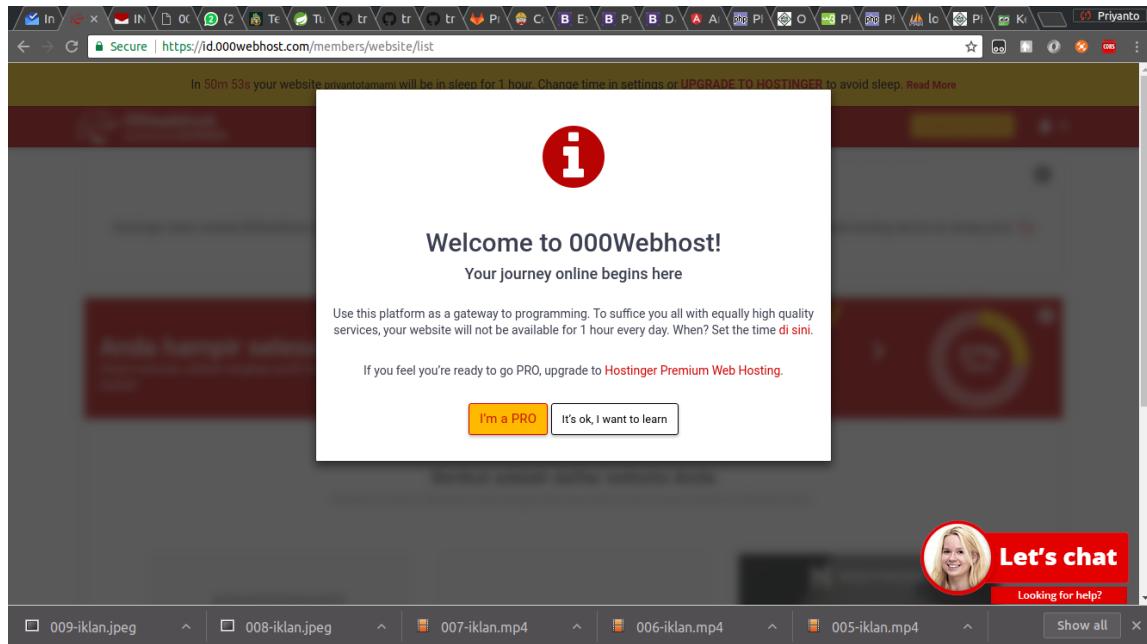
Gambar 1.32: Tampilan Awal Website id.000webhost.com

- Setelah menekan tombol "Daftar Gratis", maka kemudian akan disajikan halaman seperti pada gambar 1.33, kita perlu mengisikan alamat email, *password* untuk masuk ke halaman manajemen *website* yang kita bangun, serta alamat dari *website* yang kita inginkan. Karena sifatnya gratis, *url* yang disediakan pun mengikuti aturan dari penyedia *hosting*.



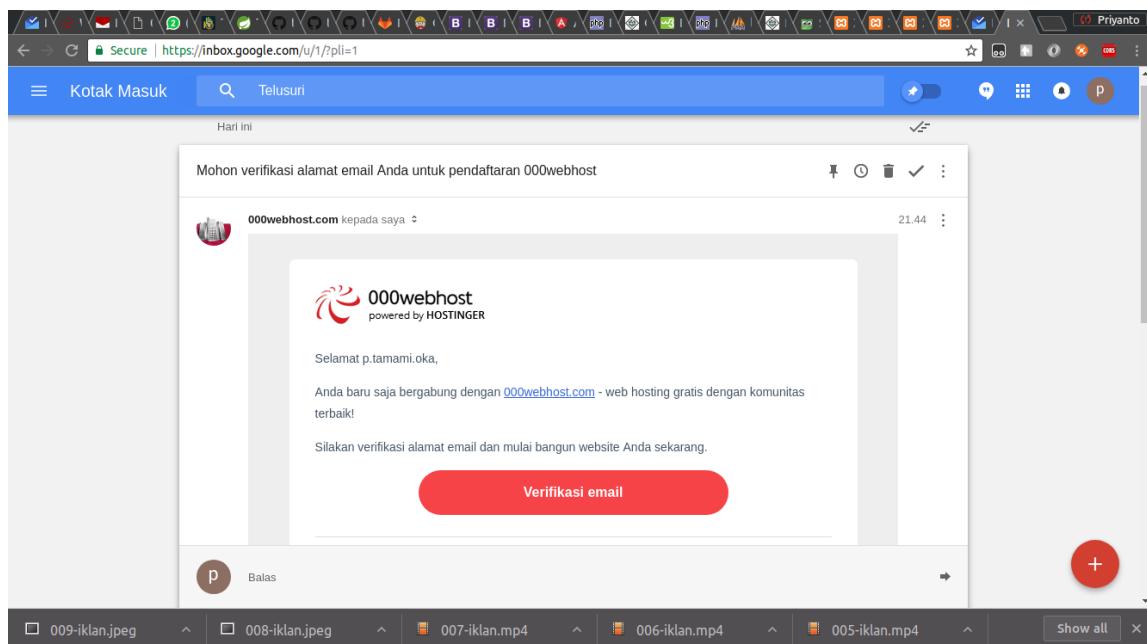
Gambar 1.33: Halaman Registrasi

3. Setelah menekan tombol pendaftaran, maka akan ditampilkan halaman selamat datang seperti pada gambar 1.34.



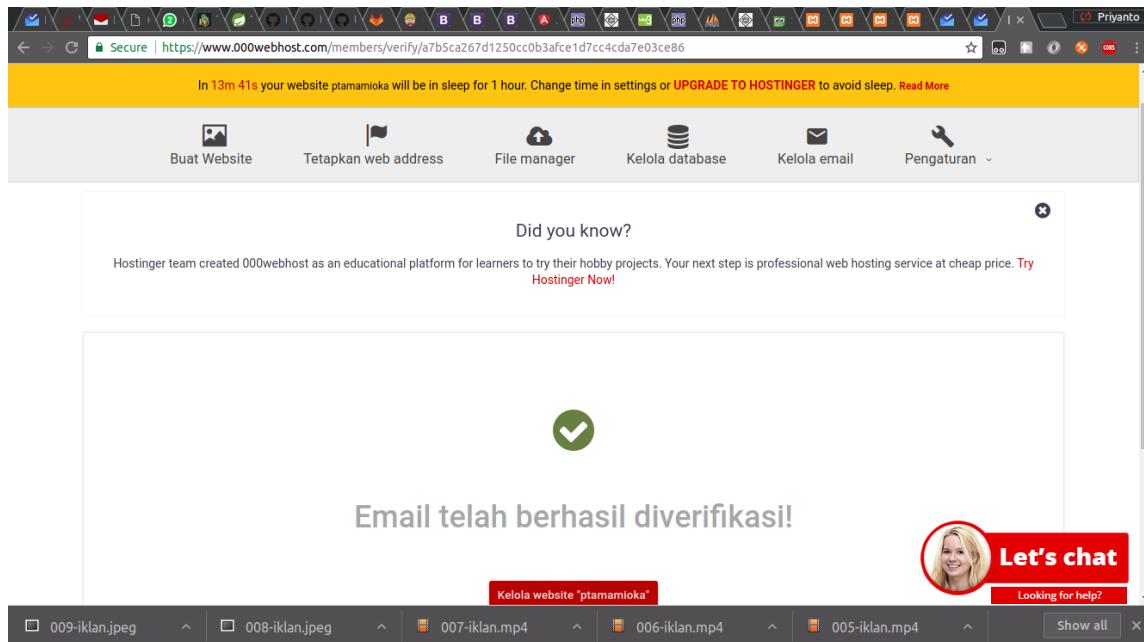
Gambar 1.34: Halaman Selamat Datang

4. Langkah berikutnya adalah melakukan verifikasi surel (*e-mail*) seperti pada gambar 1.35.



Gambar 1.35: Verifikasi Surel

5. Apabila verifikasi surel berhasil, maka akan tampil halaman seperti pada gambar 1.36.



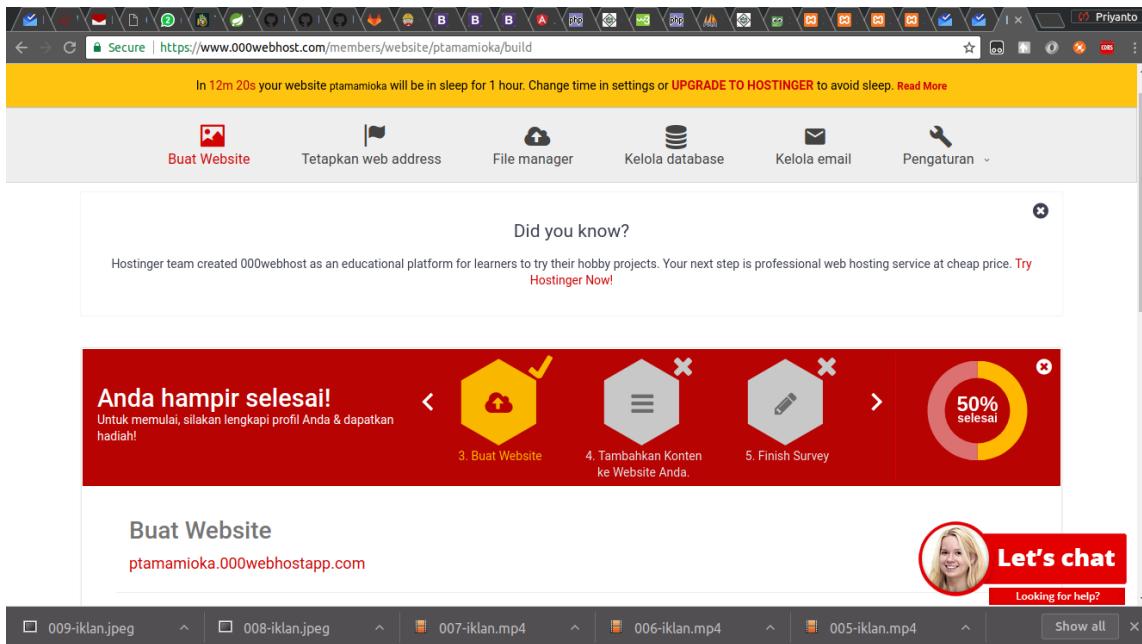
Gambar 1.36: Hasil verifikasi surel

Sampai sini, tahapan pendaftaran anggota telah berhasil kita lakukan.

1.3.5 Aplikasi Selamat Datang

Kita akan mencoba melakukan *publish* terhadap sebuah *file html* untuk membuktikan bahwa halaman *website* yang kita unggah ke id.000webhost.com dapat berhasil. Berikut adalah langkah-langkahnya :

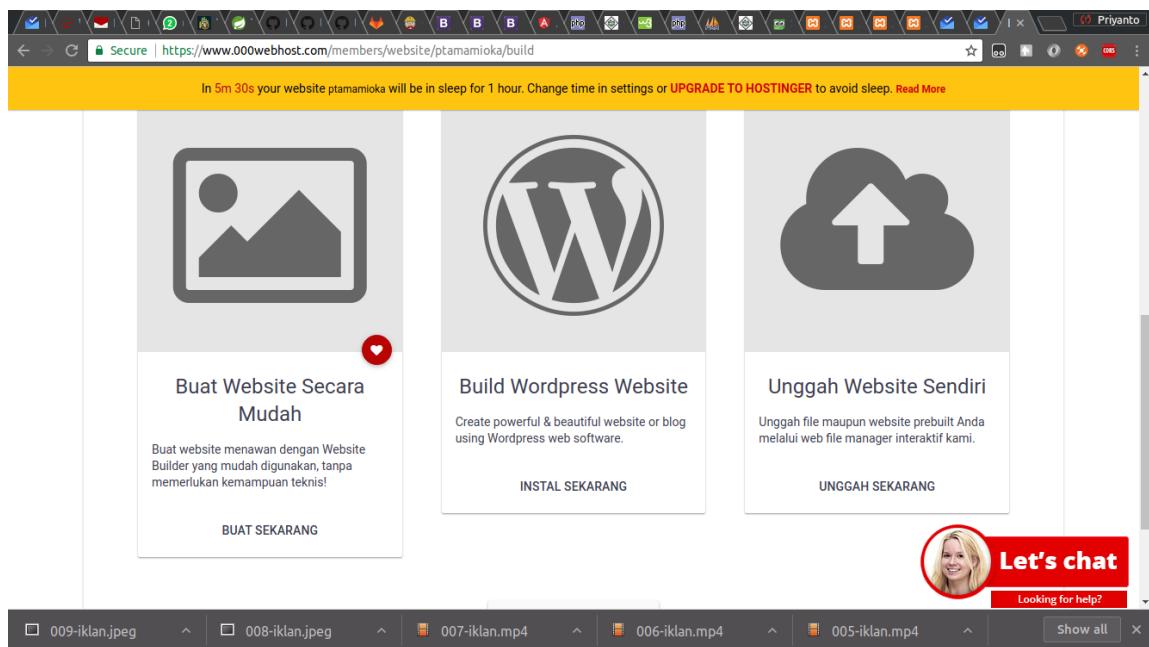
1. Langkah pertama adalah membuat sebuah halaman *website* dengan menekan tombol kiri atas sehingga tampil halaman seperti pada gambar 1.37.

Gambar 1.37: Membuat *website*

2. Selanjutnya kita membuat *file* dengan nama `index.html` terlebih dahulu, pembuatan *file* ini dapat kita lakukan dengan notepad, vim, atau Visual Studio Code yang telah kita *install*. Isi dari *file* ini adalah sebagai berikut :

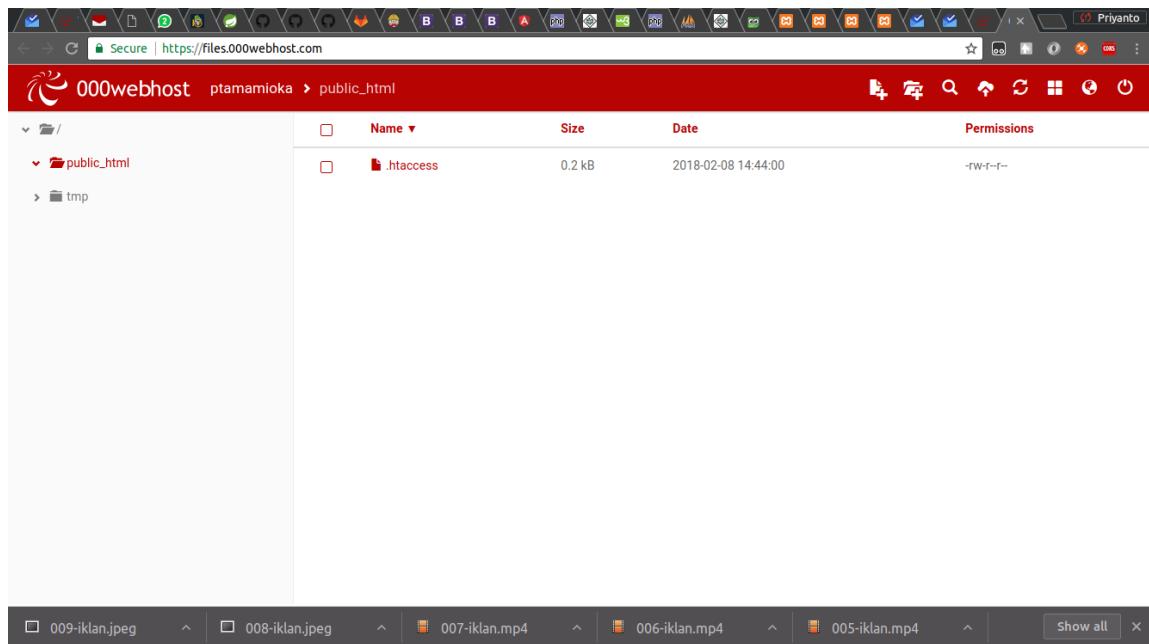
`1 Hai , selamat datang .`

3. Kemudian kita *scroll* ke bawah halaman `id.000webhost.com`, maka akan muncul tampilan seperti pada gambar 1.38. Karena kita telah membuat sebuah *file* `index.html`, maka kita memilih "Unggah Website Sendiri".



Gambar 1.38: Pilihan cara membuat *website*

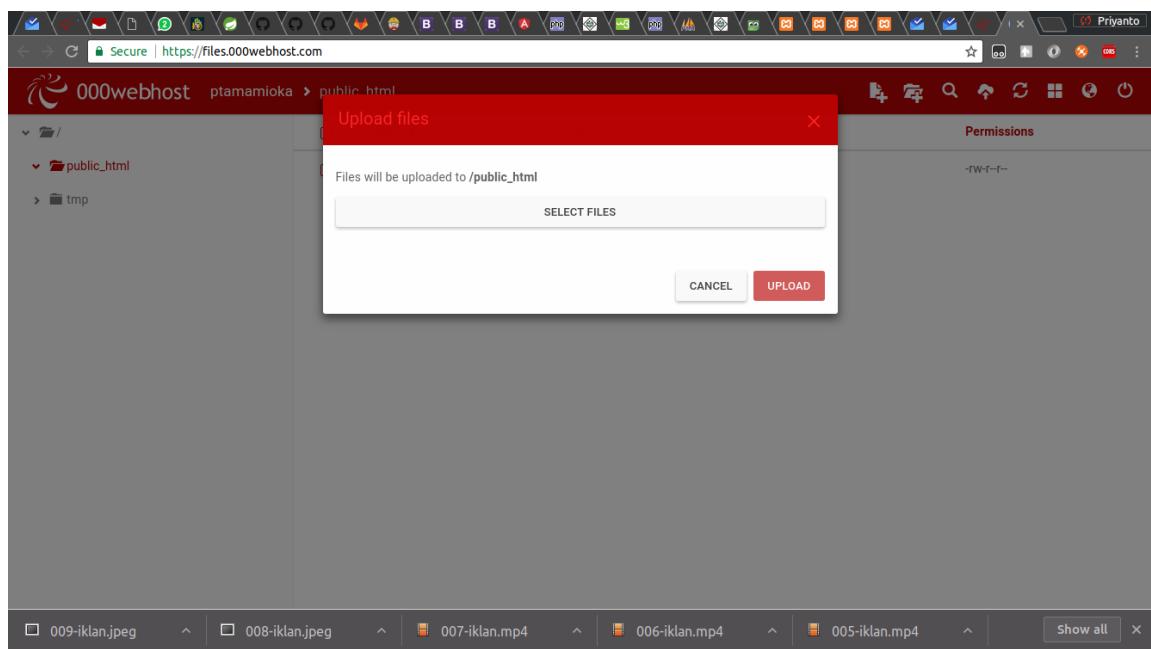
4. Selanjutnya kita akan ditunjukkan halaman *file manager* dimana nantinya di dalam *folder public_html* ini *project* kita ditempatkan. Tampilannya seperti pada gambar 1.39.



Gambar 1.39: Halaman *File Manager*

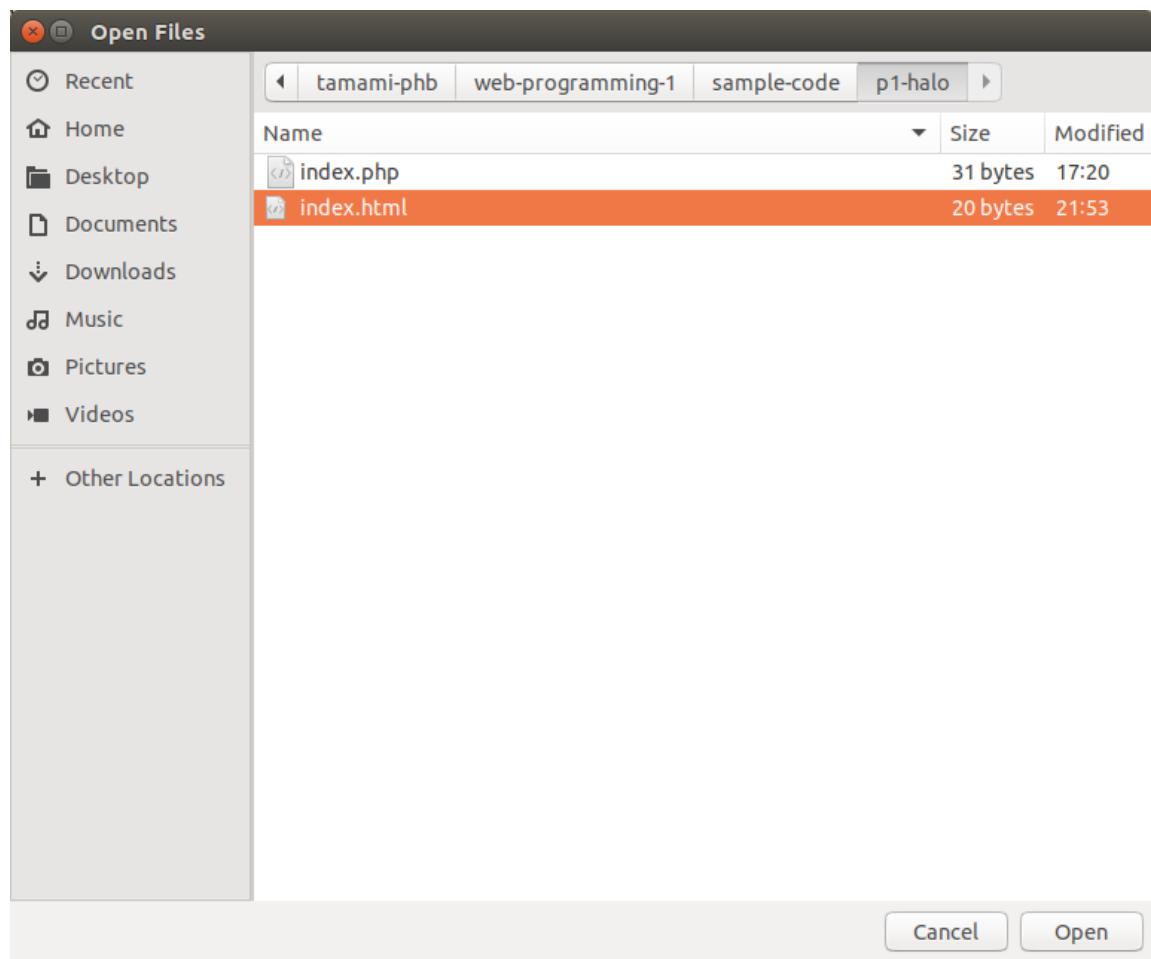
5. Pilihlah ikon dengan gambar awan dan tanda panah atas pada bagian atas kanan jendela *file manager* untuk mengunggah *file index.html* yang telah

kita buat sebelumnya. Nantinya akan muncul tampilan seperti pada gambar 1.40.



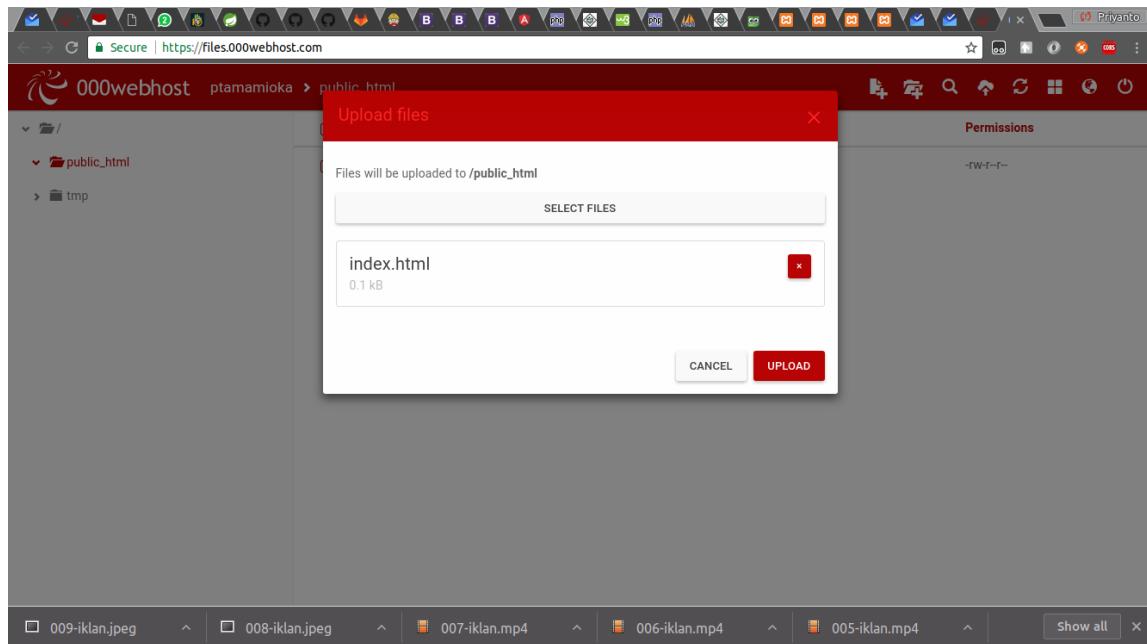
Gambar 1.40: Jendela Unggah *File*

6. Tekanlah tombol **SELECT FILES** yang berada di tengah sehingga muncul jendela pemilihan *file* seperti pada gambar 1.41, lalu memilih *file* dengan nama **index.html** yang akan kita unggah ke *server*.

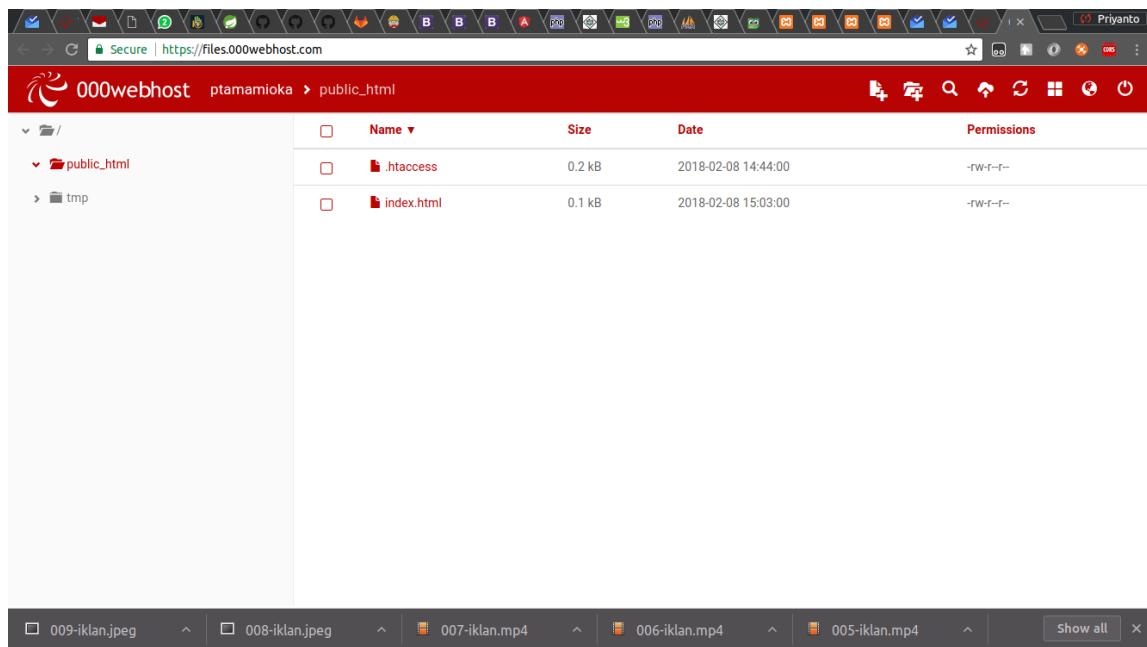


Gambar 1.41: Jendela Pemilihan *File*

7. Setelah *file* dipilih, maka jendela *Upload Files* akan menampilkan *file* yang terpilih untuk selanjutnya siap diunggah seperti pada gambar 1.42.

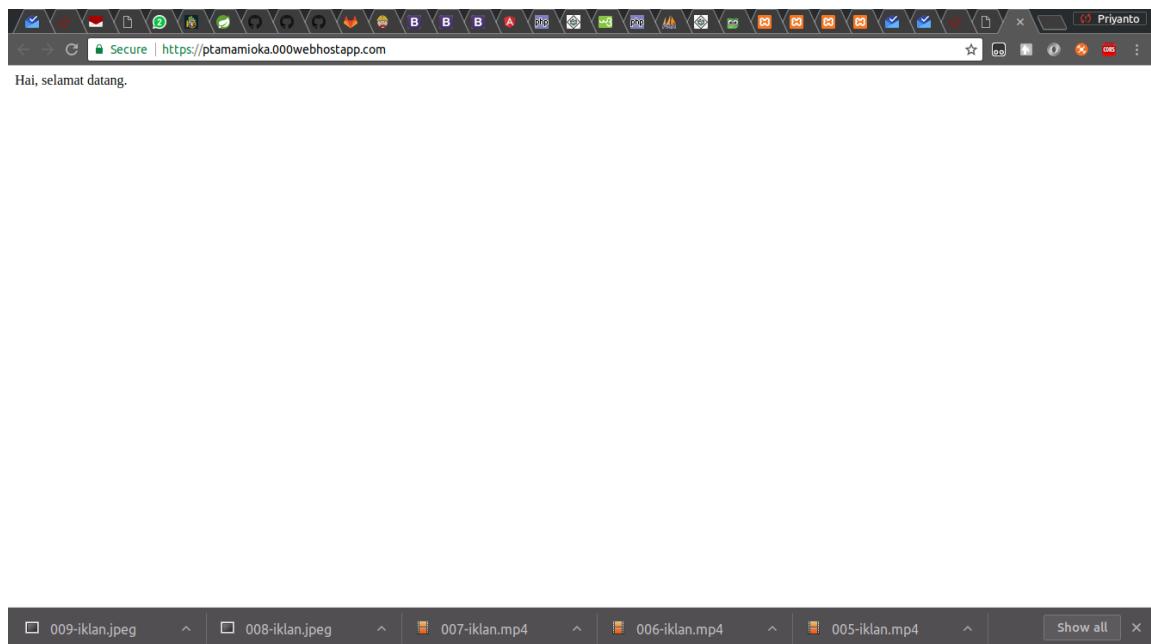
Gambar 1.42: Jendela *File Terpilih*

8. Setelah selesai terunggah, maka akan tampil *file* yang telah diunggah seperti pada gambar 1.43.

Gambar 1.43: Jendela *File Manager* Setelah File Terunggah

9. Lakukan akses ke halaman yang telah disediakan oleh www.000webhost.com yang hasilnya seperti terlihat pada gambar 1.44. Karena ini layanan gratis, jadi jeda antara waktu *file* telah terunggah dengan hasil *website* memakan

waktu yang bervariasi. Apabila protokol `https` tidak berhasil menampilkan halaman yang telah kita buat, cobalah menggunakan protokol `http`.

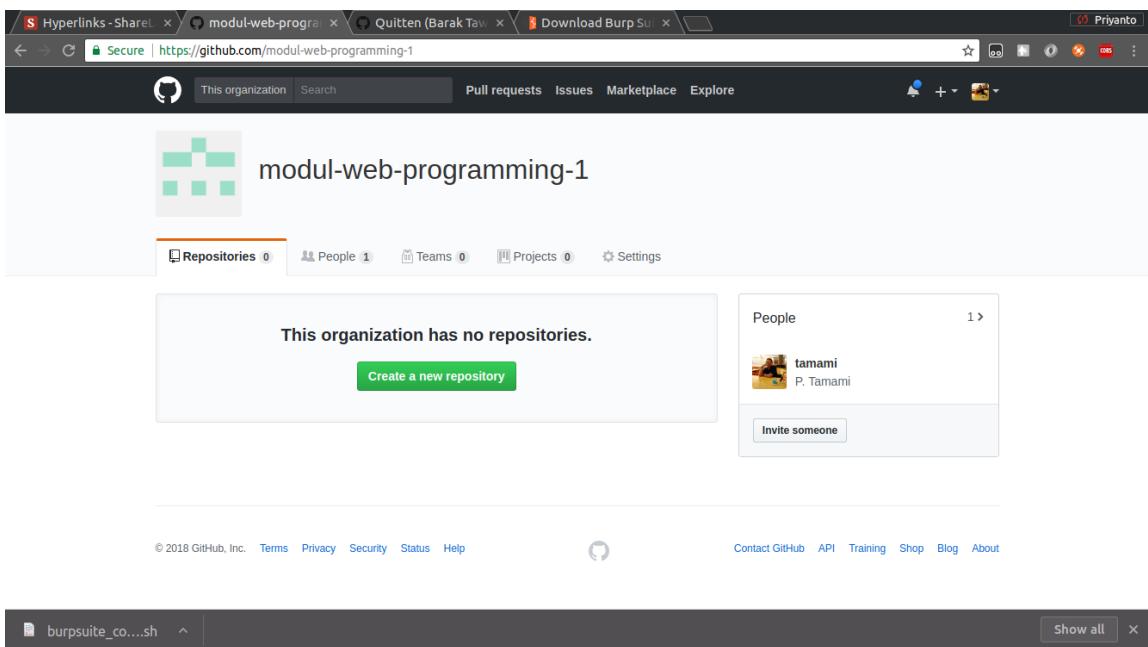


Gambar 1.44: Hasil *Website*

1.3.6 Unggah ke Github

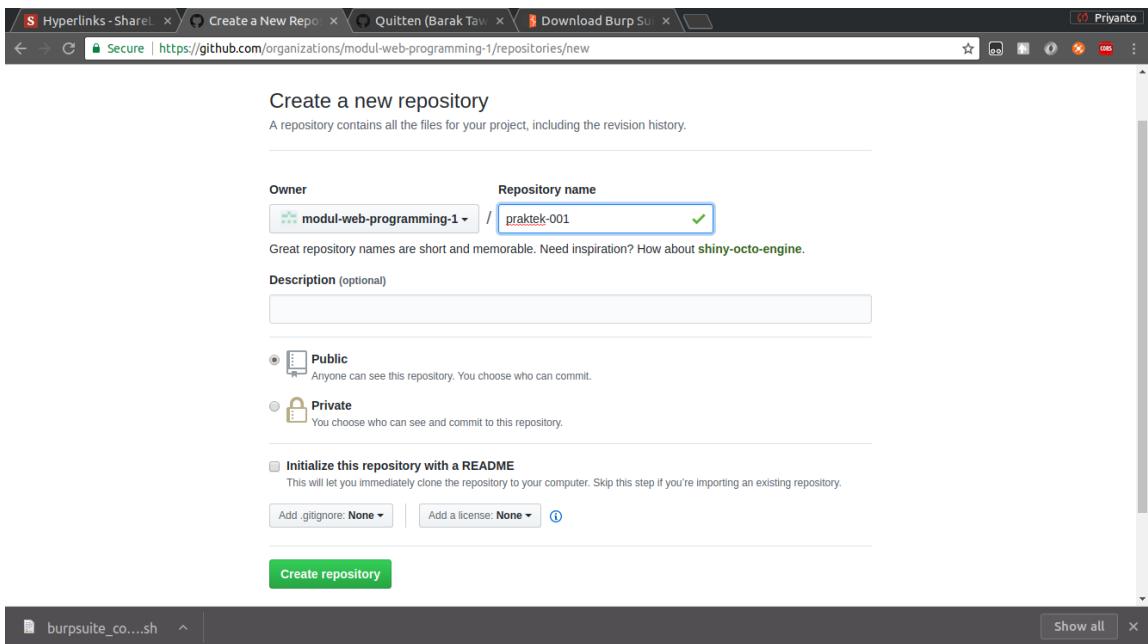
Setelah kode awal berhasil kita unggah ke `id.000webhost.com`, kita akan unggah pula kode yang kita buat ke `Github` untuk keperluan *versioning* kode. Langkahnya cukup mudah, yaitu :

1. Membuat *repository* di Github dengan menekan tombol hijau bertuliskan "Create a new repository" seperti pada gambar 1.45.



Gambar 1.45: Halaman Awal Github

2. Kemudian mengisikan nama repositorinya pada kolom yang tersedia, lalu tekan tombol **Enter** atau klik tombol ”Create Repository” seperti pada gambar 1.46 yang nantinya.

Gambar 1.46: Membuat *Repository*

3. Membuka terminal atau command prompt atau console untuk melakukan *init git* dengan lokasi direktoriya adalah tempat file *index.html* berada dengan

kode berikut :

```
1 > git init
```

4. Melakukan *staging file* dengan perintah berikut :

```
1 > git add .
```

5. Melakukan *commit* terhadap *staging file* dengan perintah berikut :

```
1 > git commit -m "init commit"
```

Opsi perintah `-m "init commit"` sebetulnya wajib, jadi setiap kita melakukan *commit*, kita diminta untuk memberikan keterangan / komentar di tiap *commit* untuk memudahkan kita mencari tahap perubahan tertentu.

6. Mendaftarkan alamat penyedia layanan repositori `git` dengan perintah berikut :

```
1 > git remote add github https://github.com/modul-web-programming  
-1/praktek-001.git
```

`github` di atas adalah nama alias, jadi boleh diganti apapun, sedangkan `url` setelahnya adalah alamat yang diberikan oleh Github setelah kita membuat sebuah *repository*.

7. Melakukan unggah kode ke Github dengan perintah berikut :

```
1 > git push -u github master
```

`github` adalah alias yang sebelumnya kita buat, sedangkan `master` adalah *branch* utama dari `git`.

1.4 Kesimpulan

Bahwa membangun sebuah aplikasi *web* menggunakan PHP yang bersifat dinamis diperlukan beberapa perangkat / aplikasi pendukung.

Dengan fasilitas gratis pun kita masih dapat melakukan *publish project* aplikasi web yang dibangun dengan PHP kita ke internet sehingga dapat diakses oleh semua orang.

1.5 Tugas

Membuat halaman selamat datang dalam bentuk HTML dan di-*publish* ke internet.

Bab 2

Operator, Variabel, dan Tipe Data

2.1 Tujuan

Mahasiswa mampu mengimplementasikan alur sederhana untuk menggunakan operator, variabel, dan tipe data pada PHP.

2.2 Pengantar

Variabel sebetulnya bisa diibaratkan sebagai wadah yang akan diisikan oleh data. Sedangkan *tipe data* bisa dianalogikan seperti jenis data yang berada pada suatu variabel. *Operator* sendiri adalah perintah untuk melakukan operasi tertentu pada satu atau lebih variabel / data.

Untuk variabel sebenarnya akan memiliki *scope* seperti berikut :

- **local**, variabel akan bersifat lokal, tidak dapat digunakan di luar deklarasinya.
- **global**, variabel akan bersifat global, dan dapat digunakan di luar deklarasinya.
- **static**, variabel jenis ini tidak akan dihapus dari memori ketika sudah dieksekusi.

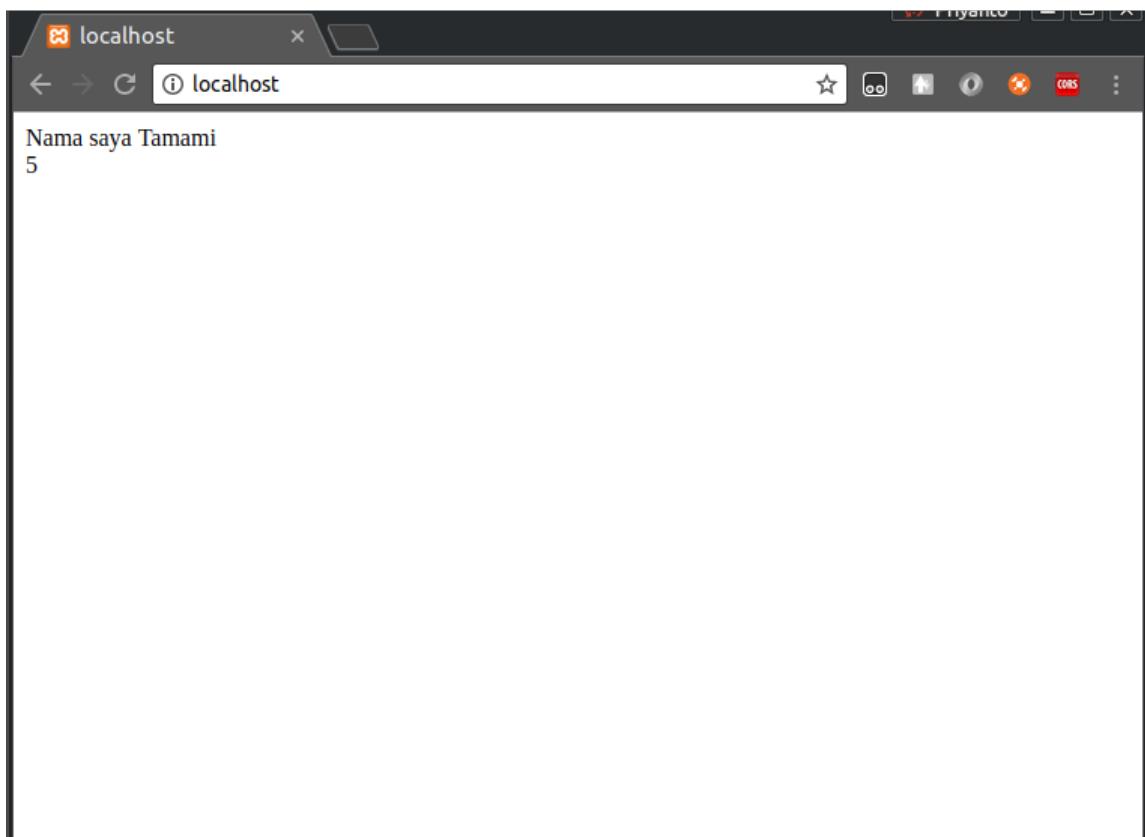
2.3 Praktek

2.3.1 Variabel

Untuk menggunakan variabel, kita cukup memberikan tanda dollar (\$) di awal nama variabel. Contoh kodennya adalah sebagai berikut :

```
1 <?php
2 $teks = "Nama saya Tamami";
3 $angka = 5;
4
5 echo $teks;
6 echo "<br>";
7 echo $angka;
8 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 2.1 berikut :



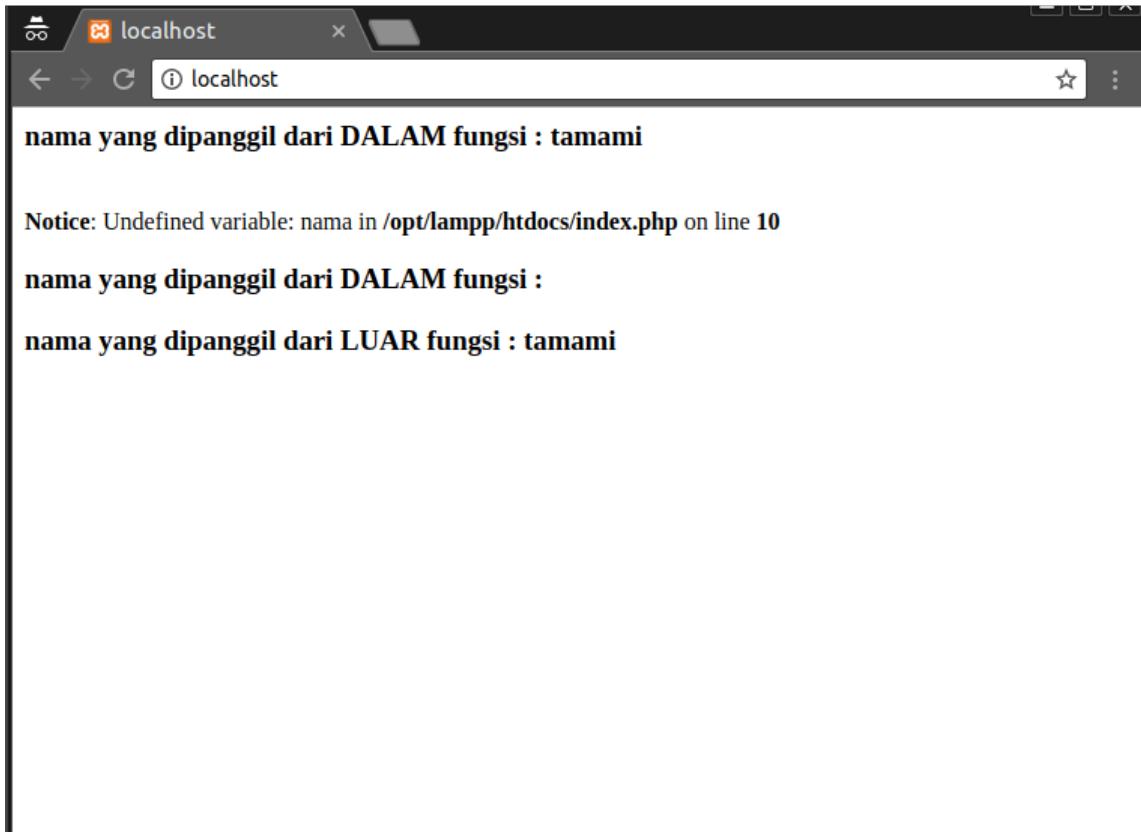
Gambar 2.1: Hasil Keluaran Variabel

Variabel *Global*

Contoh kode untuk menjelaskan bagaimana sifat variabel *global* adalah sebagai berikut :

```
1 <?php
2 $nama = "tamami";
3
4 function cetakNama1() {
5     global $nama;
6     echo "<h3>nama yang dipanggil dari DALAM fungsi : $nama</h3>";
7 }
8
9 function cetakNama2() {
10    echo "<h3>nama yang dipanggil dari DALAM fungsi : $nama</h3>";
11 }
12
13 cetakNama1();
14 cetakNama2();
15
16 echo "<h3>nama yang dipanggil dari LUAR fungsi : $nama</h3>";
17 ?>
```

Hasil dari kode di atas adalah seperti tampilan pada gambar ?? berikut :



Gambar 2.2: Hasil Keluaran Variabel Global

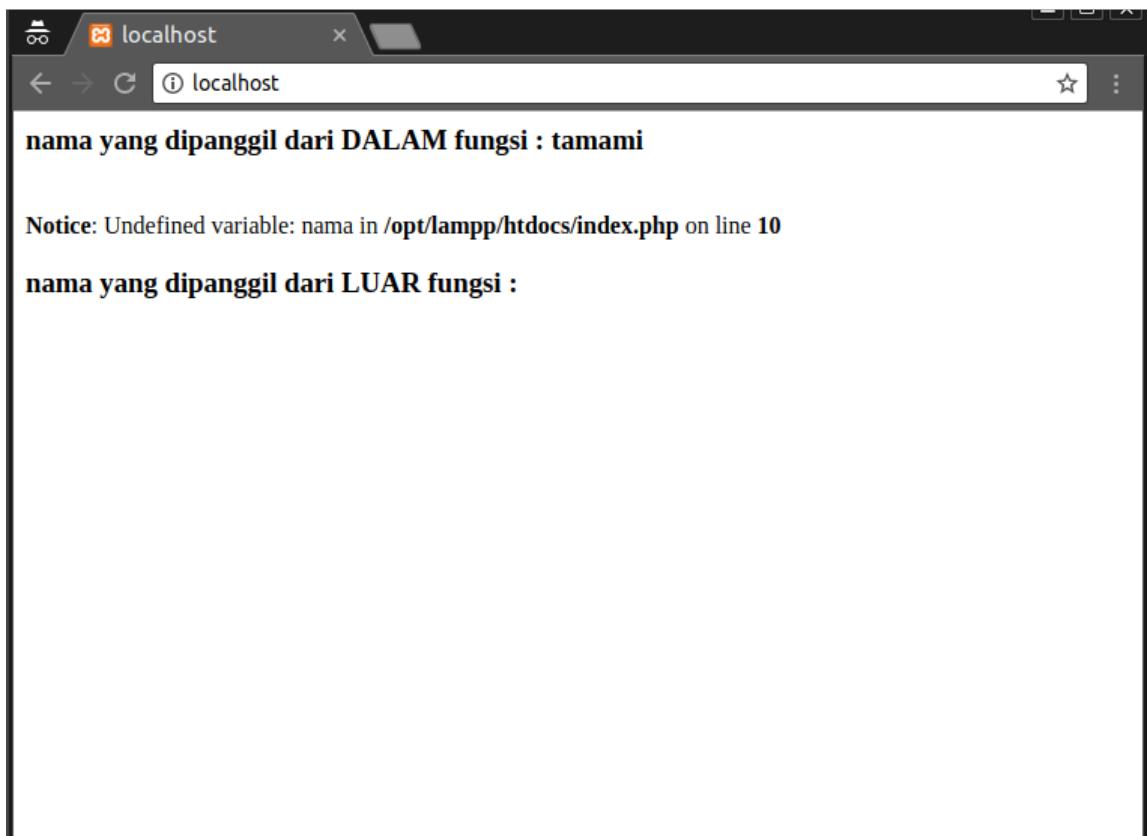
Artinya, variabel *global* dapat digunakan di dalam fungsi maupun di luar fungsi, apabila penggunaannya di dalam sebuah fungsi, maka perlu dideklarasikan terlebih dahulu di dalam fungsi tersebut dengan kata kunci **global**.

Variabel *Local*

Untuk penggunaan variabel *local*, contoh kodennya adalah sebagai berikut :

```
1 <?php
2
3 function cetakNama() {
4     $nama = "tamami";
5     echo "<h3>nama yang dipanggil dari DALAM fungsi : $nama</h3>";
6 }
7
8 cetakNama();
9
10 echo "<h3>nama yang dipanggil dari LUAR fungsi : $nama</h3>";
11 ?>
```

Hasil keluaran dari kode program tersebut adalah seperti pada gambar 2.3 berikut :



Gambar 2.3: Hasil Keluaran Variabel *Local*

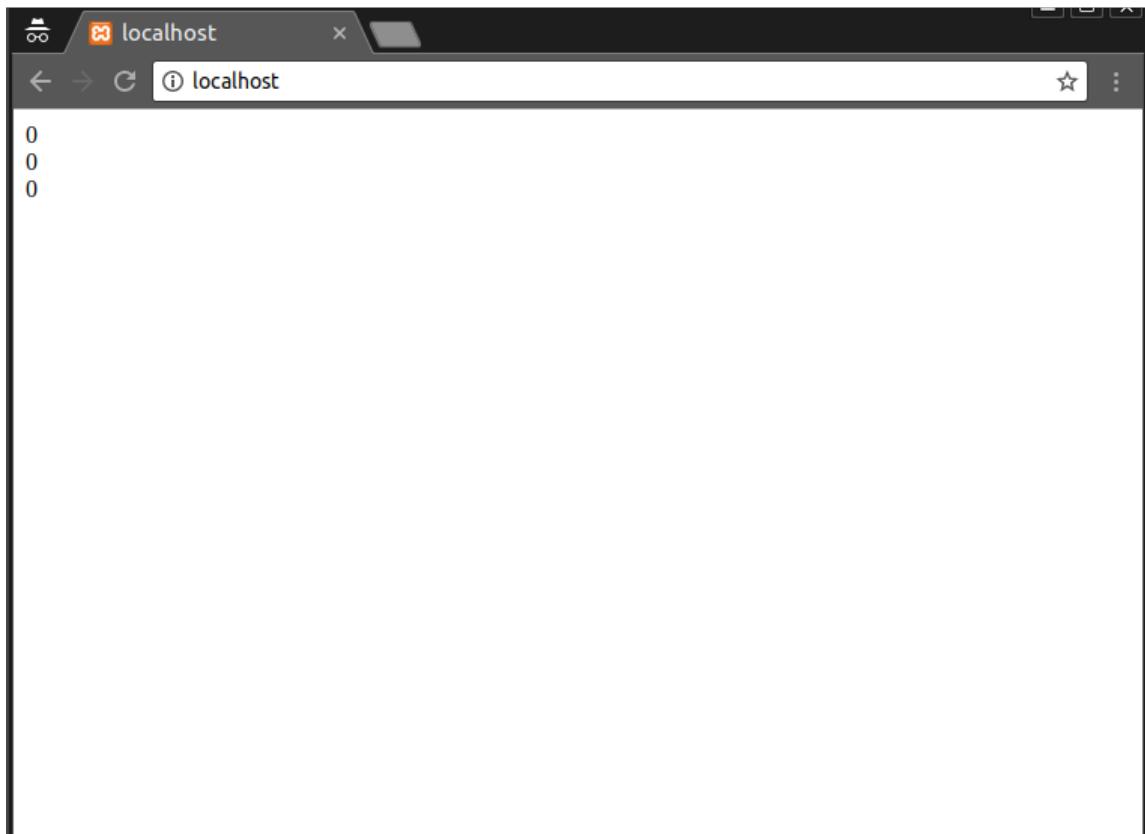
Artinya, variabel `$nama` yang dideklarasikan di dalam sebuah fungsi hanya dapat digunakan di dalam fungsi yang bersangkutan.

Variabel *Static*

Karena sifat dari variabel di bahasa pemrograman PHP ini akan dihapus dari memori ketika telah dieksekusi, maka bila kita ingin menggunakan variabel ini lebih lama, maka kita harus menjadikannya `static`, berikut adalah contoh kode dimana jenis variabel tidak `static` :

```
1 <?php
2 function hitung() {
3     $nilai = 0;
4     echo $nilai;
5     $nilai += 5;
6 }
7
8 hitung();
9 echo "<br>";
10 hitung();
11 echo "<br>";
12 hitung();
13 ?>
```

Hasil dari kode di atas adalah seperti gambar 2.4 berikut :



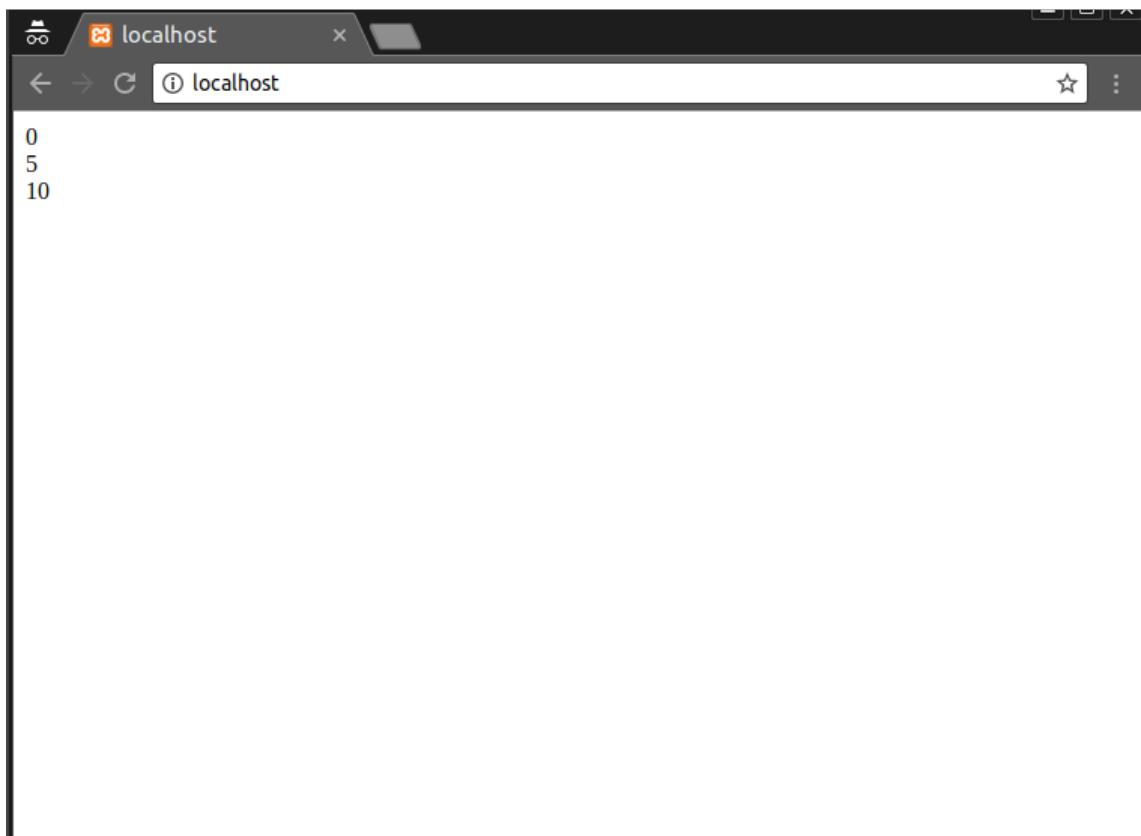
Gambar 2.4: Hasil Keluaran Variabel Tanpa static

Tampak bahwa variabel \$nilai akan selalu bernilai 0 (nol) karena setiap kali eksekusi \$nilai += 5 maka begitu keluar dari fungsi hitung(), isi variabel akan kembali menjadi 0 (nol).

Apabila kita jadikan variabel \$nilai, maka kodennya menjadi seperti ini :

```
1 <?php
2 function hitung() {
3     static $nilai = 0;
4     echo $nilai;
5     $nilai += 5;
6 }
7
8 hitung();
9 echo "<br>";
10 hitung();
11 echo "<br>";
12 hitung();
13 ?>
```

Hasil keluaran dari kode tersebut adalah seperti pada gambar 2.5 berikut :



Gambar 2.5: Hasil Keluaran Variabel Dengan `static`

Hasil dari keluaran pada gambar 2.5 tampak bahwa variabel `$nilai` akan melakukan eksekusi dan menyimpan hasil tanpa menghapusnya dari memori.

2.3.2 Tipe Data

Variabel pada dasarnya menyimpan sebuah data dengan tipe data tertentu, masing-masing tipe data dapat melakukan hal yang berbeda-beda. Tipe data yang didukung oleh PHP ini adalah sebagai berikut :

- String, tipe data ini untuk menyimpan teks
- Integer, tipe data ini untuk menyimpan angka bulat, tanpa desimal
- Float / Double, tipe data ini untuk menyimpan angka yang memiliki bilangan desimal
- Boolean, tipe data ini hanya menampung nilai `true` atau `false`
- Array, tipe data ini akan menampung baris data baik dengan tipe data yang sama, maupun tipe data yang berbeda.
- Object, tipe data ini akan menampung objek yang kita buat sendiri

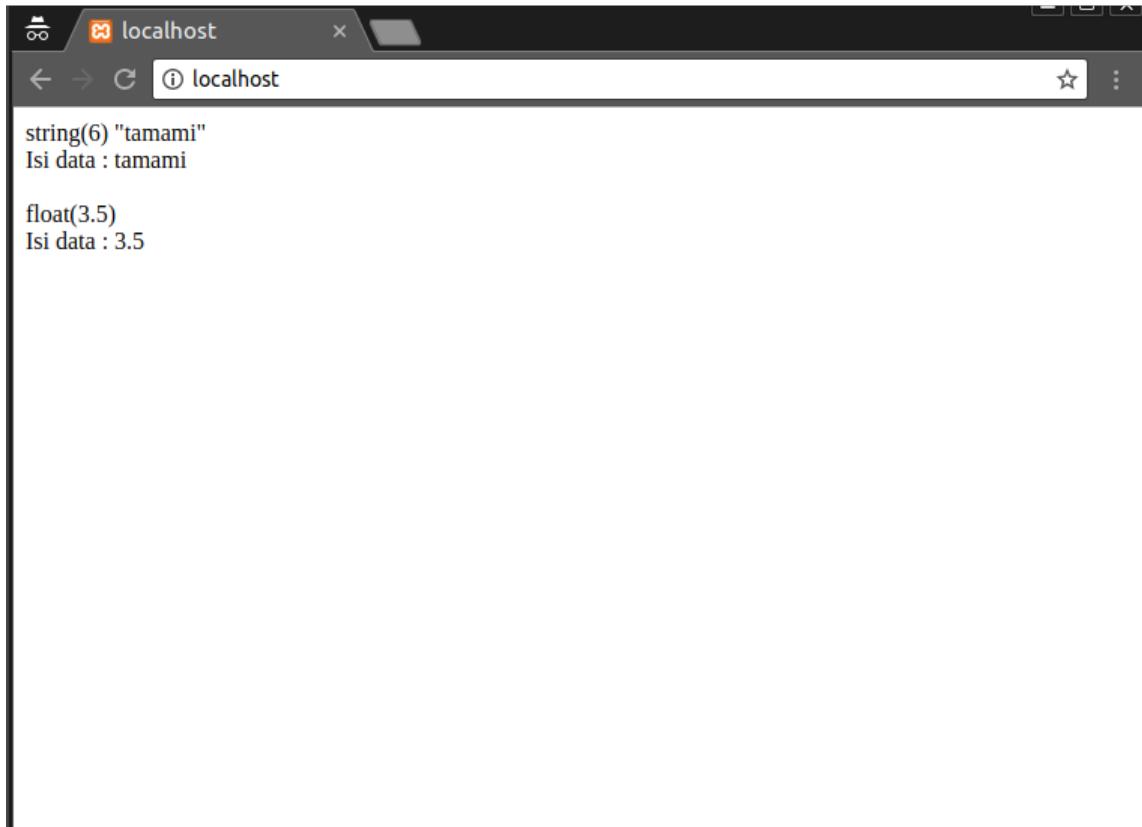
- NULL, tipe data ini artinya tidak memiliki data sama sekali.

Tipe data di PHP tidak dideklarasikan secara langsung, melainkan pada saat pengisian data ke sebuah variabel, pada saat itu juga tipe data akan ditentukan terhadap variabel tersebut.

Perhatikan kode berikut ini :

```
1 <?php
2 $data = "tamami";
3 var_dump($data);
4 echo "<br>";
5 echo "Isi data : $data";
6 echo "<br>";
7 echo "<br>";
8 $data = 3.5;
9 var_dump($data);
10 echo "<br>";
11 echo "Isi data : $data";
12 ?>
```

Kode tersebut akan menghasilkan tampilan seperti pada gambar 2.6 berikut :



Gambar 2.6: Melihat Tipe Data

Terlihat pada saat kita mengisikan variabel \$data dengan teks, maka tipe data

langsung terisi menjadi **string**, kemudian saat kita mengganti isinya menjadi 3.5 maka tipe data ikut berubah menjadi **float**.

2.3.3 Operator

Operator pada PHP akan dibagi menjadi beberapa bagian seperti berikut :

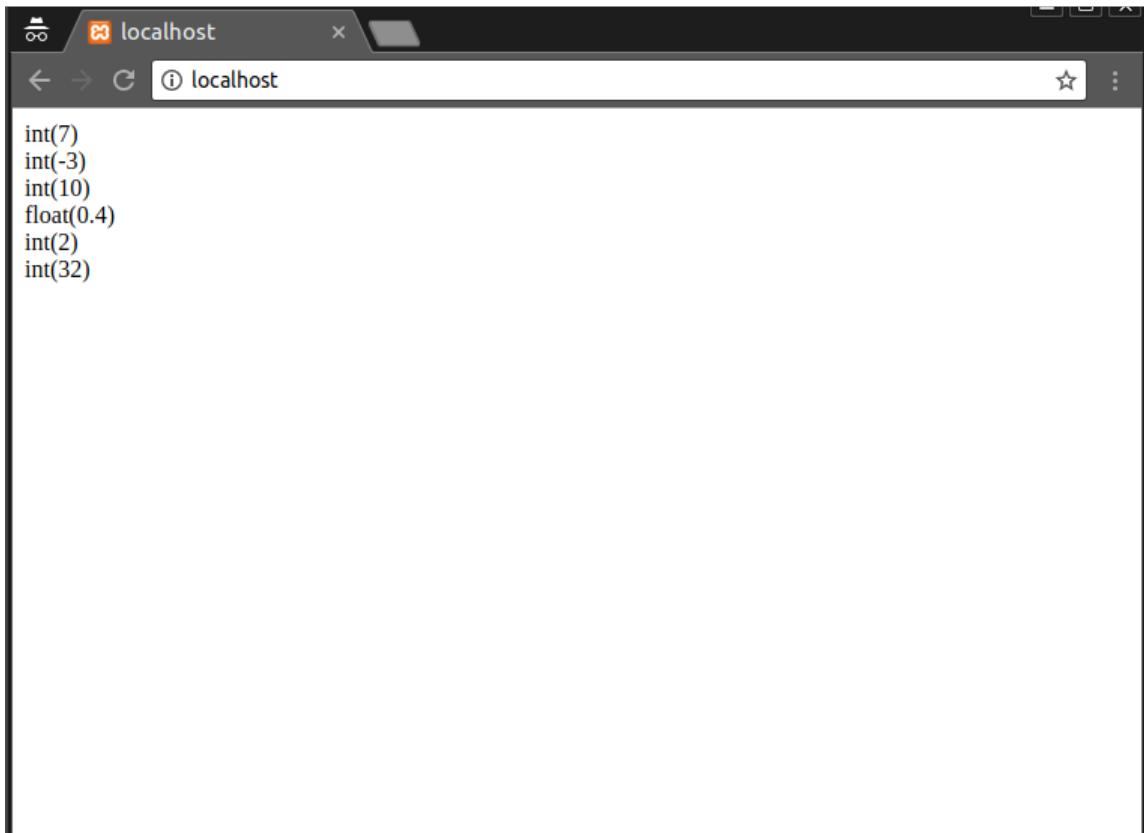
Arithmetric Operators

Operator ini berlaku untuk melakukan operasi aritmatika dasar di PHP, contoh kodennya adalah sebagai berikut :

```

1 <?php
2 $x = 2;
3 $y = 5;
4
5 $result = $x + $y; // operasi penjumlahan
6 var_dump($result);
7 echo "<br>";
8
9 $result = $x - $y; // operasi pengurangan
10 var_dump($result);
11 echo "<br>";
12
13 $result = $x * $y; // operasi perkalian
14 var_dump($result);
15 echo "<br>";
16
17 $result = $x / $y; // operasi pembagian
18 var_dump($result);
19 echo "<br>";
20
21 $result = $x % $y; // operasi sisa bagi
22 var_dump($result);
23 echo "<br>";
24
25 $result = $x ** $y; // operasi pangkat
26 var_dump($result);
27 echo "<br>";
28 ?>
```

Hasil dari kode di atas adalah seperti pada gambar 2.7 berikut :



Gambar 2.7: Hasil Keluaran Operasi Aritmatika

Assignment Operators

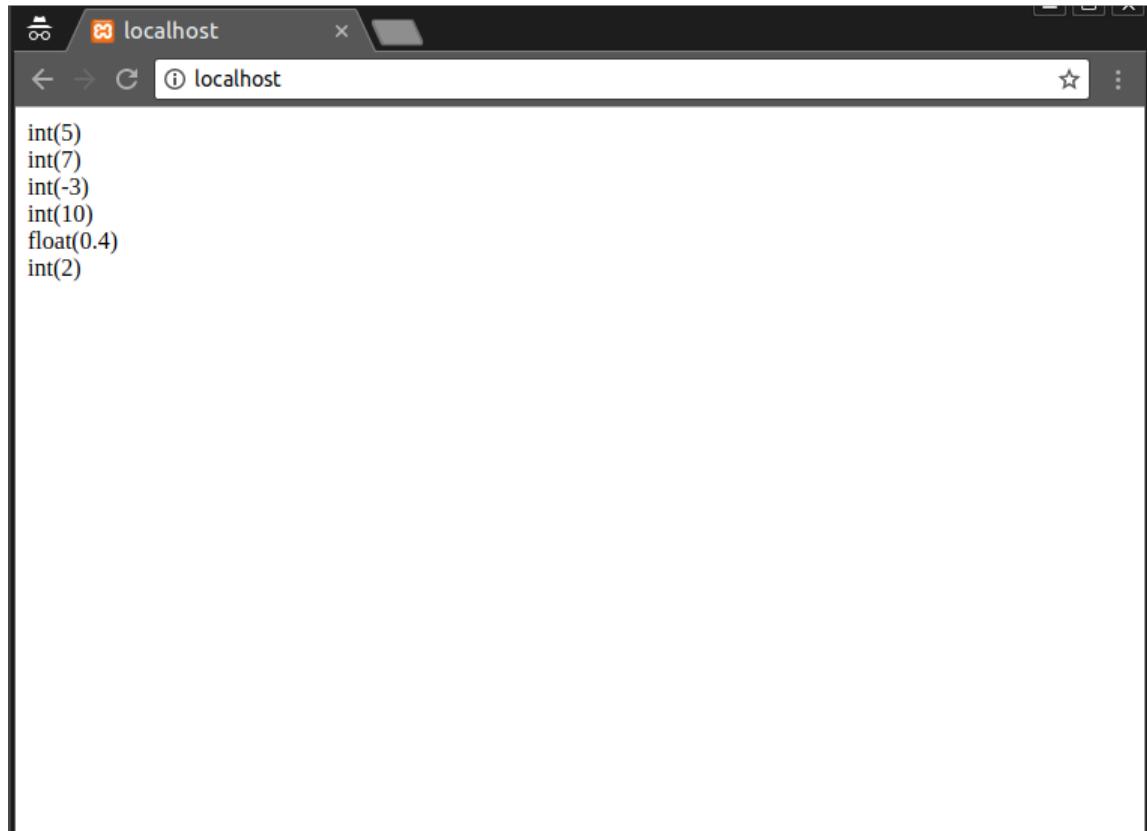
Untuk operator pengisian ditandai dengan tanda = (sama dengan). Nantinya variabel yang ada di sebelah kiri tanda = (sama dengan) akan terisi dengan ekspresi yang ada di sebelah kanan tanda = (sama dengan). Contoh kodennya adalah sebagai berikut :

```
1 <?php
2 $x = 2;
3 $y = 5;
4
5 $x = $y;           // operasi pengisian , dimana variabel x diisikan nilai
6      dari variabel y
7 var_dump($x);
8 echo "<br>";
9 $x = 2;
10
11 $x += $y;          // operasi penambahan , sama seperti x = x + y;
12 var_dump($x);
13 echo "<br>";
14 $x = 2;
15
16 $x -= $y;          // operasi pengurangan , sama seperti x = x - y;
17 var_dump($x);
18 echo "<br>";
```

```

18 $x = 2;
19
20 $x *= $y;           // operasi perkalian , sama seperti x = x * y;
21 var_dump($x);
22 echo "<br>";
23 $x = 2;
24
25 $x /= $y;           // operasi pembagian , sama seperti x = x / y;
26 var_dump($x);
27 echo "<br>";
28 $x = 2;
29
30 $x %= $y;           // operasi sisa bagi , sama seperti x = x % y;
31 var_dump($x);
32 echo "<br>";
33 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 2.8 berikut :



Gambar 2.8: Hasil Keluaran Dari Operator Penugasan

Comparison Operators

Operator ini digunakan untuk membandingkan dua data baik angka maupun teks, berikut adalah operator yang dapat digunakan disertai dengan contohnya :

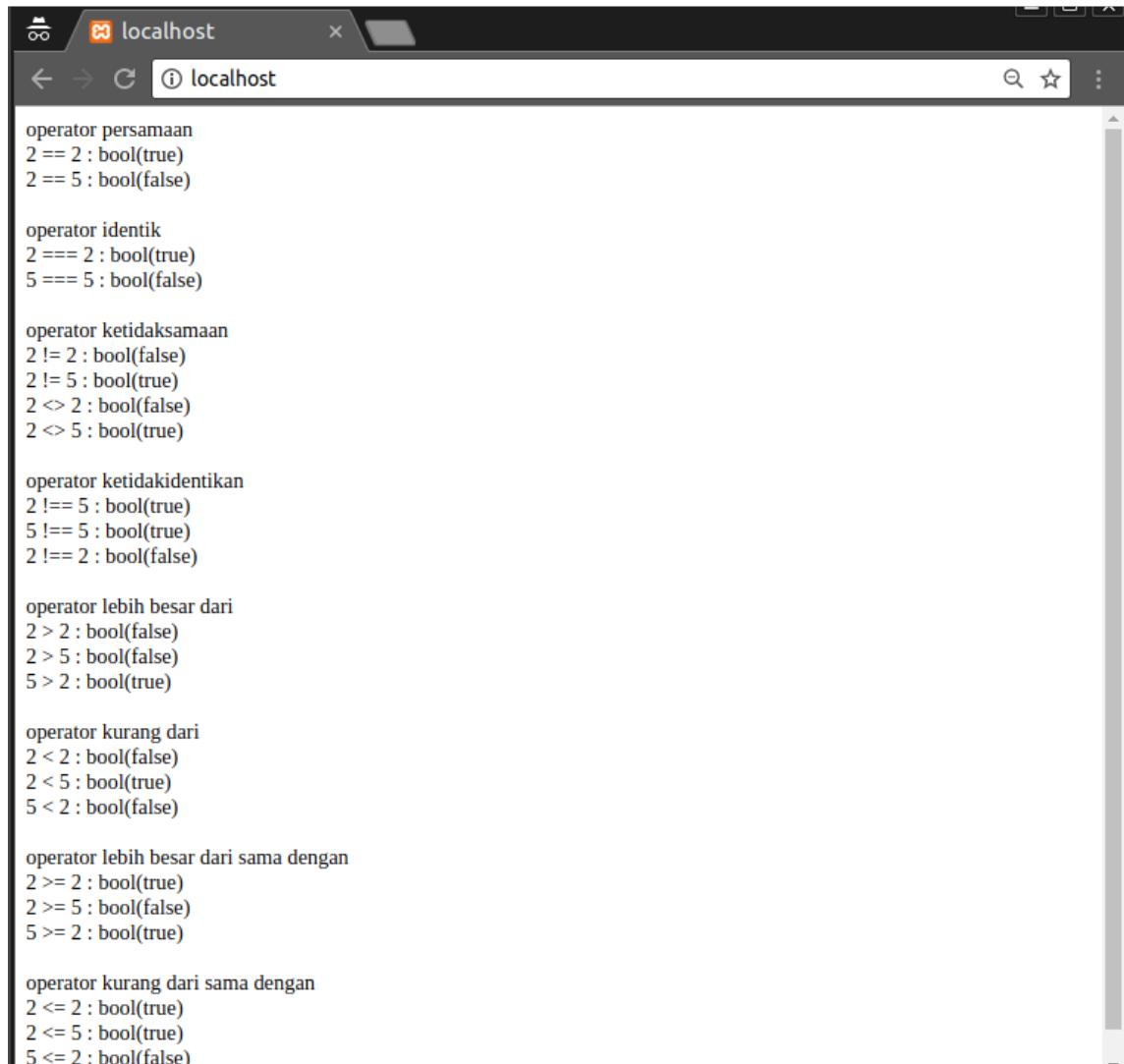
```
1 <?php
2 $w = 2;
3 $x = 2;
4 $y = 5;
5 $z = "5";
6
7 echo "operator persamaan<br>";
8 echo "$w == $x : ";
9 var_dump($w == $x);
10 echo "<br>";
11 echo "$w == $y : ";
12 var_dump($w == $y);
13 echo "<br><br>";
14
15 echo "operator identik<br>";
16 echo "$w === $x : ";
17 var_dump($w === $x);
18 echo "<br>";
19 echo "$y === $z : ";
20 var_dump($y === $z);
21 echo "<br><br>";
22
23 echo "operator ketidaksamaan<br>";
24 echo "$w != $x : ";
25 var_dump($w != $x);
26 echo "<br>";
27 echo "$w != $y : ";
28 var_dump($w != $y);
29 echo "<br>";
30 echo "$w <> $x : ";
31 var_dump($w <> $x);
32 echo "<br>";
33 echo "$w <> $y : ";
34 var_dump($w <> $y);
35 echo "<br><br>";
36
37 echo "operator ketidakidentikan<br>";
38 echo "$w !== $y : ";
39 var_dump($w !== $y);
40 echo "<br>";
41 echo "$y !== $z : ";
42 var_dump($y !== $z);
43 echo "<br>";
44 echo "$w !== $x : ";
45 var_dump($w !== $x);
46 echo "<br><br>";
47
48 echo "operator lebih besar dari<br>";
49 echo "$w > $x : ";
50 var_dump($w > $x);
51 echo "<br>";
52 echo "$w > $y : ";
53 var_dump($w > $y);
54 echo "<br>";
```

```

55 echo "$y > $w : ";
56 var_dump($y > $w);
57 echo "<br><br>";
58
59 echo "operator kurang dari<br>";
60 echo "$w < $x : ";
61 var_dump($w < $x);
62 echo "<br>";
63 echo "$w < $y : ";
64 var_dump($w < $y);
65 echo "<br>";
66 echo "$y < $w : ";
67 var_dump($y < $w);
68 echo "<br><br>";
69
70 echo "operator lebih besar dari sama dengan<br>";
71 echo "$w >= $x : ";
72 var_dump($w >= $x);
73 echo "<br>";
74 echo "$w >= $y : ";
75 var_dump($w >= $y);
76 echo "<br>";
77 echo "$y >= $w : ";
78 var_dump($y >= $w);
79 echo "<br><br>";
80
81 echo "operator kurang dari sama dengan<br>";
82 echo "$w <= $x : ";
83 var_dump($w <= $x);
84 echo "<br>";
85 echo "$w <= $y : ";
86 var_dump($w <= $y);
87 echo "<br>";
88 echo "$y <= $w : ";
89 var_dump($y <= $w);
90 echo "<br><br>";
91 ?>

```

Hasil keluaran dari kode di atas adalah seperti pada gambar 2.9 berikut ini :



The screenshot shows a browser window with the address bar set to 'localhost'. The page content displays the results of a PHP script that tests various comparison operators with values 2 and 5. The output is organized into sections:

- operator persamaan**
2 == 2 : bool(true)
2 == 5 : bool(false)
- operator identik**
2 === 2 : bool(true)
5 === 5 : bool(false)
- operator ketidaksamaan**
2 != 2 : bool(false)
2 != 5 : bool(true)
2 <> 2 : bool(false)
2 <> 5 : bool(true)
- operator ketidakidentikan**
2 !== 5 : bool(true)
5 !== 5 : bool(true)
2 !== 2 : bool(false)
- operator lebih besar dari**
2 > 2 : bool(false)
2 > 5 : bool(false)
5 > 2 : bool(true)
- operator kurang dari**
2 < 2 : bool(false)
2 < 5 : bool(true)
5 < 2 : bool(false)
- operator lebih besar dari sama dengan**
2 >= 2 : bool(true)
2 >= 5 : bool(false)
5 >= 2 : bool(true)
- operator kurang dari sama dengan**
2 <= 2 : bool(true)
2 <= 5 : bool(true)
5 <= 2 : bool(false)

Gambar 2.9: Hasil Kode Operator Perbandingan

Terlihat bahwa operator ini akan membandingkan dua nilai yang ada, yang perlu diperhatikan adalah bahwa operator persamaan dengan operator identik adalah bahwa operator identik akan memeriksa persamaannya sampai ke jenis tipe data. Apabila tipe datanya berbeda, maka akan dianggap tidak identik.

Sedangkan untuk operator ketidakidentikan, maka operator akan memeriksa, apabila datanya memang berbeda, maka akan memberikan nilai `true`, kemudian apabila datanya sama, namun tipe datanya berbeda, maka akan memberikan nilai `true` pula.

Increment / Decrement Operators

Operator ini digunakan untuk menambahkan nilai atau mengurangkan nilai 1 ke dalam variabel. Berikut contoh kodenya :

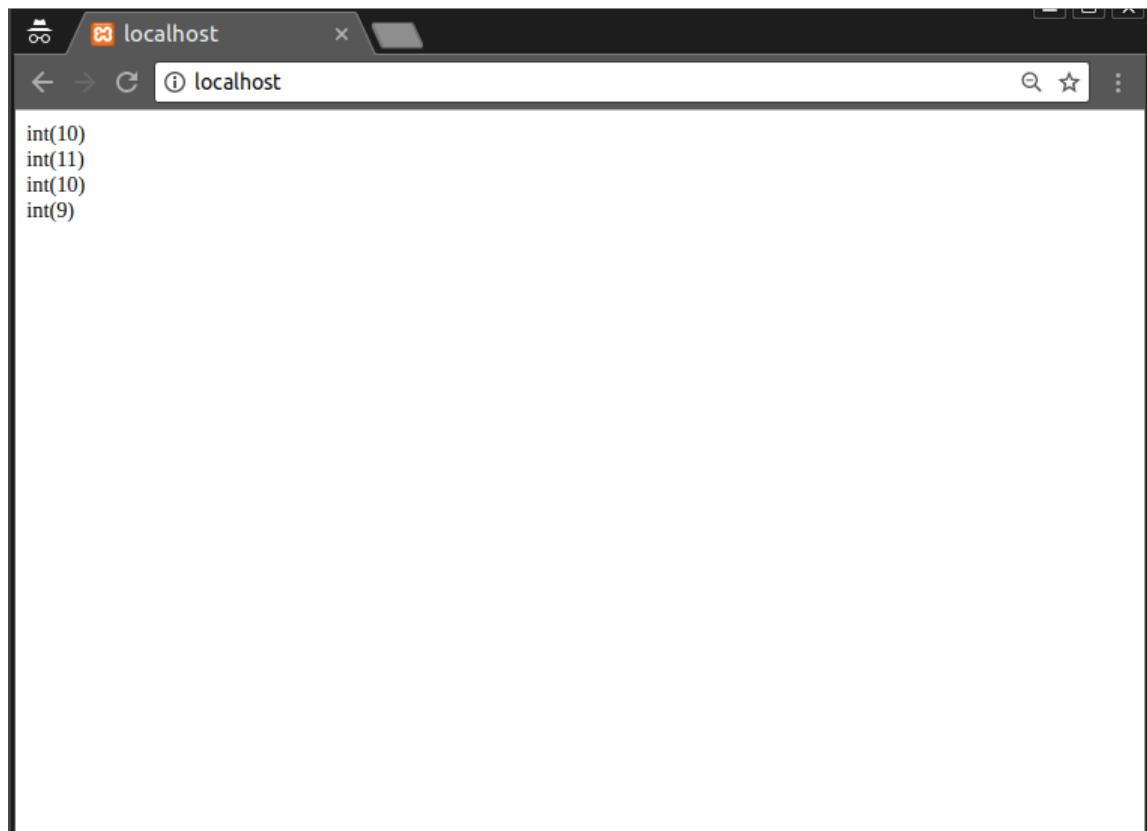
```

1 <?php
2 $nilai = 10;

```

```
3 var_dump($nilai++);  
4 echo "<br>";  
5  
6 $nilai = 10;  
7 var_dump(++$nilai);  
8 echo "<br>";  
9  
10 $nilai = 10;  
11 var_dump($nilai--);  
12 echo "<br>";  
13  
14 $nilai = 10;  
15 var_dump(--$nilai);  
16 echo "<br>";  
17 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 2.10 berikut :



Gambar 2.10: Hasil Operator Penambahan dan Pengurangan

Terlihat bahwa pada saat operator di tambahkan di belakang variabel, maka proses penambahan atau pengurangan dilakukan setelah ekspresi, sedangkan saat operator di tambahkan di depan variabel, maka proses penambahan atau pengurangan dilakukan sebelum ekspresi dieksekusi.

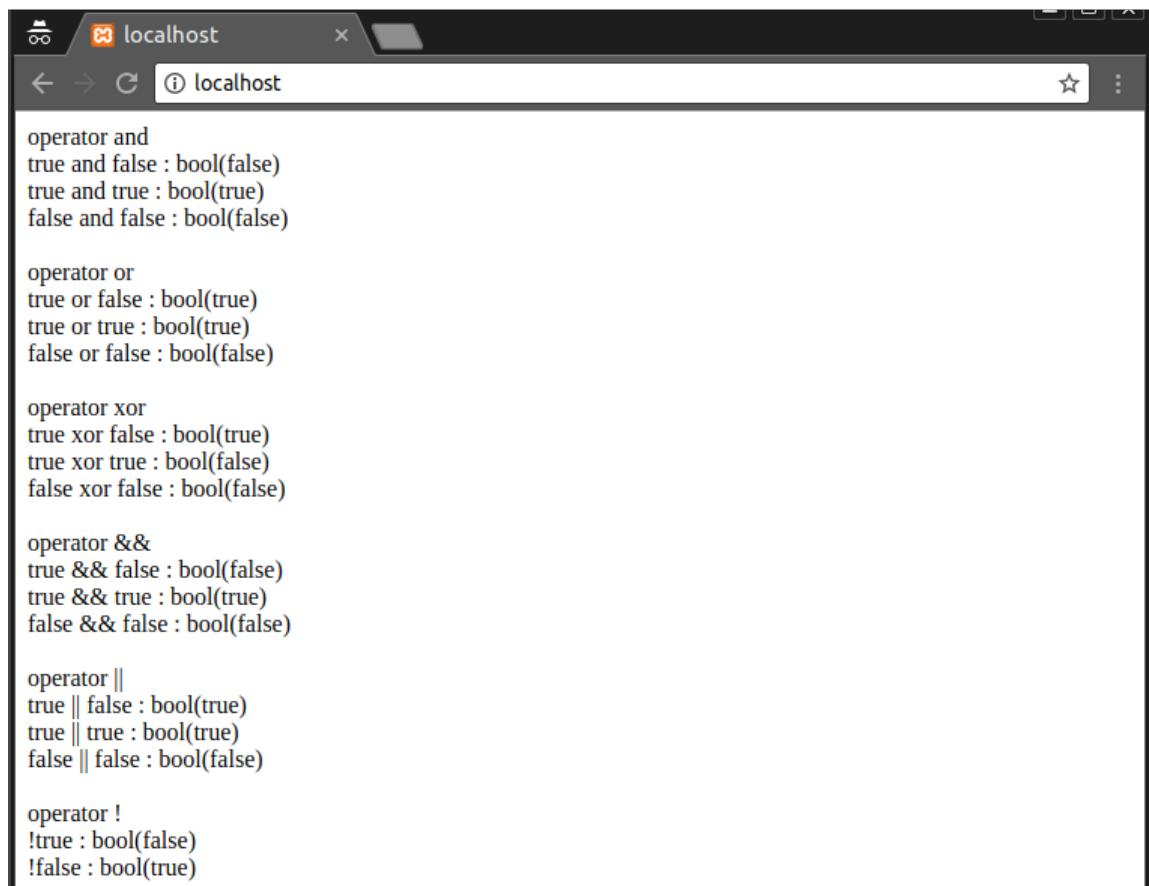
Logical Operators

Operator logika digunakan untuk melakukan pemeriksaan benar dan salah dari dua buah data. Contoh kode untuk penggunaan operator logika ini adalah sebagai berikut :

```

1 <?php
2 $a = true;
3 $b = false;
4 $c = true;
5 $d = false;
6
7 echo "operator and<br>";
8 echo "true and false : "; var_dump($a and $b); echo "<br>";
9 echo "true and true : "; var_dump($a and $c); echo "<br>";
10 echo "false and false : "; var_dump($b and $d); echo "<br><br>";
11
12 echo "operator or<br>";
13 echo "true or false : "; var_dump($a or $b); echo "<br>";
14 echo "true or true : "; var_dump($a or $c); echo "<br>";
15 echo "false or false : "; var_dump($b or $d); echo "<br><br>";
16
17 echo "operator xor<br>";
18 echo "true xor false : "; var_dump($a xor $b); echo "<br>";
19 echo "true xor true : "; var_dump($a xor $c); echo "<br>";
20 echo "false xor false : "; var_dump($b xor $d); echo "<br><br>";
21
22 echo "operator &&<br>";
23 echo "true && false : "; var_dump($a && $b); echo "<br>";
24 echo "true && true : "; var_dump($a && $c); echo "<br>";
25 echo "false && false : "; var_dump($b && $d); echo "<br><br>";
26
27 echo "operator ||<br>";
28 echo "true || false : "; var_dump($a || $b); echo "<br>";
29 echo "true || true : "; var_dump($a || $c); echo "<br>";
30 echo "false || false : "; var_dump($b || $d); echo "<br><br>";
31
32 echo "operator !<br>";
33 echo "!true : "; var_dump(!$a); echo "<br>";
34 echo "!false : "; var_dump(!$b); echo "<br>";
35 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 2.11 berikut :



The screenshot shows a browser window titled 'localhost' with the URL 'localhost'. The page content displays the results of a PHP script. It lists the behavior of various logical operators:

- operator and**
true and false : bool(false)
true and true : bool(true)
false and false : bool(false)
- operator or**
true or false : bool(true)
true or true : bool(true)
false or false : bool(false)
- operator xor**
true xor false : bool(true)
true xor true : bool(false)
false xor false : bool(false)
- operator &&**
true && false : bool(false)
true && true : bool(true)
false && false : bool(false)
- operator ||**
true || false : bool(true)
true || true : bool(true)
false || false : bool(false)
- operator !**
!true : bool(false)
!false : bool(true)

Gambar 2.11: Hasil Keluaran Dari Operator Logika

Terlihat bahwa operator **and** dan **&&** akan menghasilkan nilai yang sama, sedangkan operator **or** dan **||** akan menghasilkan nilai yang sama pula.

Untuk operator **xor**, maka apabila ada salah satu data yang bernilai **true** maka akan mengembalikan nilai **true**, namun apabila nilai kedua data adalah **false** atau **true**, maka hasilnya akan bernilai **false**.

Operator **!** (tanda seru) akan menegaskan hasil datanya.

String Operators

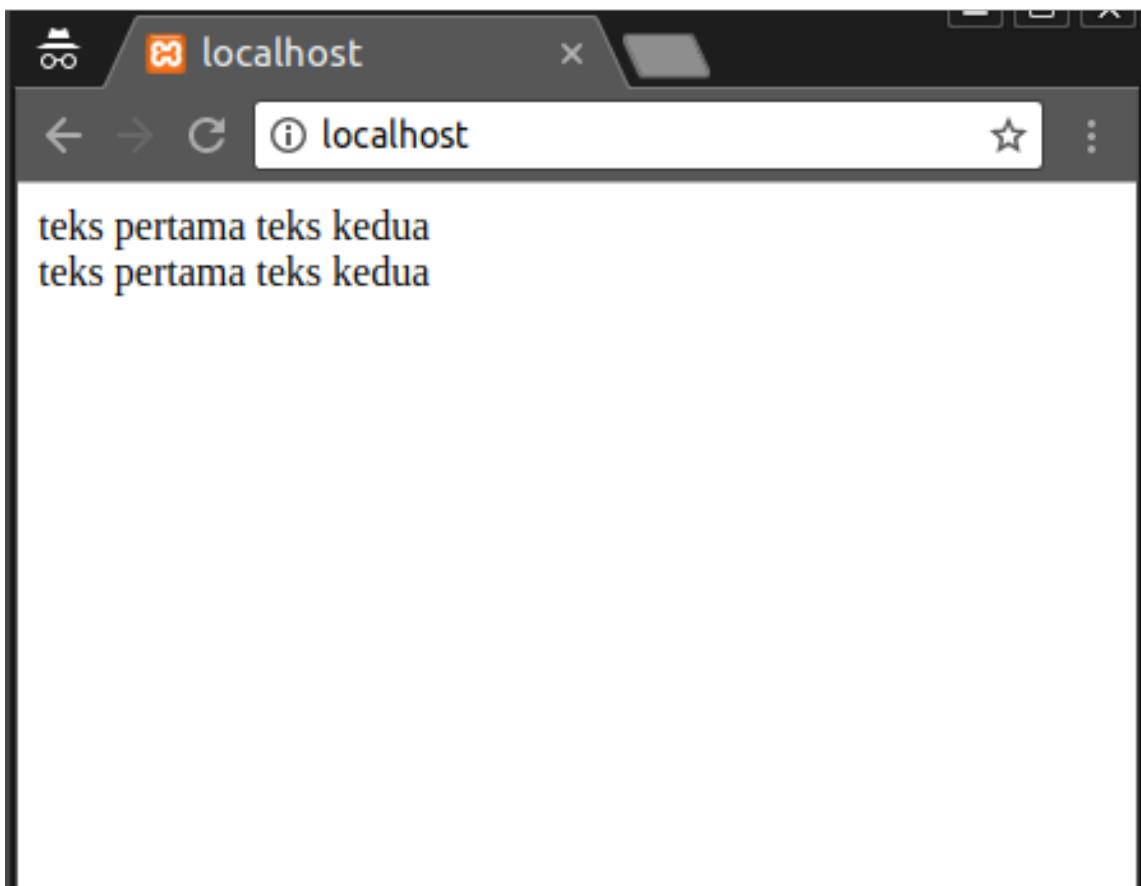
Operator teks sederhana yang disediakan oleh PHP adalah operator untuk menggabungkan 2 (dua) teks menjadi 1 rangkaian. Contoh kodennya adalah sebagai berikut :

```

1 <?php
2 $teks1 = "teks pertama ";
3 $teks2 = "teks kedua";
4
5 echo $teks1.$teks2; echo "<br>";
6 $teks1 .= $teks2;
7 echo $teks1;
8 ?>

```

Hasil keluaran dari kode di atas adalah seperti pada gambar 2.12 berikut :



Gambar 2.12: Hasil Kode Penggabungan Teks

Operatornya hanya sebuah . (titik) untuk menggabungkan 2 (dua) teks menjadi satu, apabila hasilnya akan dimasukkan ke variabel pertama, maka dapat menggunakan operator .= (titik sama dengan).

Array Operators

Beberapa operator yang dapat digunakan untuk *array* seperti terlihat pada contoh kode berikut :

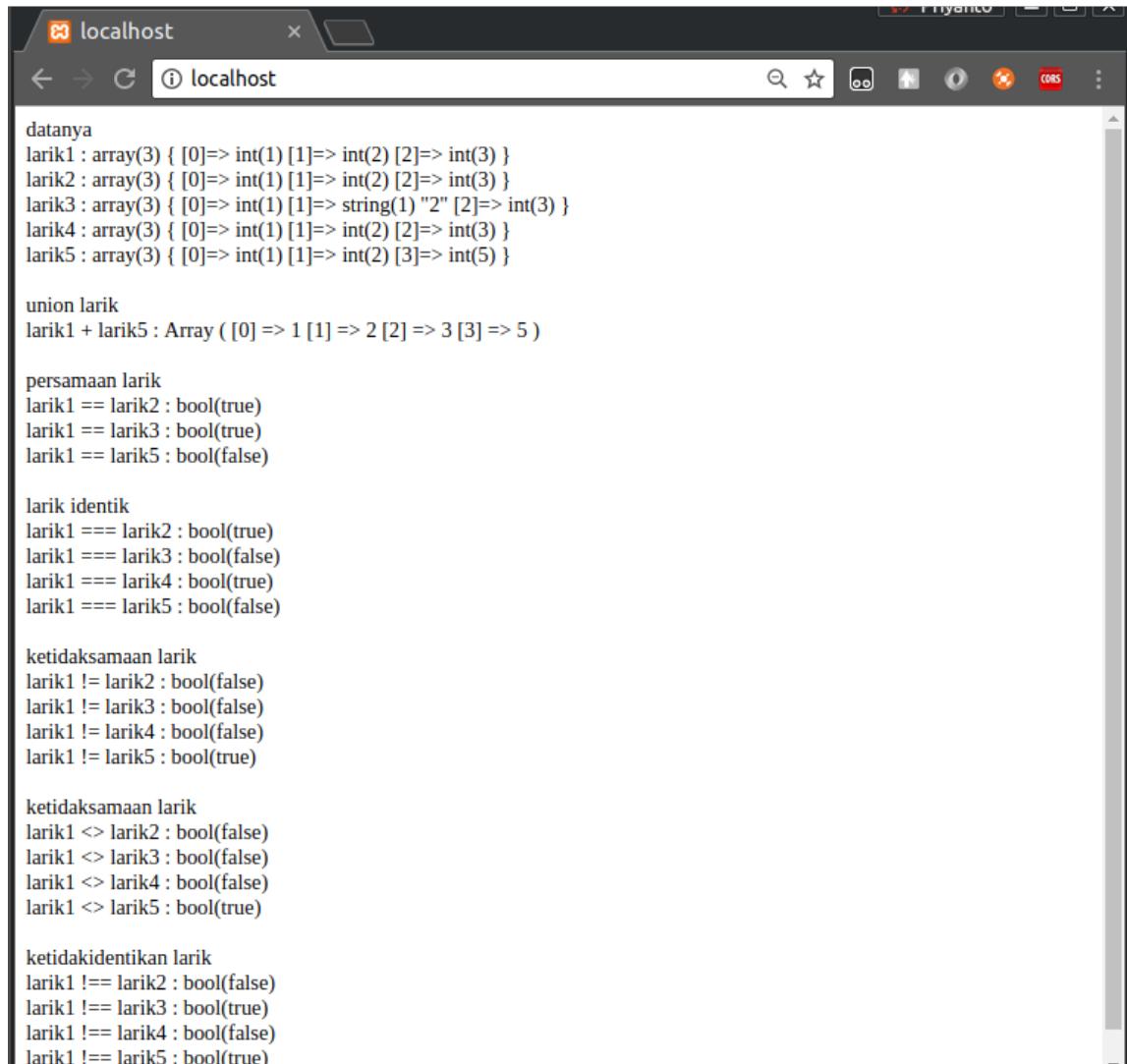
```
1 <?php
2 $larik1 = array(1, 2, 3);
3 $larik2 = array(1, 2, 3);
4 $larik3 = array(1, "2", 3);
5 $larik4 = array("0" => 1, 2, 3);
6 $larik5 = array(0 => 1, 1 => 2, 3 => 5);
7
8 echo "datanya<br>";
9 echo "larik1 : "; var_dump($larik1); echo "<br>";
```

```

10 echo "larik2 : "; var_dump($larik2); echo "<br>" ;
11 echo "larik3 : "; var_dump($larik3); echo "<br>" ;
12 echo "larik4 : "; var_dump($larik4); echo "<br>" ;
13 echo "larik5 : "; var_dump($larik5); echo "<br><br>" ;
14
15 echo "union larik<br>" ;
16 echo "larik1 + larik5 : "; print_r($larik1 + $larik5); echo "<br><br>" ;
17
18 echo "persamaan larik<br>" ;
19 echo "larik1 == larik2 : "; var_dump($larik1 == $larik2); echo "<br>" ;
20 echo "larik1 == larik3 : "; var_dump($larik1 == $larik3); echo "<br>" ;
21 echo "larik1 == larik5 : "; var_dump($larik1 == $larik5); echo "<br><br>" ;
22
23 echo "larik identik<br>" ;
24 echo "larik1 == larik2 : "; var_dump($larik1 == $larik2); echo "<br>" ;
25 echo "larik1 == larik3 : "; var_dump($larik1 == $larik3); echo "<br>" ;
26 echo "larik1 == larik4 : "; var_dump($larik1 == $larik4); echo "<br>" ;
27 echo "larik1 == larik5 : "; var_dump($larik1 == $larik5); echo "<br><br>" ;
28
29 echo "ketidak samaan larik<br>" ;
30 echo "larik1 != larik2 : "; var_dump($larik1 != $larik2); echo "<br>" ;
31 echo "larik1 != larik3 : "; var_dump($larik1 != $larik3); echo "<br>" ;
32 echo "larik1 != larik4 : "; var_dump($larik1 != $larik4); echo "<br>" ;
33 echo "larik1 != larik5 : "; var_dump($larik1 != $larik5); echo "<br><br>" ;
34
35 echo "ketidak samaan larik<br>" ;
36 echo "larik1 <>> larik2 : "; var_dump($larik1 <>> $larik2); echo "<br>" ;
37 echo "larik1 <>> larik3 : "; var_dump($larik1 <>> $larik3); echo "<br>" ;
38 echo "larik1 <>> larik4 : "; var_dump($larik1 <>> $larik4); echo "<br>" ;
39 echo "larik1 <>> larik5 : "; var_dump($larik1 <>> $larik5); echo "<br><br>" ;
40
41 echo "ketidakidentikan larik<br>" ;
42 echo "larik1 !== larik2 : "; var_dump($larik1 !== $larik2); echo "<br>" ;
43 echo "larik1 !== larik3 : "; var_dump($larik1 !== $larik3); echo "<br>" ;
44 echo "larik1 !== larik4 : "; var_dump($larik1 !== $larik4); echo "<br>" ;
45 echo "larik1 !== larik5 : "; var_dump($larik1 !== $larik5); echo "<br><br>" ;
46 ?>

```

Hasil keluaran dari kode di atas adalah seperti pada gambar 2.13 berikut :



The screenshot shows a browser window with the URL `localhost`. The page content displays the result of an array operation. It includes several sections of code and their resulting outputs:

- datanya**
larik1 : array(3) { [0]=> int(1) [1]=> int(2) [2]=> int(3) }
larik2 : array(3) { [0]=> int(1) [1]=> int(2) [2]=> int(3) }
larik3 : array(3) { [0]=> int(1) [1]=> string(1) "2" [2]=> int(3) }
larik4 : array(3) { [0]=> int(1) [1]=> int(2) [2]=> int(3) }
larik5 : array(3) { [0]=> int(1) [1]=> int(2) [3]=> int(5) }
- union larik**
`larik1 + larik5 : Array ([0] => 1 [1] => 2 [2] => 3 [3] => 5)`
- persamaan larik**
larik1 == larik2 : bool(true)
larik1 == larik3 : bool(true)
larik1 == larik5 : bool(false)
- larik identik**
larik1 === larik2 : bool(true)
larik1 === larik3 : bool(false)
larik1 === larik4 : bool(true)
larik1 === larik5 : bool(false)
- ketidaksamaan larik**
larik1 != larik2 : bool(false)
larik1 != larik3 : bool(false)
larik1 != larik4 : bool(false)
larik1 != larik5 : bool(true)
- ketidakidentikan larik**
larik1 !== larik2 : bool(false)
larik1 !== larik3 : bool(true)
larik1 !== larik4 : bool(false)
larik1 !== larik5 : bool(true)

Gambar 2.13: Hasil Keluaran Operator *Array* / Larik

2.4 Kesimpulan

Bahwa variabel dapat dibentuk untuk menyimpan sebuah data dengan tipe data tertentu, yang kemudian datanya dapat diolah menggunakan operator sederhana untuk menghasilkan data baru.

2.5 Tugas

Cobalah untuk membuat beberapa variabel dengan data sebagai berikut :

Nama Variabel	Keterangan
nama	diisi nama pribadi
jurusan	diisi jurusan program studi
usia	diisi usia sekarang

Kemudian cetak hasilnya ke *browser* dengan menggunakan echo.

Bab 3

Array

3.1 Tujuan

Mahasiswa mampu mengimplementasikan penggunaan *array* / larik pada PHP untuk alur program sederhana.

3.2 Pengantar

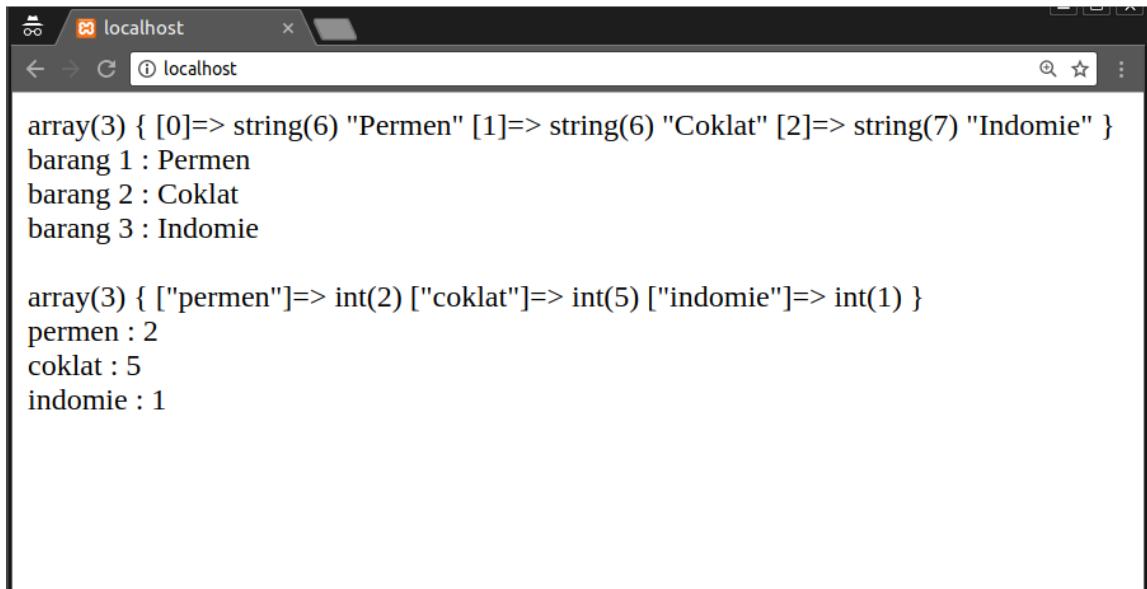
Array atau larik adalah tempat untuk menampung sekumpulan data, baik yang setipe, ataupun yang berbeda tipe. Struktur *array* / larik sendiri sebetulnya tiap elemen data terdiri dari *key* dan *value*.

3.3 Praktek

Cara membuat *array* / larik secara sederhana adalah sebagai berikut :

```
1 <?php
2
3 $daftarBarang = array("Permen", "Coklat", "Indomie");
4 $stok = array("permen" => 2, "coklat" => 5, "indomie" => 1);
5
6 var_dump($daftarBarang); echo "<br>";
7 echo "barang 1 : $daftarBarang[0]<br>";
8 echo "barang 2 : $daftarBarang[1]<br>";
9 echo "barang 3 : $daftarBarang[2]<br><br>";
10
11 var_dump($stok); echo "<br>";
12 echo "permen : $stok[permen]<br>";
13 echo "coklat : $stok[coklat]<br>";
14 echo "indomie : $stok[indomie]<br>";
15
16 ?>
```

Hasil dari kode di atas adalah seperti pada gambar 3.1 berikut ini :

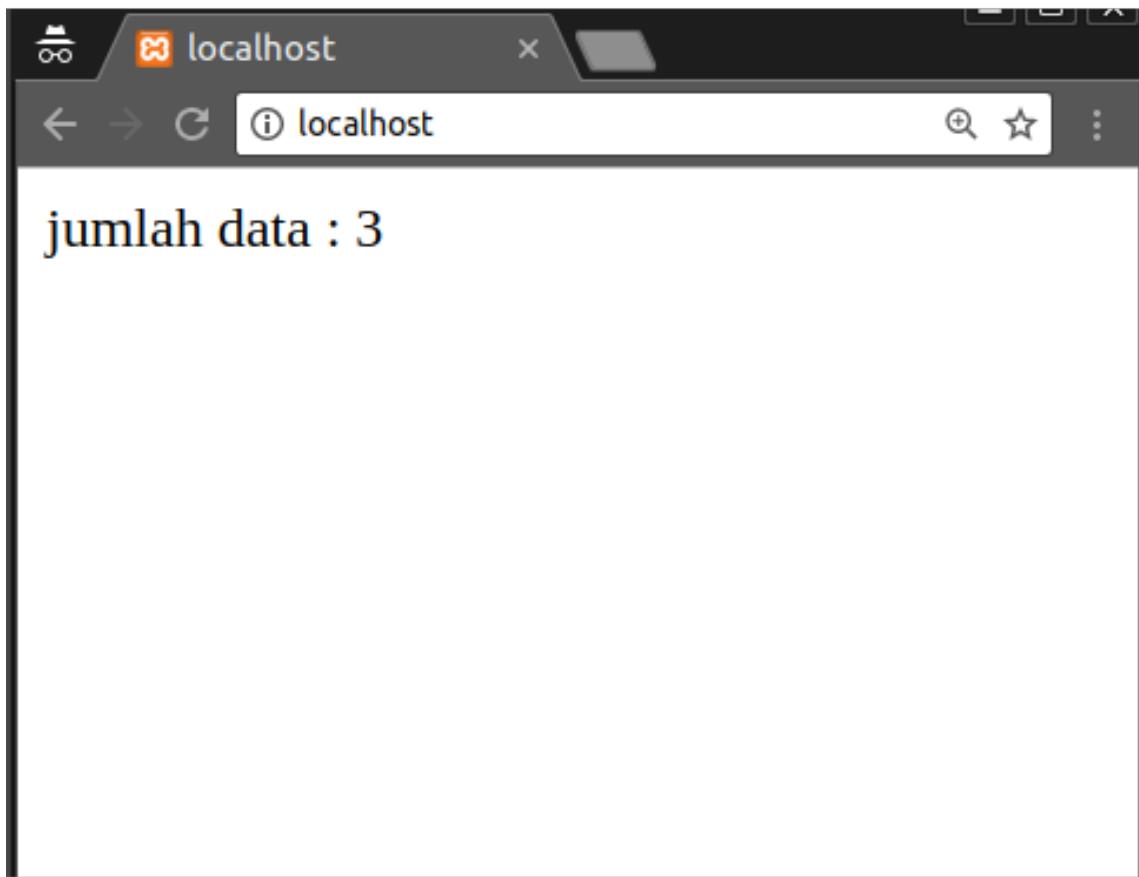


Gambar 3.1: Hasil Keluaran *Array / Larik*

Untuk mengetahui jumlah data yang ada pada *array* kita dapat menggunakan perintah berikut ini :

```
1 <?php
2 $warna = array("merah", "kuning", "hijau");
3
4 echo "jumlah data : "; echo count($warna);
5 ?>
```

Hasil keluaran dari kode tersebut adalah seperti pada gambar 3.2 berikut :



Gambar 3.2: Hasil Keluaran Melihat Jumlah Data Pada *Array*

3.4 Kesimpulan

Apabila ingin menggunakan variabel yang mampu menyimpan beberapa data sekali-gus, maka penggunaan *array* sudah tepat. Beberapa operasi untuk *array* ini pun sudah disediakan oleh PHP.

3.5 Tugas

Cobalah membuat himpunan data dengan *array* yang berisi pasangan data *nim* dan *nama* dengan jumlah data minimal 5 (lima) data.

Bab 4

Struktur Kontrol

4.1 Tujuan

Mahasiswa mampu memahami dan menggunakan kontrol pada PHP.

4.2 Pengantar

PHP dibangun dengan menyediakan beberapa fasilitas seperti seleksi, perulangan, fungsi dan sebagainya. Pada bab ini kita akan bahas bagaimana struktur seleksi dan perulangan di PHP.

4.3 Praktek

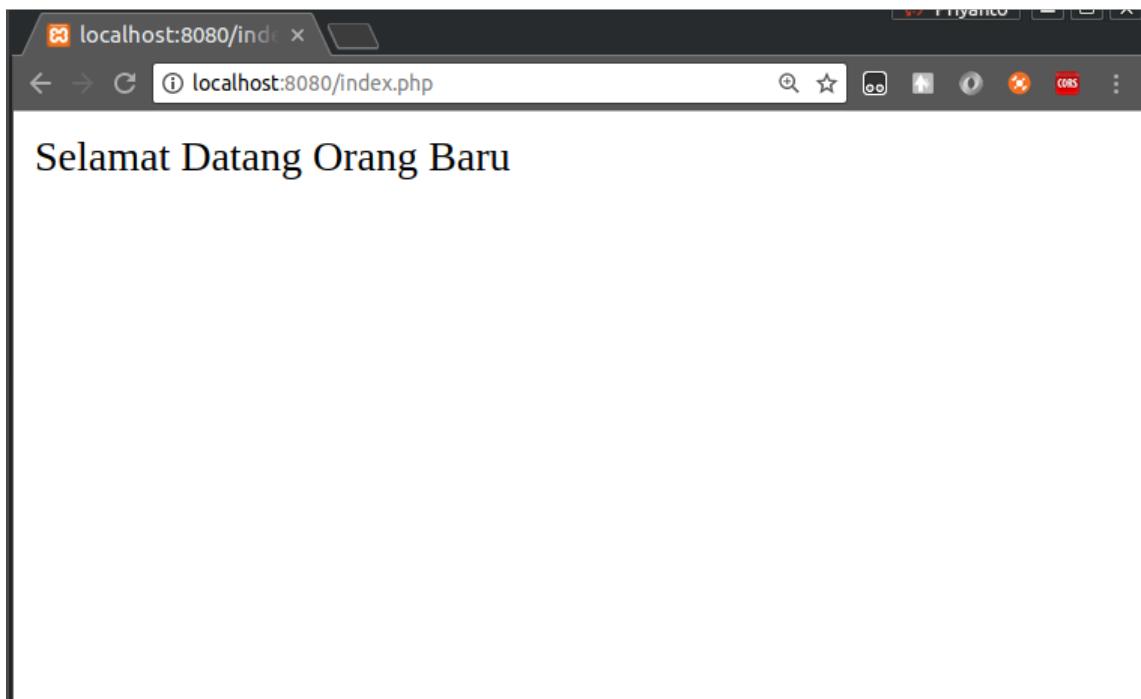
4.3.1 Seleksi / Percabangan if

Seleksi atau percabangan *if* digunakan untuk mengontrol alur berdasarkan logika *true* atau *false*.

Perhatikan kode berikut ini :

```
1 <?php  
2 $orangBaru = true;  
3  
4 if ($orangBaru) {  
5     echo "Selamat Datang Orang Baru";  
6 }  
7 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 4.1 berikut :



Gambar 4.1: Hasil Keluaran *if*

Dari hasil keluaran tersebut bahwa apabila nilai yang ada di antara tanda kurung (...) bernilai **true**, maka perintah yang berada di antara tanda kurung kurawal (...) akan dijalankan. Namun bila nilai yang berada di dalam tanda kurung (...) bernilai **false**, maka tidak akan menampilkan apapun di *browser*.

Bentuk lain dari *if* ini adalah *if...else*, contoh kodennya sebagai berikut :

```

1 <?php
2 $a = 5;
3 $b = 7;
4
5 if($a > $b) {
6     echo "$a lebih kecil dari $b";
7 } else {
8     echo "$a lebih besar dari $b";
9 }
10 ?>
```

Hasil keluaran untuk kode di atas adalah seperti pada gambar ?? berikut ini :

```

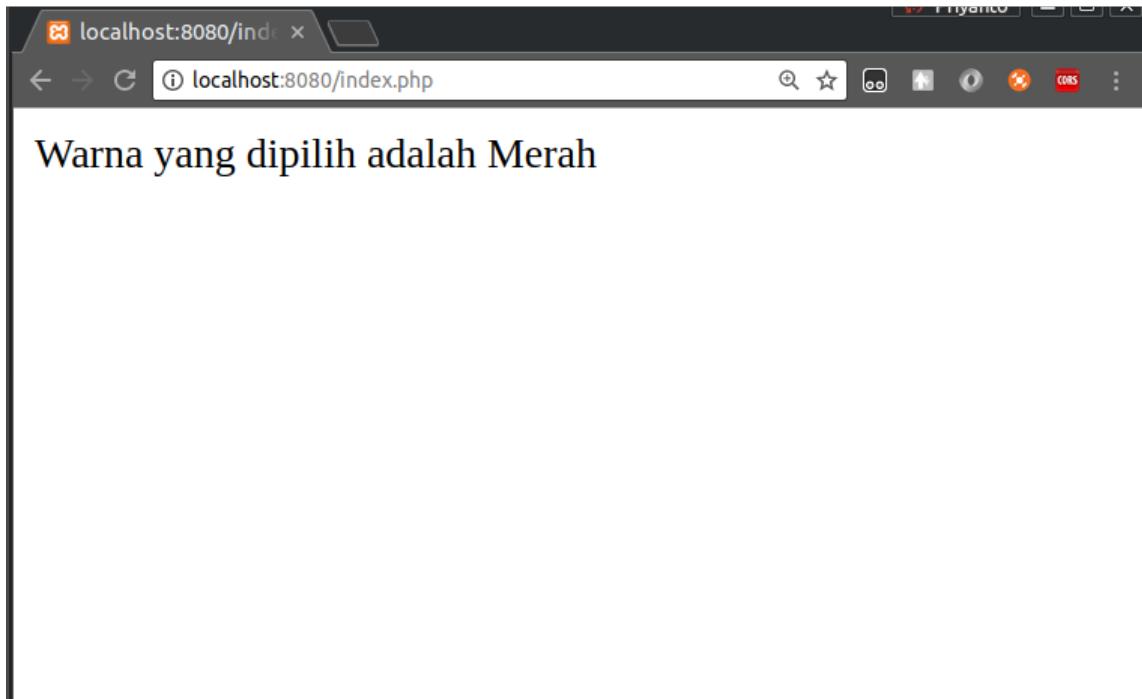
1 \centering
2 \includegraphics [width=1\textwidth]{04-03-01-002-hasil2}
3 \caption{Hasil Keluaran Contoh Kode Menggunakan \texttt{\{else\}}}
4 \label{fig:04-03-01-002}
```

Terlihat bahwa karena pernyataan yang ada di dalam tanda kurung *if(...)* menghasilkan nilai **false**, maka yang ditampilkan adalah perintah pada blok *else*.

Apabila kita ingin menyeleksi lebih dari dua pilihan **true** dan **false** saja, kita bisa menggunakan seleksi **elseif**, berikut contoh kodennya :

```
1 <?php
2 $a = "merah";
3
4 if ($a == "biru") {
5     echo "Warna yang dipilih adalah Biru";
6 } elseif($a == "merah") {
7     echo "Warna yang dipilih adalah Merah";
8 } else {
9     echo "Tidak memilih warna";
10}
11?>
```

Hasil keluaran untuk kode di atas adalah seperti pada gambar 4.2 berikut ini :



Gambar 4.2: Hasil Keluaran Perintah elseif

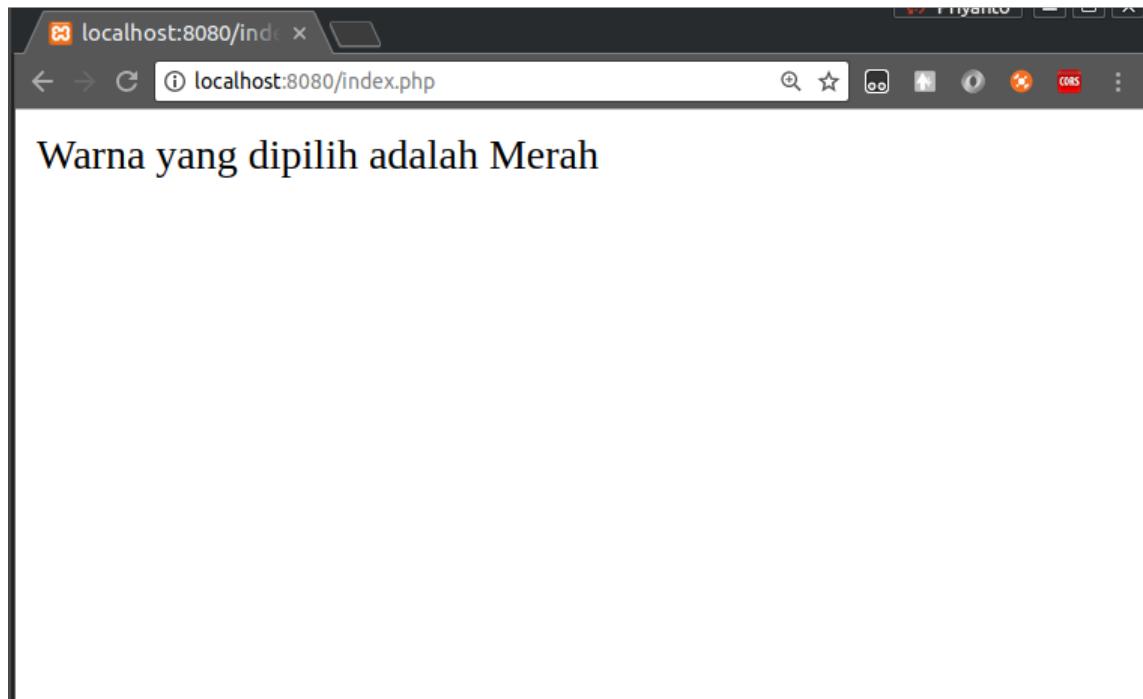
Yang dijalankan adalah perintah yang berada dalam blok `elseif` karena memenuhi kriteria bahwa nilai dari variabel `$a` adalah `merah`, apabila nilai dari variabel tersebut adalah `biru` tentu yang akan dijalankan adalah ekspresi dalam blok `if`, namun bila isi dari variabel `$a` bukan `merah` atau `biru` maka yang dijalankan adalah ekspresi dalam blok `else`.

Kode yang lain untuk melakukan seleksi dalam 1 (satu) baris adalah sebagai berikut :

```
1 <?php
2 $a = "merah";
3
4 $result = ($a == "merah") ? "Warna yang dipilih adalah Merah" : "Warna
5     yang dipilih adalah Merah";
6 echo $result;
```

```
6 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 4.3 berikut ini :



Gambar 4.3: Hasil Keluaran Ekspresi if Dalam Bentuk Lain

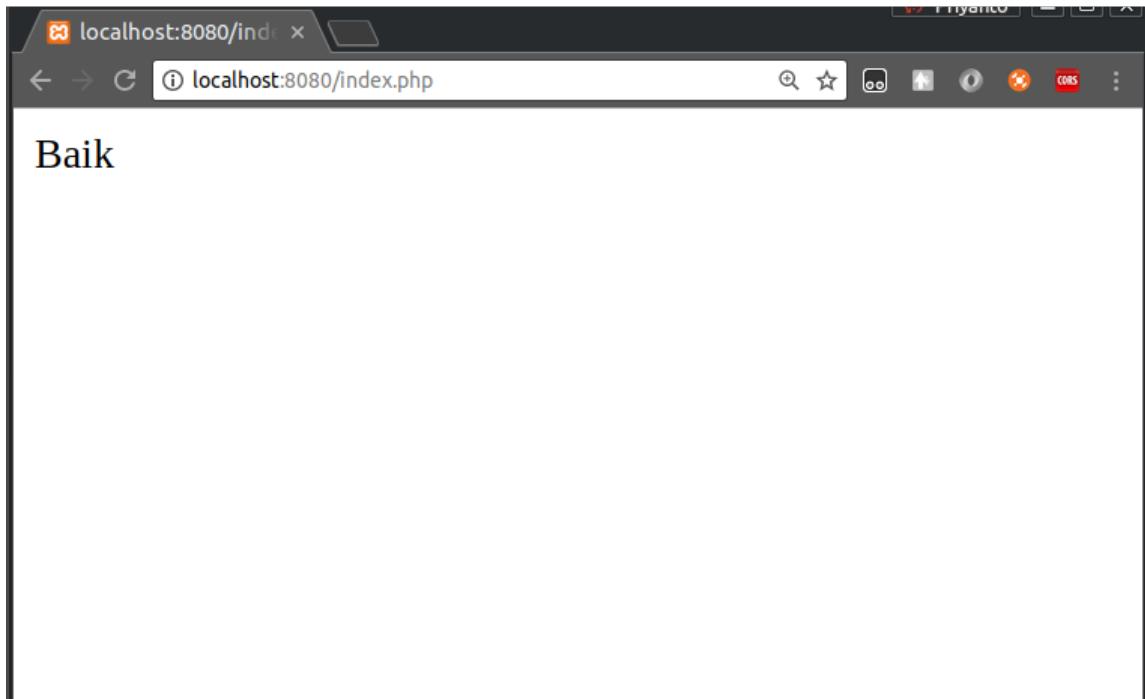
4.3.2 Seleksi / Percabangan switch

Perintah **switch** tujuan mirip dengan **if**, digunakan untuk melakukan seleksi atau percabangan alur program berdasarkan kondisi **true** atau **false**. Contoh sederhana dari penggunaan **switch** ini adalah sebagai berikut :

```
1 <?php
2 $nilai = 7;
3
4 switch($nilai) {
5     case 0:
6     case 1:
7     case 2:
8         echo "Sangat Kurang";
9         break;
10    case 3:
11    case 4:
12        echo "Kurang";
13        break;
14    case 5:
15    case 6:
16        echo "Cukup";
17        break;
18    case 7:
```

```
19 case 8:  
20     echo "Baik";  
21     break;  
22 case 9:  
23 case 10:  
24     echo "Sangat Baik";  
25     break;  
26 default :  
27     echo "Tanpa Nilai";  
28 }  
29 ?>
```

Hasil keluaran dari kode di atas adalah seperti gambar 4.4 berikut ini :



Gambar 4.4: Hasil Keluaran Ekspresi **switch**

Terlihat bahwa pada saat variabel \$nilai berisi angka 7, seleksi **switch** akan melakukan pemeriksaan nilai variabel ini, nilainya akan dibandingkan dengan tiap **case**, apabila cocok, maka blok **case** tersebut yang akan dijalankan.

Karena **case** 7 kosong dan tidak ada perintah **break**, maka **case** berikutnya akan dijalankan, yaitu **case** 8.

Apabila nilai dari variabel \$nilai tidak ada nilai yang cocok pada ekspresi **case**, maka pernyataan pada blok **default** yang akan dikerjakan.

4.3.3 Perulangan while

Perintah **while** digunakan untuk melakukan perulangan selama nilai yang ada pada parameternya bernilai **true**. Struktur dasarnya seperti ini :

```

1 while( ekspresi) {
2     ...
3 }
```

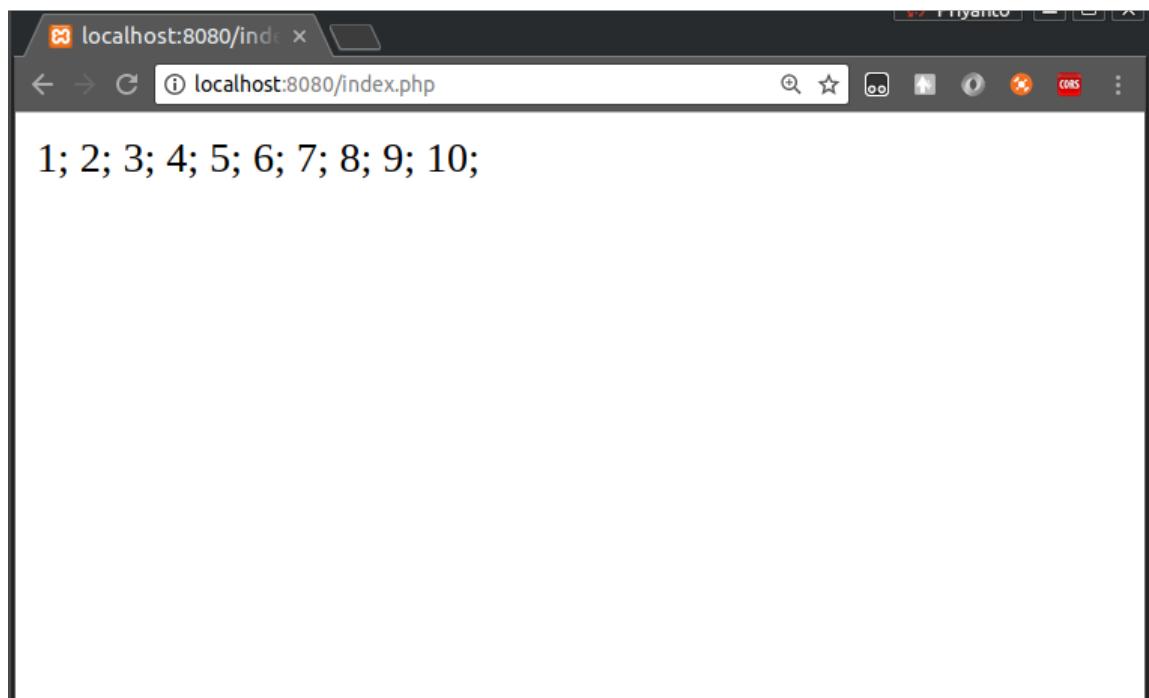
Nilai pada bagian `ekspresi` harus bernilai `true` agar perulangan dapat dijalankan, apabila dalam prosesnya nilai pada `ekspresi` bernilai `false` maka perulangan akan berhenti.

Contoh kode programnya adalah seperti berikut :

```

1 <?php
2 $nilai = 1;
3
4 while( $nilai <= 10) {
5     echo $nilai++ . " ";
6 }
7 ?>
```

Hasil dari kode di atas adalah seperti pada gambar 4.5 berikut ini :



Gambar 4.5: Hasil Keluaran Contoh Kode `while`

4.3.4 Perulangan `do-while`

Perulangan dengan menggunakan `do..while` mirip seperti struktur `while`, hanya saja pemeriksaan `ekspresi` akan dilakukan setelah proses iterasi pertama berjalan, struktur kode secara sederhana dapat dituliskan seperti berikut :

```

1 do {
2 ...
3 } while( ekspresi);
```

Dimana nilai dari ekspresi agar iterasi berlanjut ke putaran ke-2 dan seterusnya harus bernilai `true`. Contoh implementasi kodenya adalah seperti berikut :

```

1 <?php
2 $nilai = 1;
3
4 do {
5     echo "nilai urut ke—" . $nilai++ . "<br>";
6 } while ($nilai <= 10)
7 ?>
```

Hasil keluaran untuk kode di atas adalah seperti pada gambar 4.6 berikut :



Gambar 4.6: Hasil Keluaran Kode `do..while`

4.3.5 Perulangan `for`

Penggunaan perintah `for` sama seperti `while` atau `do..while`, hanya saja strukturnya sedikit berbeda, mari kita lihat struktur dasar dari perintah `for` berikut :

```

1 for (eksp1; eksp2; eksp3) {
2 ...
3 }
```

Bagian `eksp1` akan dieksekusi di awal sebelum iterasi di mulai.

Begian `eksp2` adalah tempat pemeriksaan kondisi di awal tiap iterasi, pada iterasi pertama, tepat setelah `eksp1` dikerjakan. Bila `eksp2` bernilai `true` maka blok iterasi di bawahnya akan dikerjakan, namun apabila bernilai `false` maka blok iterasi di bawahnya tidak akan dijalankan.

Bagian `eksp3` akan dikerjakan di akhir tiap siklus iterasi.

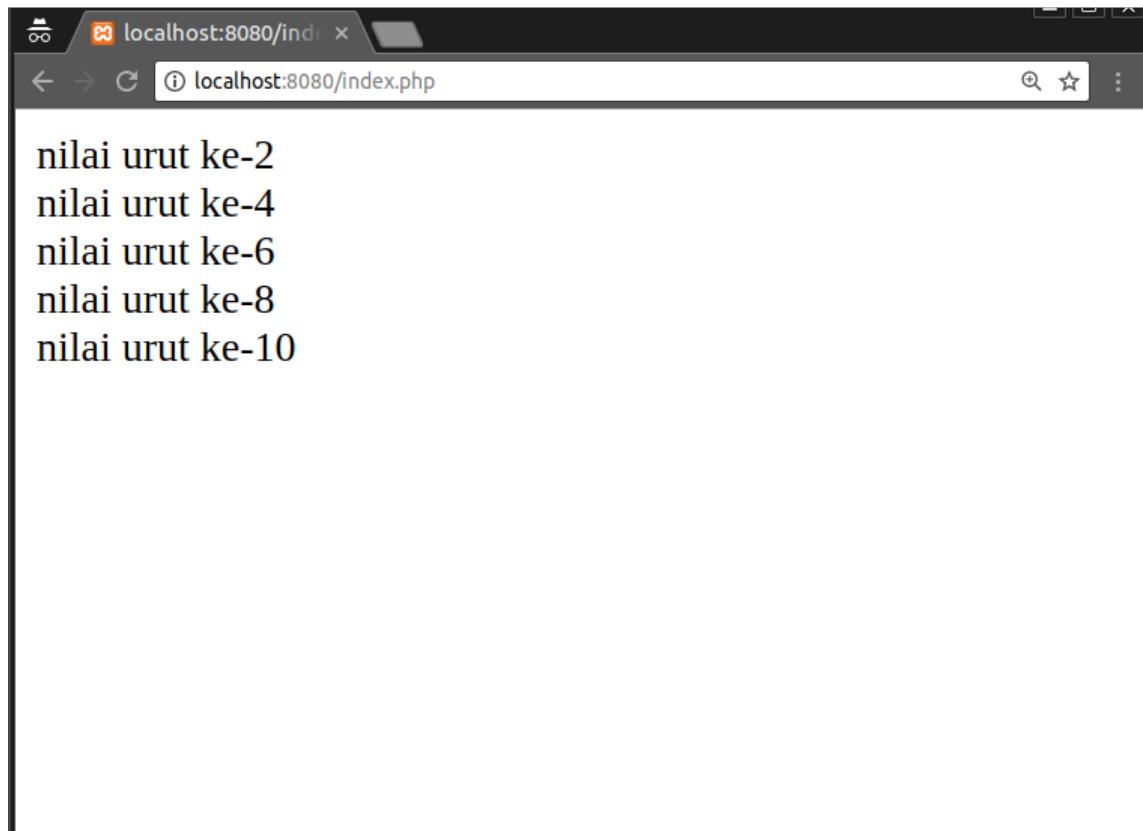
Contoh implementasi untuk perintah **for** ini adalah seperti berikut :

```

1 <?php
2 for ($nilai=1; $nilai <=10; $nilai++) {
3     if ($nilai % 2 == 0) {
4         echo " nilai urut ke-" . $nilai . "<br>" ;
5     }
6 }
7 ?>

```

Hasil keluaran dari kode di atas adalah seperti pada gambar 4.7 berikut :



Gambar 4.7: Hasil Keluaran Kode **for**

4.3.6 Perulangan **foreach**

Perintah **foreach** digunakan untuk melakukan iterasi pada sebuah *array* / larik. Struktur perintahnya secara sederhana dapat dituliskan sebagai berikut :

```

1 foreach (array as $nilai) {
2 ...
3 }

```

atau seperti ini :

```

1 foreach (array as $kunci => $nilai) {
2 ...
3 }

```

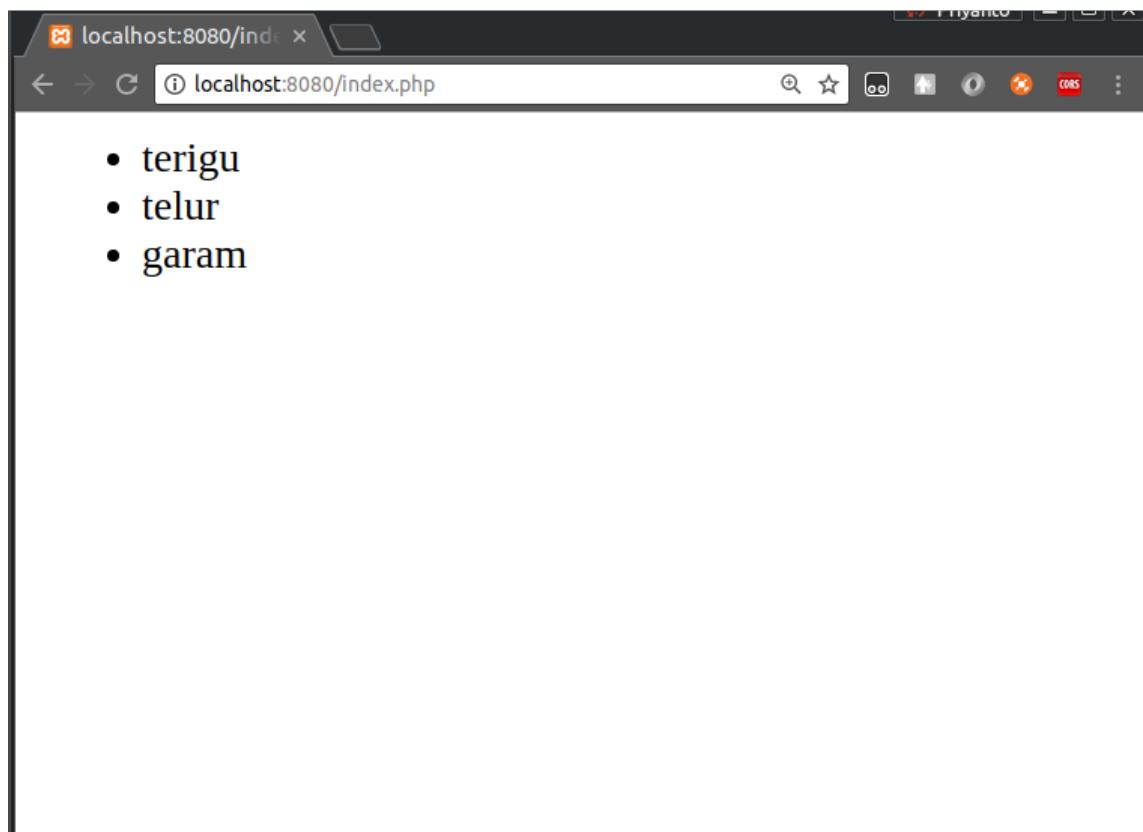
Nantinya tiap nilai yang ada pada *array* / larik akan dimasukkan kedalam variabel *\$nilai*, yang kemudian variabel ini dapat digunakan di dalam blok proses di bawahnya.

Struktur yang kedua sama saja, hanya saja isi dari *key* dari *array* akan dimasukkan ke dalam variabel *\$kunci*.

Mari kita coba implementasikan struktur perintah di atas ke dalam kode program seperti berikut :

```
1 <ul>
2 <?php
3 $data = array("terigu", "telur", "garam");
4
5 foreach($data as $nilai) {
6     echo "<li>$nilai</li>";
7 }
8 ?>
9 </ul>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 4.8 berikut :



Gambar 4.8: Hasil Keluaran `foreach` Tanpa *Key*

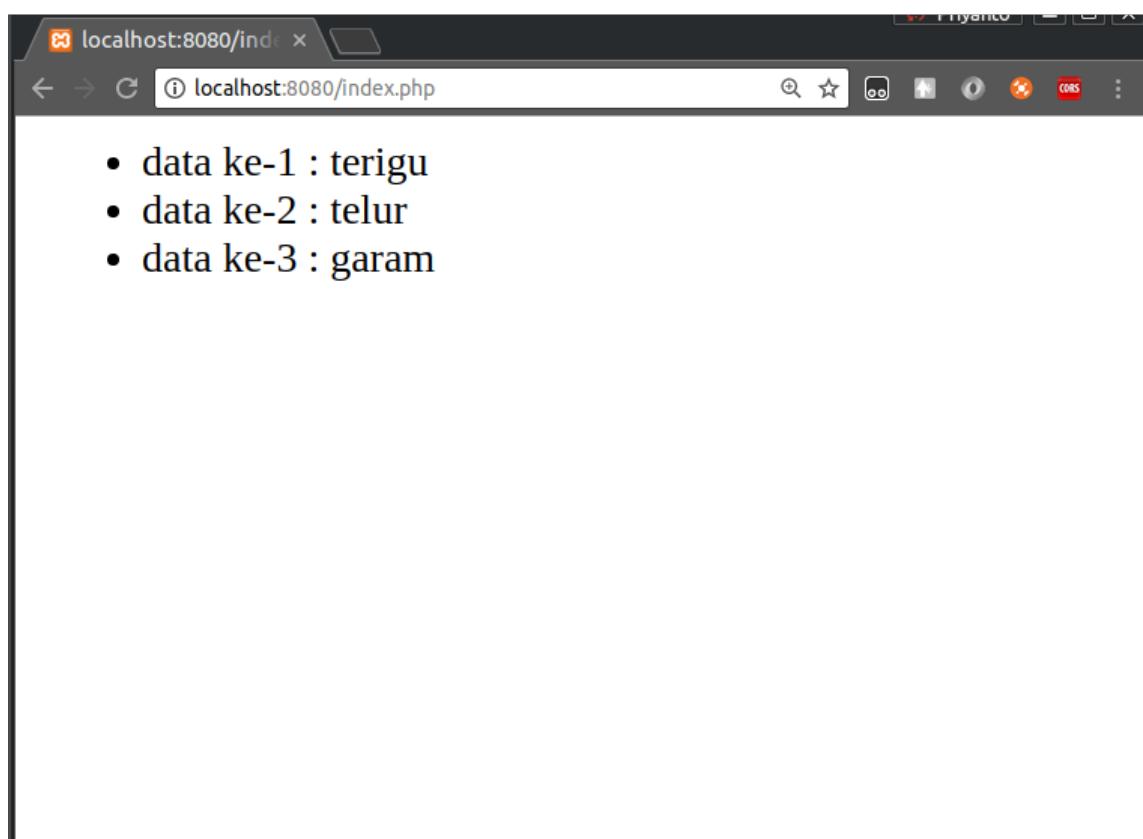
Apabila menggunakan *key* dari *array* maka kodennya dapat dibuat seperti berikut ini :

```

1 <ul>
2 <?php
3 $data = array("terigu", "telur", "garam");
4
5 foreach($data as $kunci => $nilai) {
6     echo "<li>data ke-". (++$kunci). " : $nilai</li>";
7 }
8 ?>
9 </ul>

```

Hasil keluaran dari kode di atas adalah seperti pada gambar 4.9 berikut :



Gambar 4.9: Hasil Keluaran `foreach` Dengan *Key*

4.3.7 Kontrol `break`

Perintah `break` digunakan untuk menghentikan eksekusi dari kontrol `switch`, `while`, `do-while`, `for`, dan `foreach`.

Perintah `break` ini dapat diberikan parameter berupa bilangan integer yang menandakan berapa tingkat kontrol yang akan dilakukan penghentian eksekusi.

Kita lihat terlebih dahulu bagaimana perintah `break` dapat menghentikan proses `switch` pada beberapa kode berikut :

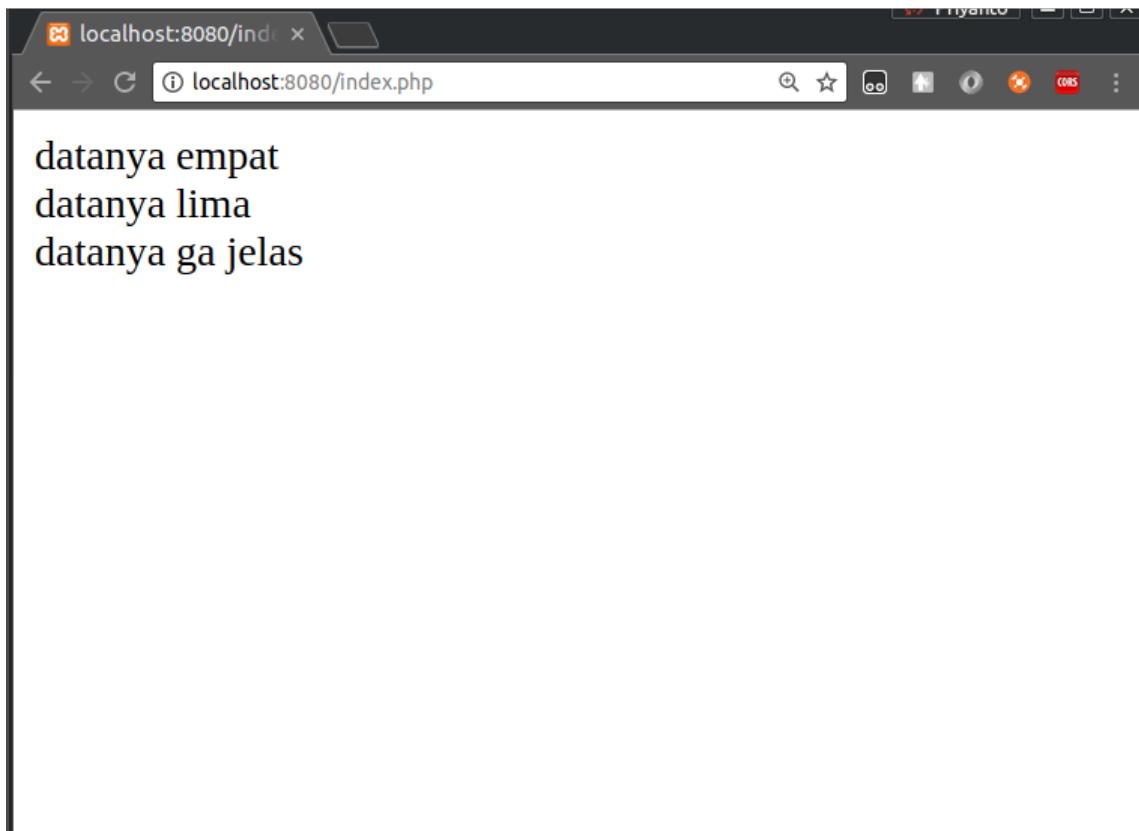
```

1 <?php
2 $data = 4;

```

```
3 switch($data) {  
4     case 4:  
5         echo "datanya empat<br>";  
6     case 5:  
7         echo "datanya lima<br>";  
8     default:  
9         echo "datanya ga jelas<br>";  
10    }  
11?  
12?>
```

Hasil keluaran untuk kode di atas adalah seperti pada gambar 4.10 berikut :



Gambar 4.10: Hasil Keluaran **switch** tanpa **break**

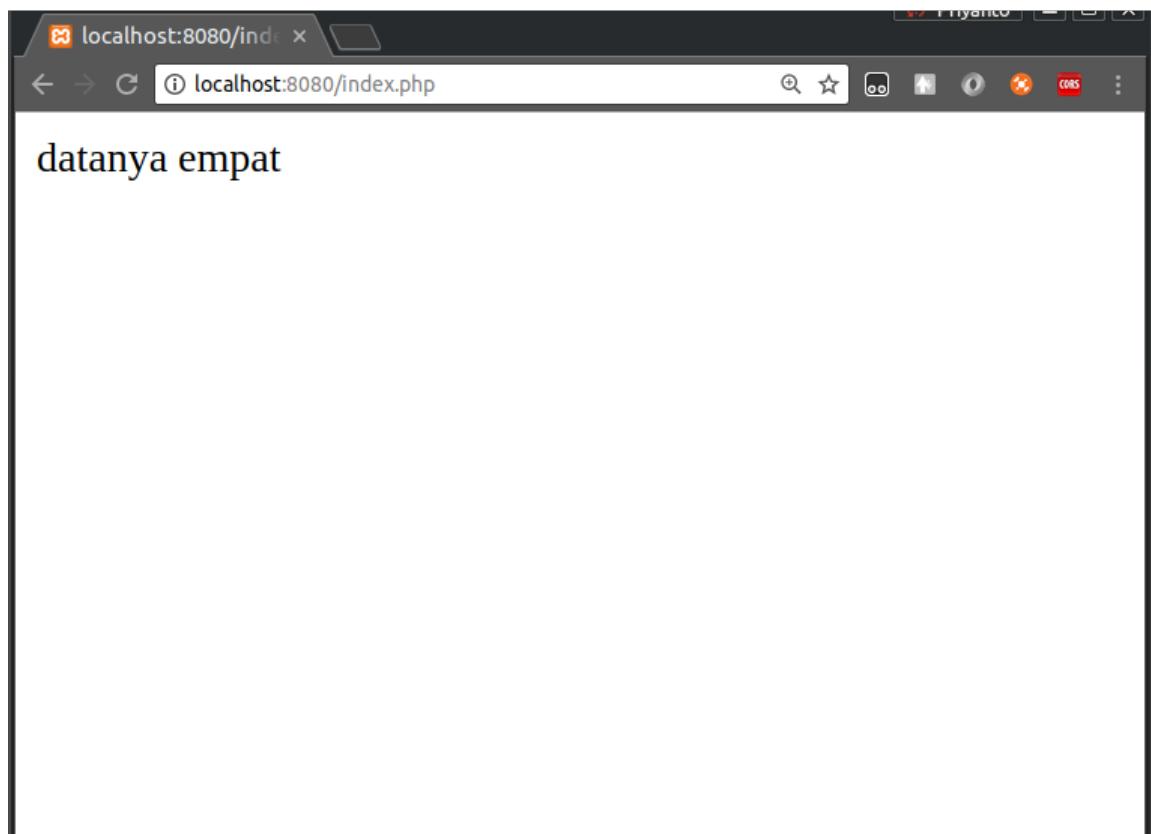
Kodenya akan terlihat lebih tepat apabila diberikan perintah **break** seperti berikut ini :

```
1 <?php  
2 $data = 4;  
3  
4 switch($data) {  
5     case 4:  
6         echo "datanya empat<br>";  
7         break;  
8     case 5:
```

```

9      echo "datanya lima<br>" ;
10     break;
11 default:
12     echo "datanya ga jelas<br>" ;
13     break;
14 }
15 ?>
```

Hasil keluaran kode di atas adalah seperti pada gambar 4.11 berikut ini :



Gambar 4.11: Hasil Keluaran **switch** dengan **break**

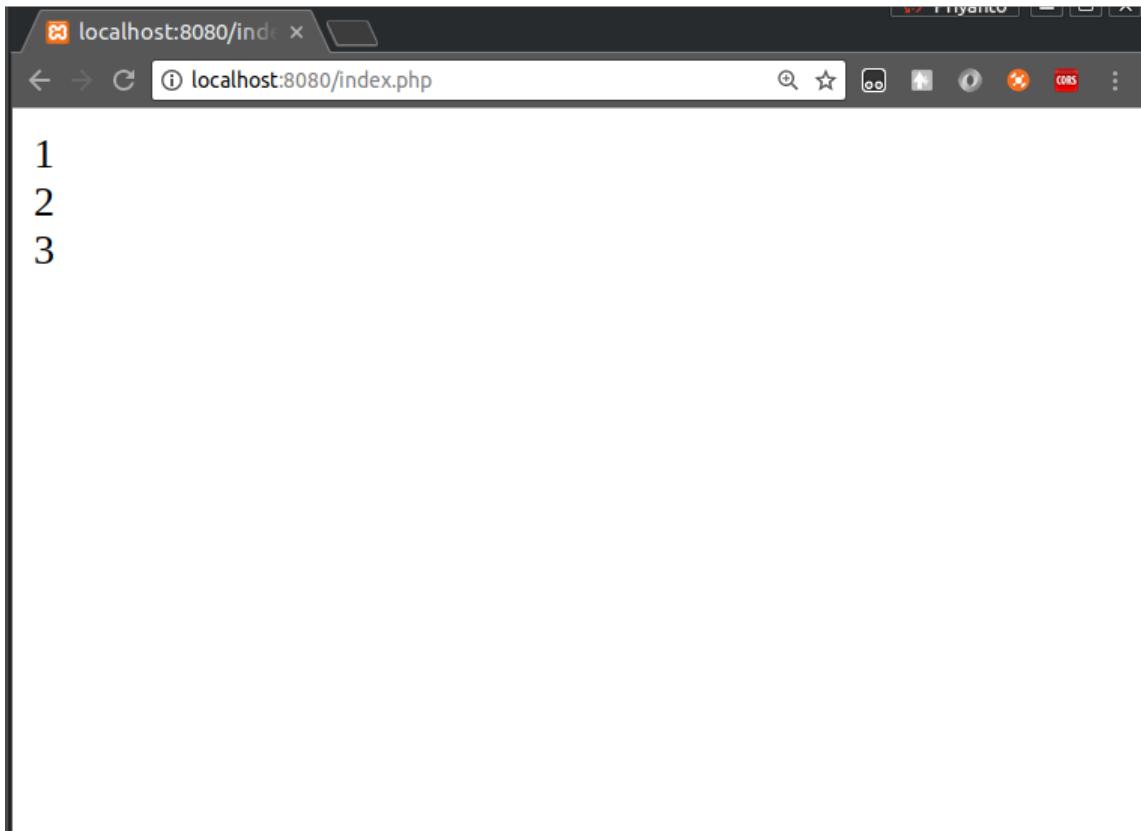
Ketika menggunakan **break** maka hanya **case** yang terseleksi saja yang akan dijalankan, sedangkan **case** berikutnya tidak akan dijalankan.

Kontrol **break** yang digunakan pada struktur **while** akan menghentikan proses iterasi yang sedang berjalan, contoh penggunaan kodenya adalah sebagai berikut :

```

1 <?php
2 $data = 4;
3 $counter = 1;
4
5 while($counter < 10) {
6     echo $counter++."<br>";
7     if($data == $counter) break;
8 }
9 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 4.12 berikut :



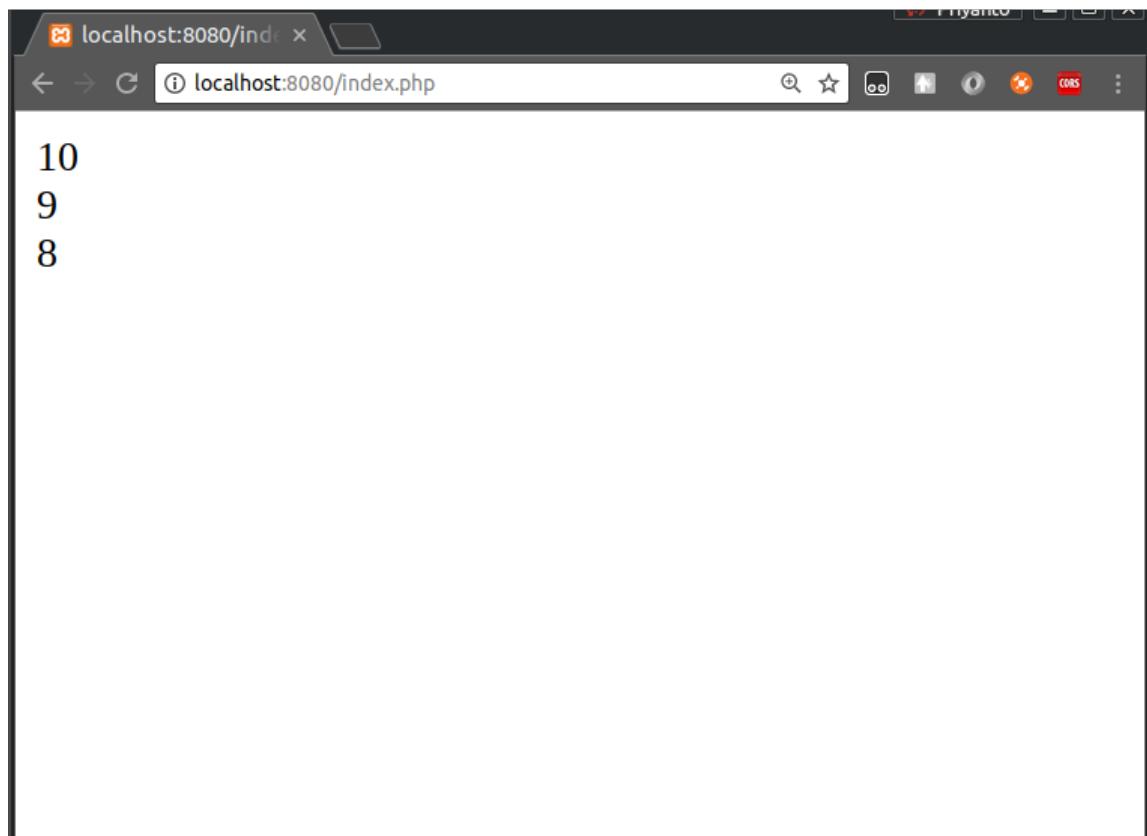
Gambar 4.12: Hasil Keluaran Penggunaan `break` pada `while`

Pada hasil keluaran, angka yang muncul hanya sampai dengan 3 (tiga), karena begitu nilai variabel `$counter` sama dengan `$data` (yang bernilai 4 (empat) maka iterasi `while` dipaksa untuk menghentikan proses iterasinya dengan `break`.

Proses penghentian iterasi pada `do-while` pun akan sama, mari kita perhatikan contoh kode program berikut :

```
1 <?php
2 $data = 7;
3 $counter = 10;
4
5 do {
6     echo $counter--."<br>";
7     if ($data == $counter) break;
8 } while($counter > 0);
9 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 4.13 berikut :

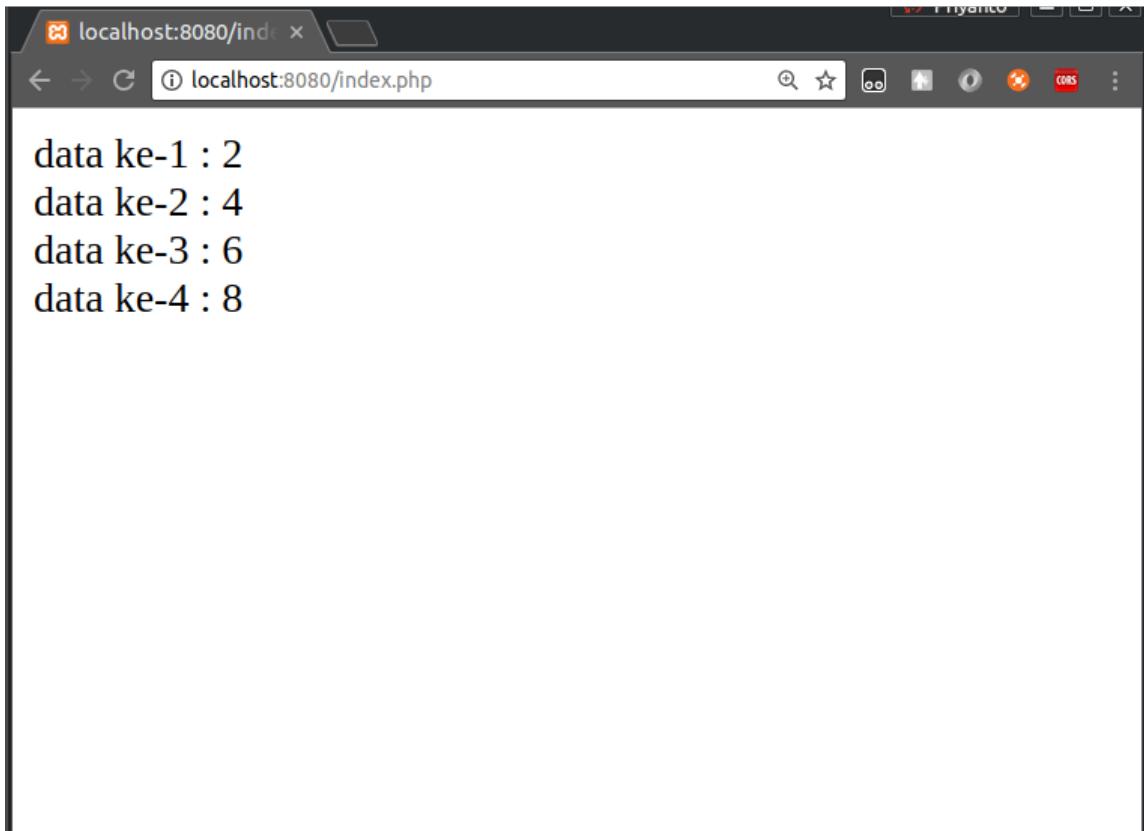


Gambar 4.13: Hasil Keluaran Kode `do-while` Dengan `break`

Untuk penggunaan perintah `break` pada iterasi `for` sama saja, mari kita perhatikan contoh kode program berikut :

```
1 <?php
2 for($counter=1; $counter < 100; $counter++) {
3     echo "data ke-". $counter . " : " . ($counter * 2) . "<br>";
4     if($counter == 4) break;
5 }
6 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 4.14 berikut :



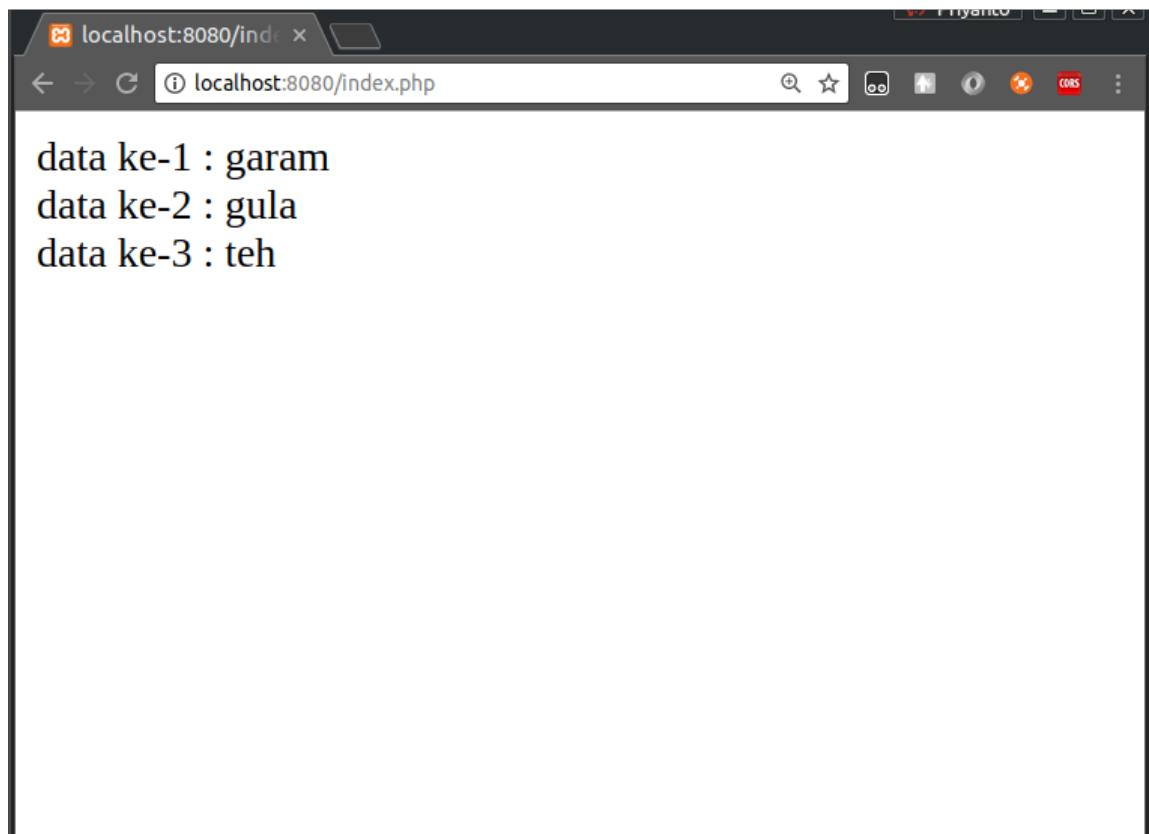
Gambar 4.14: Hasil Keluaran Penggunaan **break** pada **for**

Terlihat bahwa iterasi yang dilakukan hanya sampai iterasi ke-4, karena begitu nilai pada variabel **\$counter** bernilai 4 (empat) maka proses iterasi dihentikan oleh perintah **break**.

Untuk contoh kode **break** yang menghentikan struktur iterasi **foreach** dapat kita lihat pada baris kode berikut :

```
1 <?php
2 $data = array("garam" , "gula" , "teh" , "kopi");
3 $counter = 1;
4
5 foreach($data as $nilai) {
6     echo "data ke-". $counter++ . " : " . $nilai . "<br>";
7     if( $nilai == "teh" ) break;
8 }
9 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 4.15 berikut :



Gambar 4.15: Hasil Keluaran Penggunaan **break** Pada **foreach**

Pada *array* yang ada di kode, ada 4 (empat) komponen disana, tetapi yang dicetak hanya 3, karena pada iterasi yang mengeluarkan data **teh**, iterasi dihentikan dengan **break**.

4.3.8 Kontrol continue

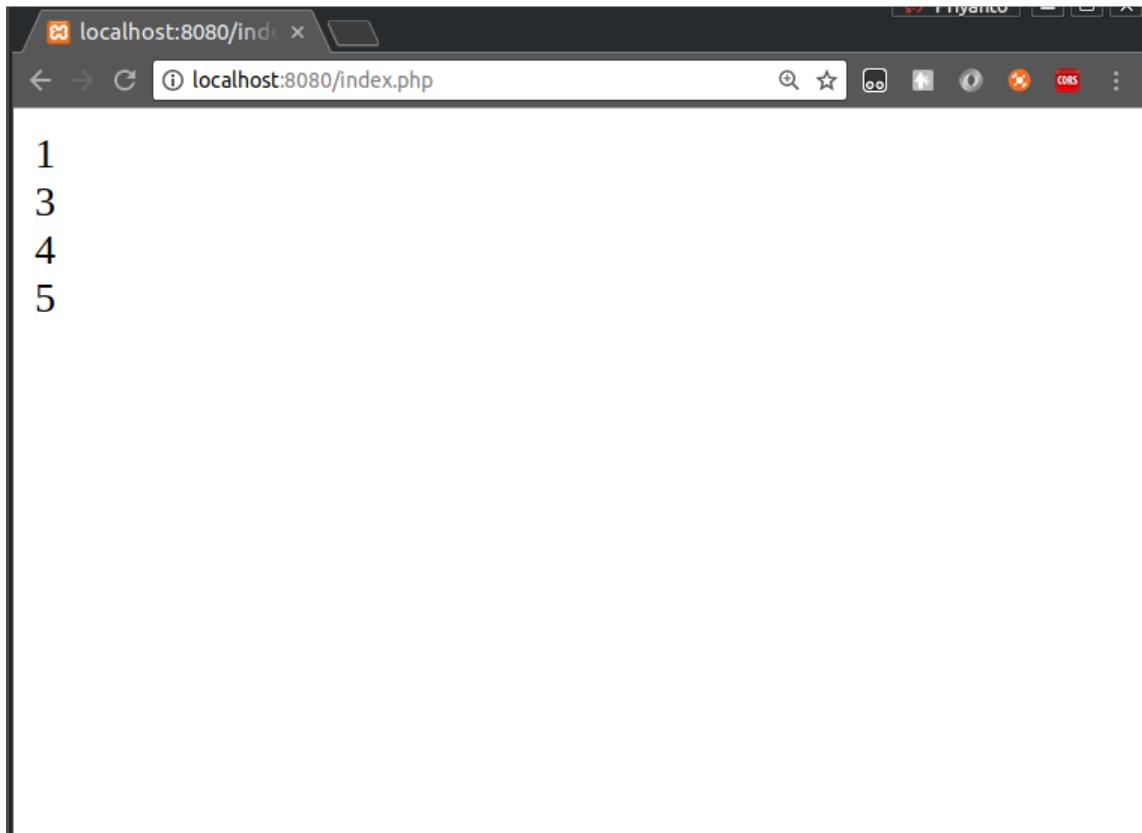
Penggunaan **continue** mirip dengan **break**, hanya saja tujuannya berbeda, pada penggunaan kontrol **continue** proses iterasi akan dilewatkan sekali ke proses iterasi berikutnya.

Mari kita perhatikan kode berikut ini :

```

1 <?php
2 for($counter=1; $counter < 6; $counter++) {
3     if($counter == 2) continue;
4     echo $counter."<br>";
5 }
6 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 4.16 berikut :



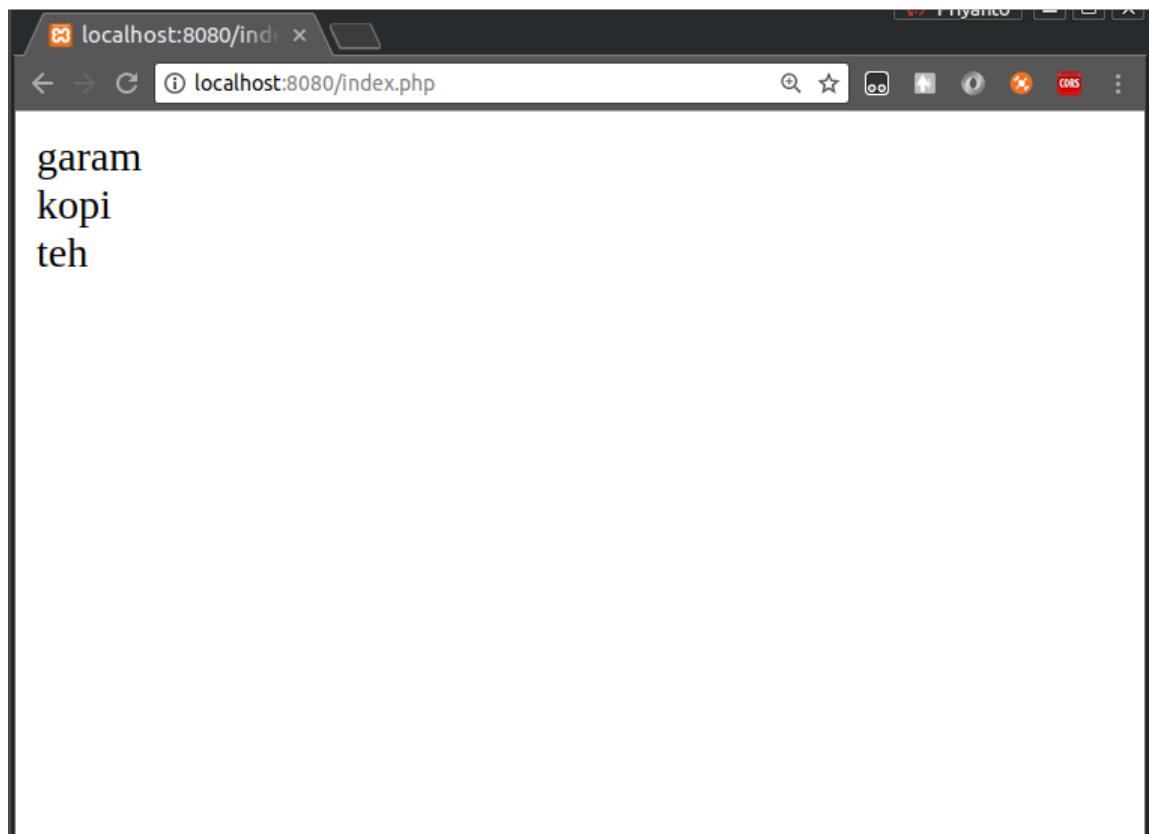
Gambar 4.16: Hasil Keluaran Penggunaan `continue` Pada `for`

Dari hasil keluaran di atas, angka 2 (dua) tidak dicetak ke *browser* karena pada saat nilai dari variabel `$counter` bernilai 2 (dua), ada perintah `continue` yang melewatkannya proses iterasi yang sedang berjalan ke proses iterasi berikutnya.

Untuk penggunaan pada `foreach`, mari kita lihat contoh kode berikut :

```
1 <?php
2 $data = array("garam" , "gula" , "kopi" , "teh");
3
4 foreach($data as $nilai) {
5     if($nilai == "gula") continue;
6     echo $nilai."<br>";
7 }
8 ?>
```

Hasil keluaran dari kode tersebut adalah seperti pada gambar 4.17 berikut :



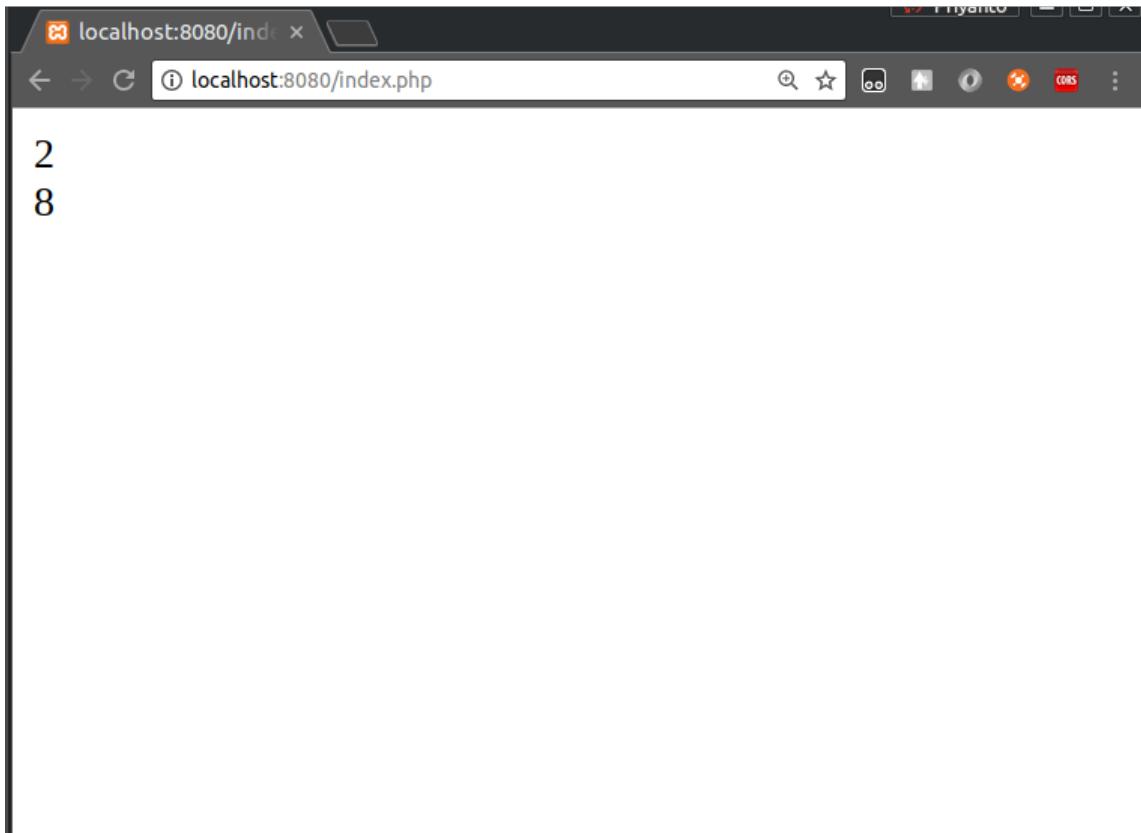
Gambar 4.17: Hasil Keluaran Penggunaan `continue` Pada `foreach`

Pada hasil keluaran tampak bahwa `gula` tidak ikut ditampilkan di *browser* karena dalam iterasinya, begitu ditemukan nilai berupa `gula` maka proses akan dilewati ke iterasi berikutnya.

Contoh penggunaan `continue` pada blok iterasi `while` adalah seperti berikut ini :

```
1 <?php
2 $counter = 1;
3 while(true) {
4     $counter *= 2;
5     if($counter == 4) continue;
6     if($counter > 15) break;
7     echo $counter."<br>";
8 }
9 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 4.18 berikut :



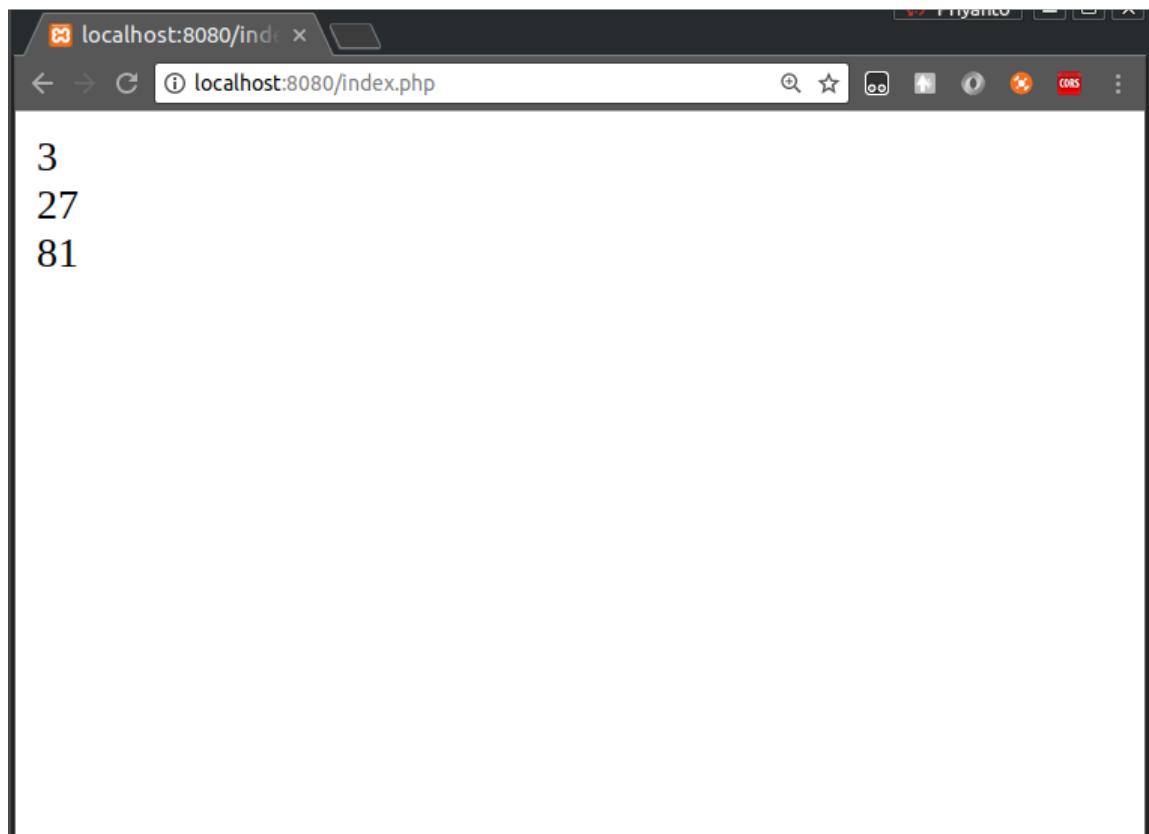
Gambar 4.18: Hasil Keluaran Penggunaan `continue` Pada `while`

Dari deretan angka di atas, angka 4 (empat) akan dilewatkan karena begitu nilai dari variabel `$counter` bernilai 4 (empat) diberikan perintah `continue` untuk melewatkannya proses dibawahnya sehingga masuk ke proses iterasi berikutnya.

Contoh penggunaan `continue` pada blok iterasi `do-while` adalah sebagai berikut :

```
1 <?php
2 $counter = 1;
3 $ok = true;
4
5 do {
6     $counter *= 3;
7     if ($counter == 9) continue;
8     $ok = ($counter > 30) ? false : true;
9     echo $counter."<br>";
10 } while($ok);
11 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 4.19 berikut :



Gambar 4.19: Hasil Keluaran Penggunaan `continue` Pada `do-while`

Hasil keluaran di atas akan melewatkkan proses pencetakan angka 9 (sembilan) dan dilanjutkan ke iterasi berikutnya.

4.3.9 Kontrol include dan require

Perintah `include` digunakan untuk memasukan dan mengevaluasi *file* yang ada pada parameter `include` yang isinya dapat digunakan pada *file* saat ini.

Apabila *file* yang disertakan pada parameter tidak ditemukan, maka perintah ini akan menghasilkan *warning*. Mari kita perhatikan kode berikut :

```

1 <?php
2 echo "Hai, ". $nama . "<br><br>";
3
4 include "data.php";
5 echo $nama . " baru keluar";
6 ?>

```

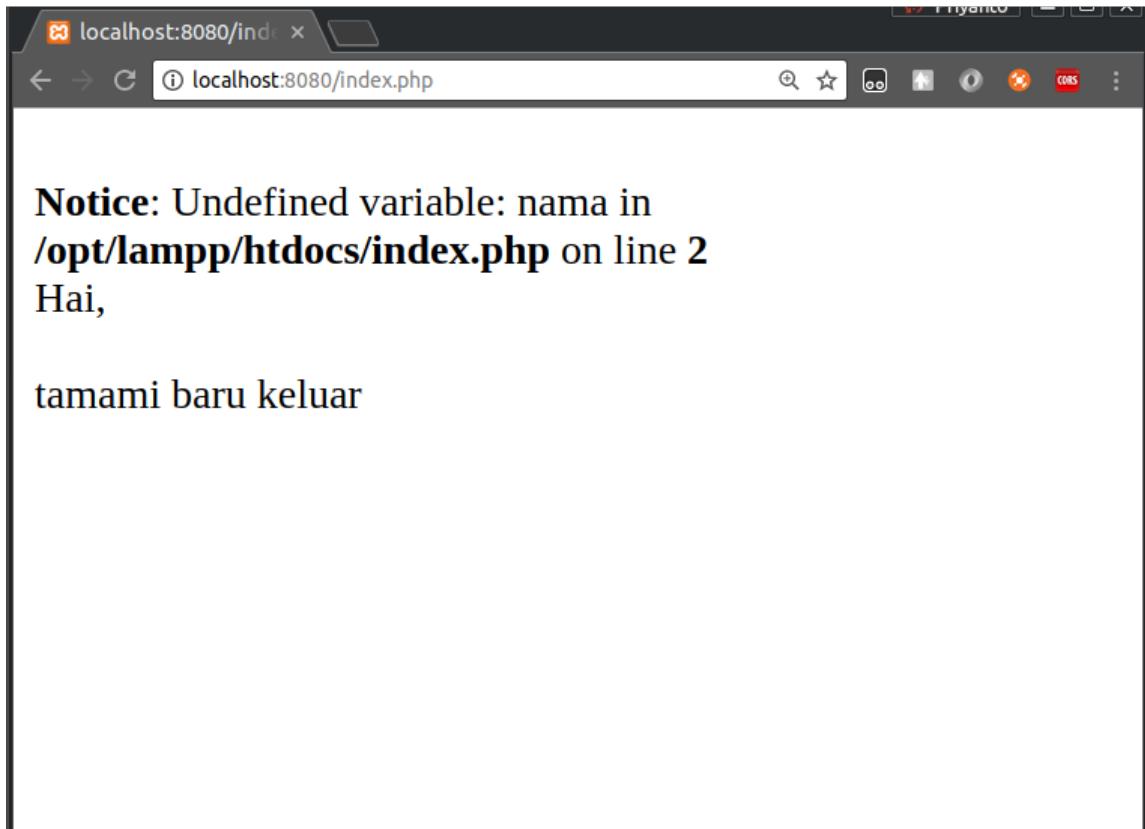
Isi dari kode di atas kita simpan dalam *file* dengan nama `index.php`, selanjutnya kita buat *file* baru dengan nama `data.php` yang disimpan dalam satu *folder* yang sama dengan `index.php`, isi kode dari *file* `data.php` adalah sebagai berikut :

```

1 <?php
2     $nama = "tamami";
3 ?>

```

Hasil keluaran saat kita melakukan akses ke `localhost/index.php` adalah seperti pada gambar 4.20 berikut :



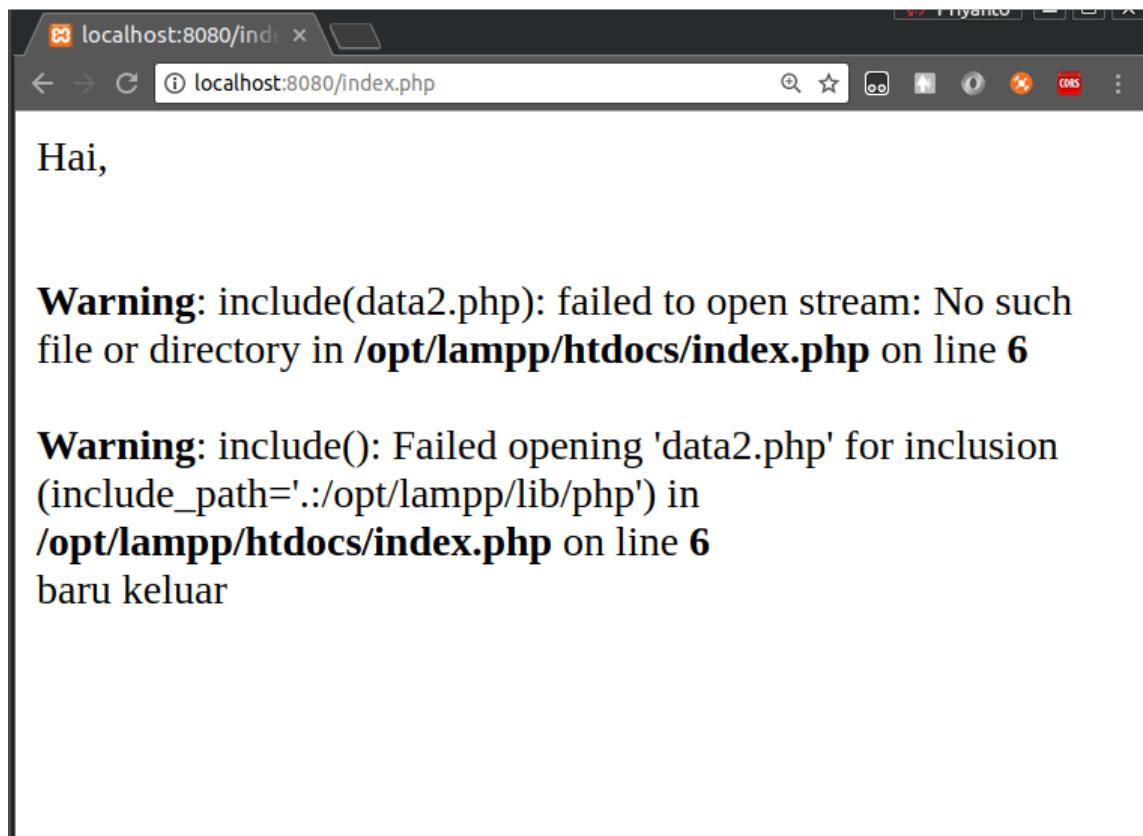
Gambar 4.20: Hasil Keluaran Contoh Penggunaan `include`

Pada gambar 4.20, pemanggilan variabel `$nama` pada baris ke-2 mengalami kegalan, karena perintah `include` baru dijalankan pada baris berikutnya, dan pada baris ke-5 variabel `$nama` berhasil dipanggil.

Sekarang coba kita ubah kode pada file `index.php` menjadi seperti berikut ini :

```
1 <?php
2 $nama = "";
3
4 echo "Hai, " . $nama . "<br><br>";
5
6 include "data2.php";
7 echo $nama . " baru keluar";
8 ?>
```

Hasil keluaran untuk kode di atas adalah seperti pada gambar 4.21 berikut :



Gambar 4.21: Muncul *Warning* Saat *File* Tidak Ditemukan

Pada gambar 4.21 akan keluar *warning* bahwa *file* data2.php tidak ditemukan, tetapi proses tetap dijalankan.

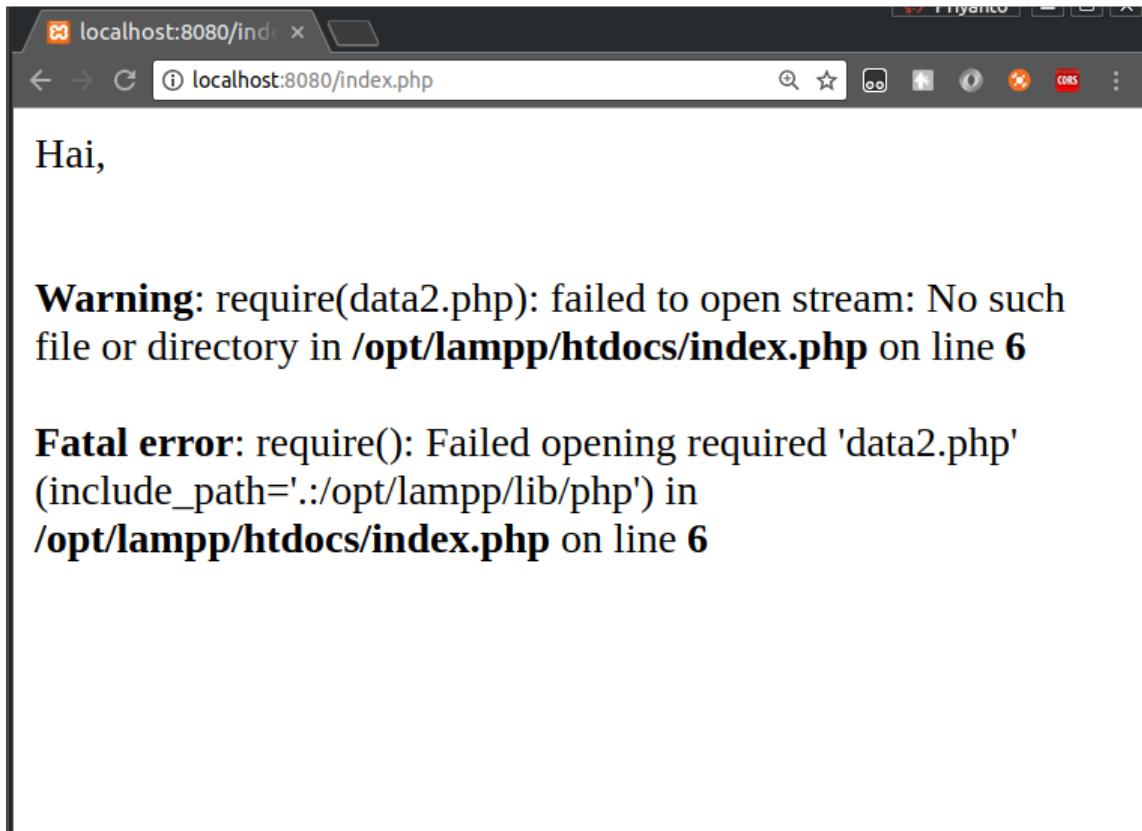
Disinilah letak bedanya antara penggunaan **include** dengan **require**, fungsi keduanya sama-sama akan memasukkan data dari *file* lain, namun bedanya, saat **require** tidak menemukan *file* yang dicari, maka perintah **require** akan mengembalikan pesan **error** dan menghentikan prosesnya. Perhatikan kode dari *file* index.php berikut :

```

1 <?php
2 $nama = "";
3
4 echo "Hai , " . $nama . "<br><br>" ;
5
6 require "data2.php";
7 echo $nama . " baru keluar";
8 ?>

```

Hasil keluaran untuk kode tersebut adalah seperti pada gambar 4.22 berikut :



Gambar 4.22: Hasil Keluaran Kesalahan Saat Menggunakan `require`

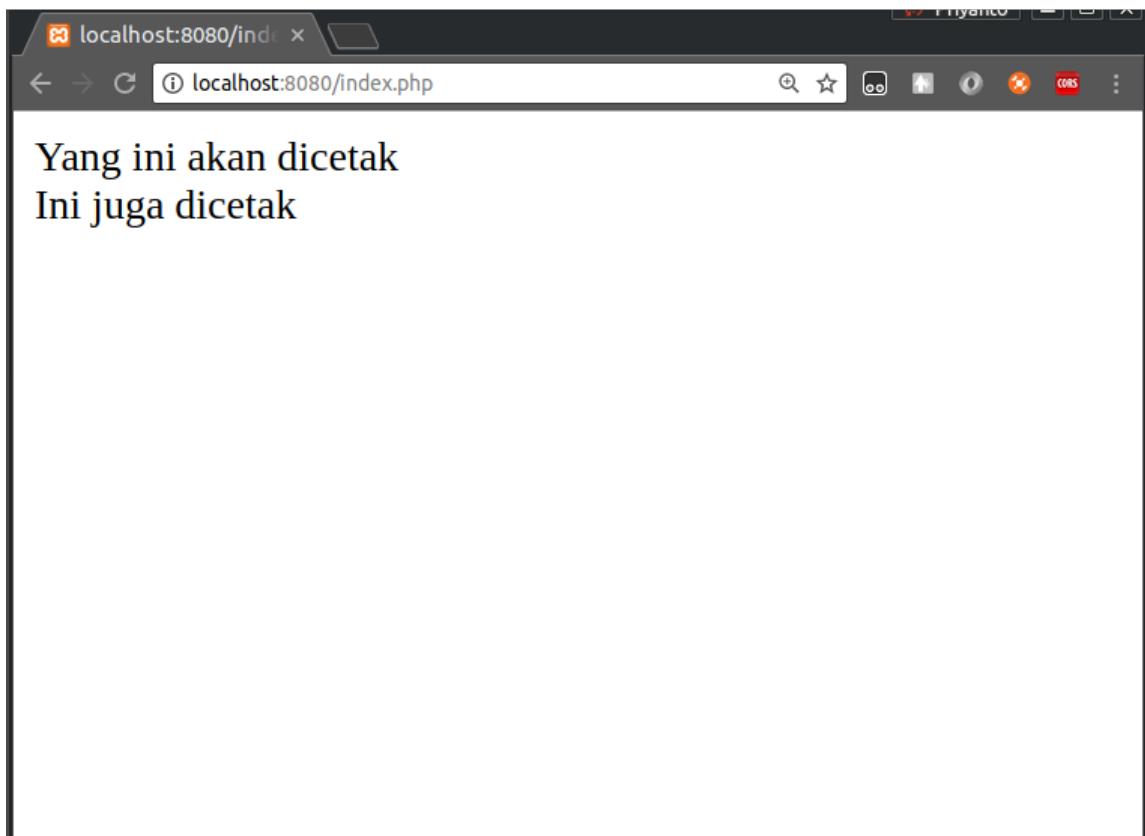
Perhatikan bahwa di akhir baris pada *browser* tidak mengeluarkan teks baru **keluar** seperti pada kode sebelumnya, ini karena perintah `require` tidak menemukan *file data2.php* dan langsung menghentikan proses dengan mengeluarkan pesan **fatal error**.

4.3.10 Kontrol goto

Perintah `goto` digunakan untuk melompat ke nama label tertentu. Mari kita lihat contoh kodennya berikut :

```
1 <?php
2 goto cetakDonk;
3 echo "Teks ini ga akan dicetak<br>";
4 echo "teks ini pun akan terlewat<br>";
5
6 cetakDonk:
7 echo "Yang ini akan dicetak<br>";
8 echo "Ini juga dicetak";
9 ?>
```

Hasil keluaran untuk kode di atas adalah seperti pada gambar 4.23 berikut :



Gambar 4.23: Hasil Keluaran Menggunakan Perintah `goto`

Perhatikan bahwa perintah pada baris ke-3 dan ke-4 tidak pernah dikerjakan karena kodenya akan dilewatkan ke baris 6 dan selanjutnya.

4.4 Kesimpulan

Dengan menggunakan beberapa perintah kontrol, kita dapat mengatur alur kode program apakah akan berjalan bercabang, berjalan berulang dengan iterasi, atau akan melakukan lompatan kode ke label tertentu.

4.5 Tugas

Langkah dari tugas kali ini adalah sebagai berikut :

1. Buatlah sebuah variabel `$username` lalu isikan namanya bebas.
2. Buatlah sebuah variabel `$daftar` yang berisi daftar 5 (lima) `username` dalam *array*.

3. lakukan iterasi terhadap variabel `$daftar` kemudian cocokan isinya satu per satu dengan variabel `$username`, bila cocok tampilkan pesan `Selamat Datang`, bila tidak ada yang cocok maka tampilkan pesan `Anda belum terdaftar`.

Bab 5

Fungsi, Modularisasi, dan Pemrosesan *File*

5.1 Tujuan

Mahasiswa mampu mengimplementasikan penggunaan fungsi, membagi sebuah aplikasi dalam beberapa modul, dan melakukan pemrosesan *file* dengan PHP.

5.2 Pengantar

Fungsi sebetulnya adalah sebuah blok perintah yang dikumpulkan menjadi satu dan diberikan nama agar dapat digunakan di beberapa tempat. Setiap fungsi yang dideklarasikan tidak akan dieksekusi atau di jalankan kecuali ada pemanggilan terhadap fungsi tersebut.

Apabila sebuah aplikasi semakin besar, maka diperlukan lebih banyak fungsi. Untuk memudahkan pemeliharaan dan pengembangan, biasanya aplikasi akan dipecah menjadi beberapa modul atau kelas atau objek. Di dalam masing-masing objek nantinya akan memiliki beberapa properti dan fungsi yang masing-masing akan melakukan beberapa tugas yang berkaitan dengan objek yang dibangun.

Untuk pemrosesan *file* sendiri fungsinya adalah bagaimana caranya kita akan menuliskan sebuah data ke *file* dan membaca isi dari sebuah *file* yang datanya dapat kita gunakan pada program yang kita bangun.

5.3 Praktek

5.3.1 Fungsi

Deklarasi sederhana dari fungsi adalah seperti berikut ini :

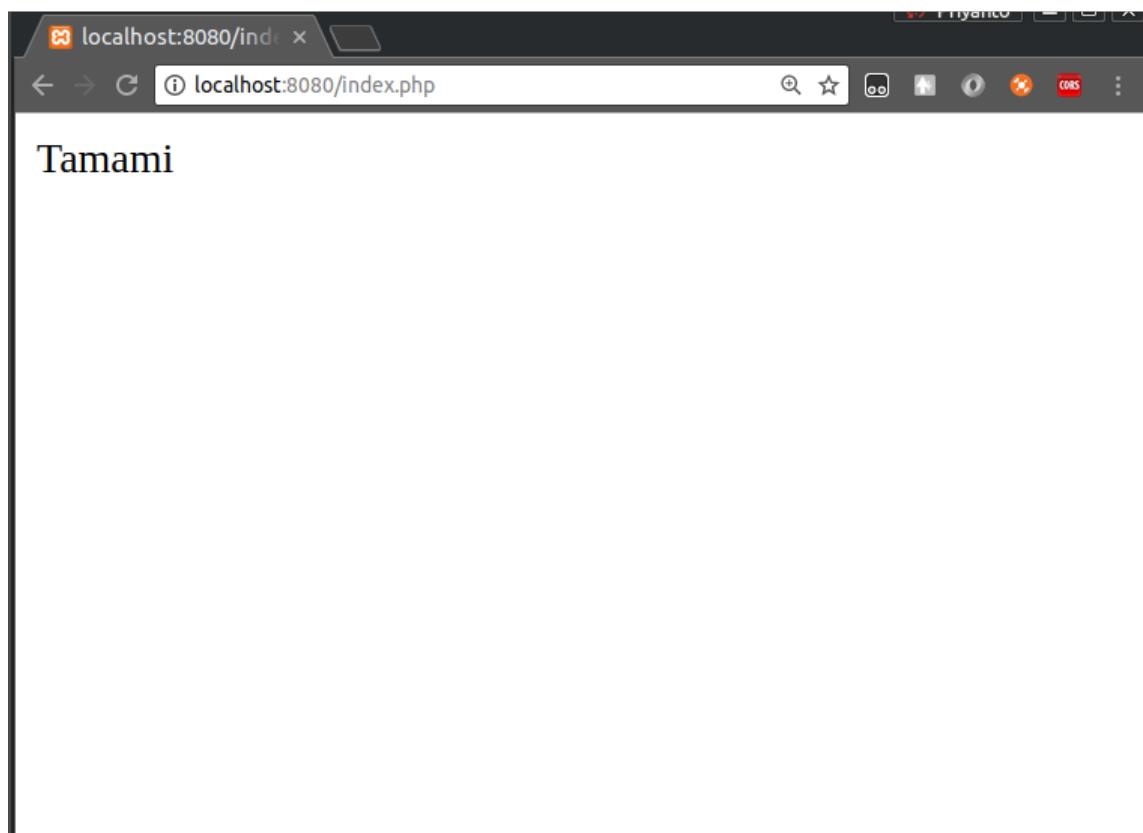
```
1 function namaFungsi() {  
2     ...  
3 }
```

Implementasi sederhananya yaitu, kita akan membuat sebuah fungsi untuk melakukan pencetakan sebuah nama, fungsi ini tanpa argumen atau parameter, berikut kodennya :

```
1 <?php
2 function cetakNama() {
3     echo "Tamami";
4 }
5
6 cetakNama();
7 ?>
```

Perhatikan bahwa pada baris ke-6, fungsi `cetakNama()` dipanggil, bila pada baris ke-6 kita hapus, maka pada *browser* tidak akan tampil apa-apa, karena fungsi tersebut hanya dideklarasikan tetapi tidak pernah dieksekusi.

Hasil keluaran dari kode di atas akan tampak sederhana sekali, seperti ditunjukkan pada gambar 5.1 berikut :



Gambar 5.1: Hasil Keluaran Fungsi `cetakNama` Sederhana

Dalam deklarasi fungsi, sebetulnya dapat menerima nilai dari luar fungsi untuk digunakan didalam fungsi dengan menggunakan argumen / parameter, contoh kode yang akan dibuat berikutnya adalah mencoba menerima sebuah parameter angka, kemudian angka tersebut akan dipangkatkan 2 (dua) lalu dicetak hasilnya, berikut kodennya :

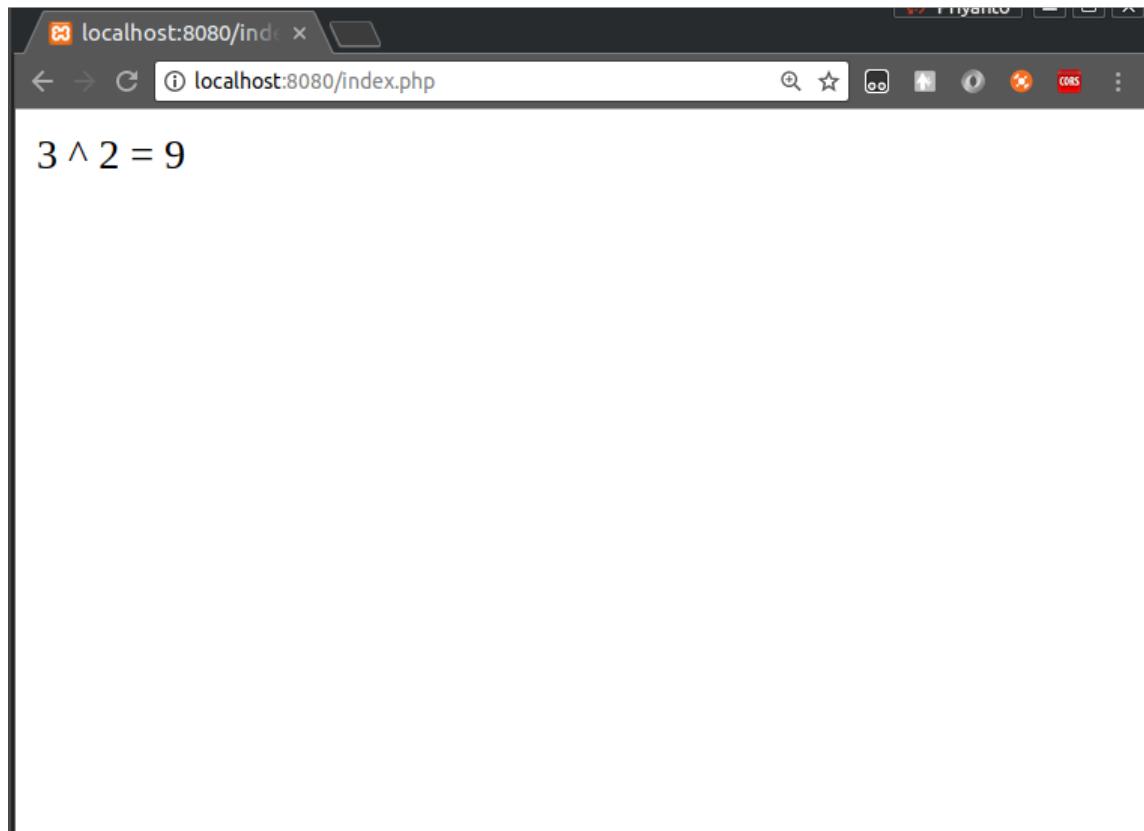
```

1 <?php
2 function pangkat($angka) {
3     echo $angka . " ^ 2 = ". ($angka ** 2);
4 }
5
6 pangkat(3);
7 ?>

```

Fungsi `pangkat` dideklarasikan untuk menerima 1 (satu) parameter `$angka` dan dipanggil dengan memberikan sebuah parameter pada baris ke-6, sehingga parameter (yang bernilai 3) tersebut dioleh dengan operator pangkat seperti pada baris ke-3.

Hasil keluaran dari kode tersebut adalah seperti pada gambar 5.2 berikut :



Gambar 5.2: Hasil Keluaran Fungsi `pangkat` Dengan Sebuah Parameter

Sebuah fungsi sebetulnya dapat mengembalikan sebuah nilai yang dapat digunakan oleh *flow* di luar fungsi. Sebagai contoh, perhatikan kode berikut ini :

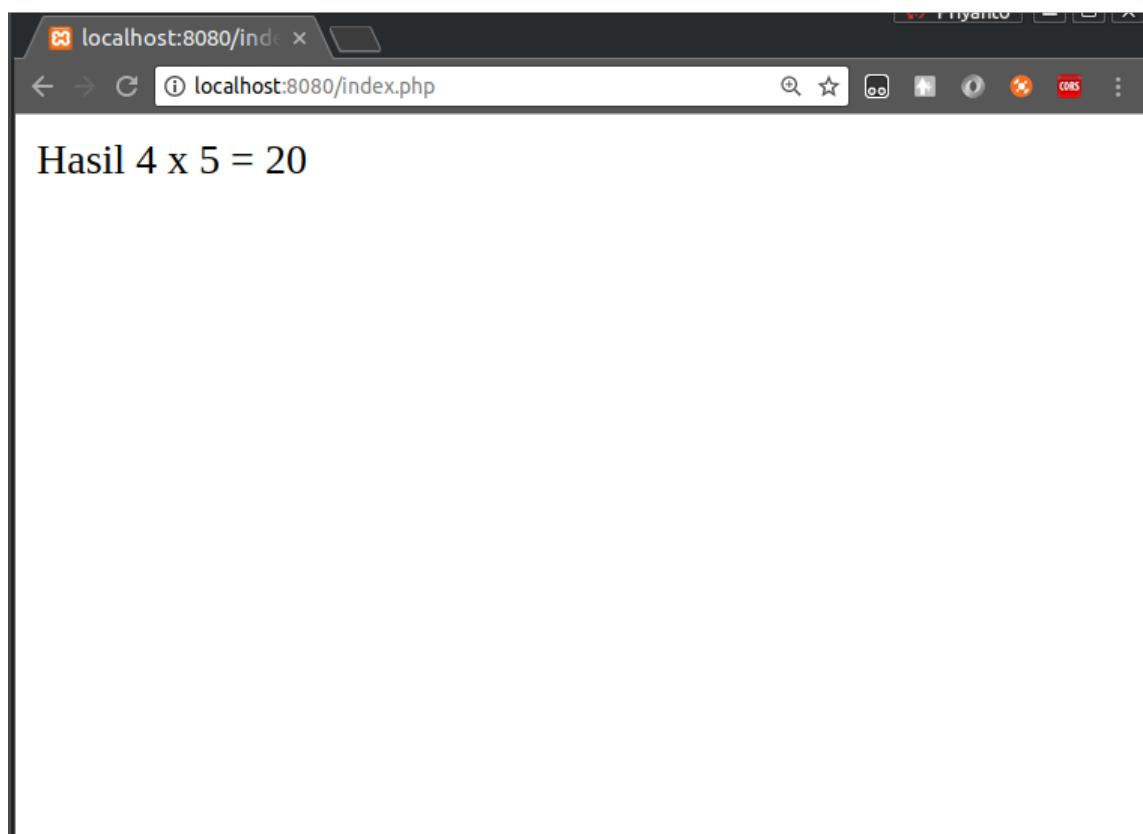
```

1 <?php
2 function kali($angka1, $angka2) {
3     return ($angka1 * $angka2);
4 }
5
6 $data1 = 4;
7 $data2 = 5;
8 echo "Hasil " . $data1 . " x " . $data2 . " = " . kali($data1, $data2);

```

9 ?>

Hasil keluaran dari kode di atas adalah seperti pada gambar 5.3 berikut :



Gambar 5.3: Hasil Keluaran Fungsi Yang Memiliki Nilai Balikan

5.3.2 Kelas dan Objek

Kelas dan Objek sebetulnya digunakan untuk mempermudah baik dalam pengembangan sebuah aplikasi maupun dalam hal pemeliharaannya. Dalam kelas sendiri nantinya akan terdiri dari satu atau lebih properti (yang biasa dikenal dengan sebutan variabel), dan satu atau lebih *method* (yang biasa kita deklarasikan sebagai fungsi).

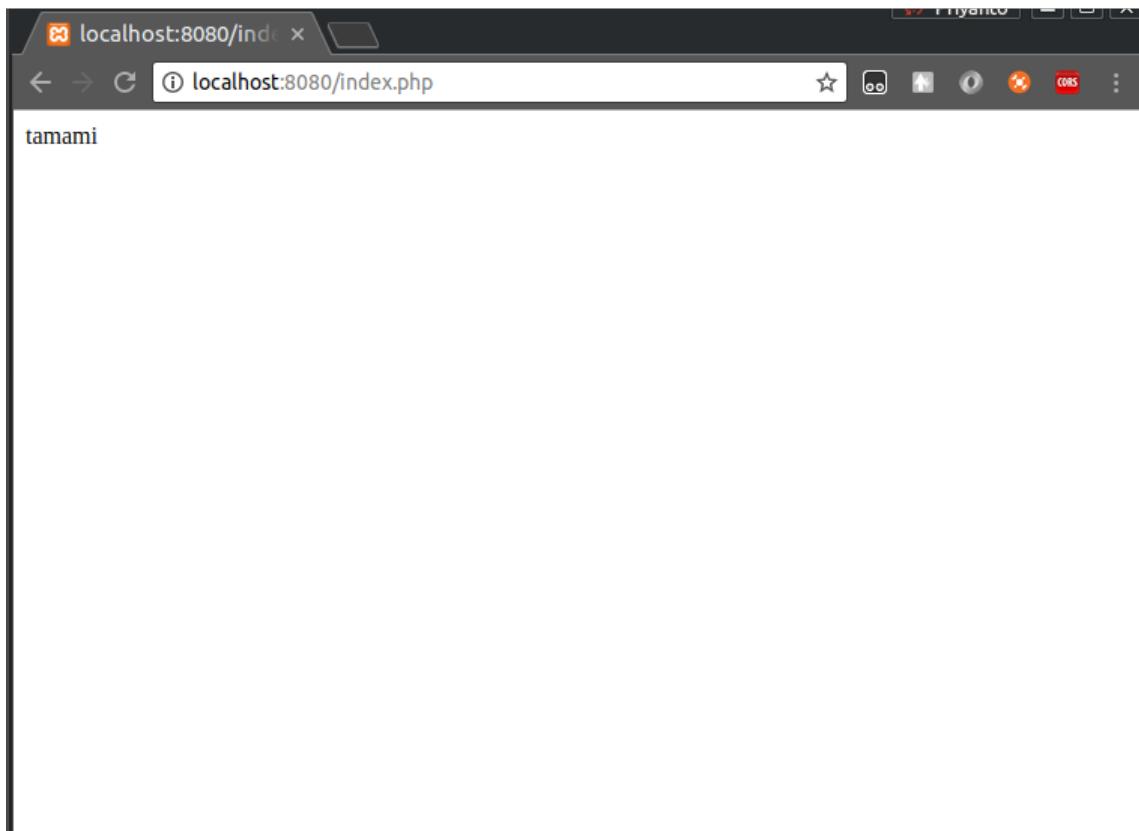
```

1 <?php
2 class Mahasiswa {
3     public $nama = "tamami";
4
5     public function getNama() {
6         return $this->nama;
7     }
8 }
9
10 $mhs = new Mahasiswa();
11 echo $mhs->getNama();
```

12 ?>

Pada kode di atas, kita membuat sebuah kelas dengan nama `Mahasiswa`, yang didalamnya terdapat sebuah properti `$nama` (yang diberi nilai `tamami`) dan sebuah *method* `getNama` (yang mengembalikan sebuah nilai dari variabel `$nama` milik kelas `Mahasiswa`).

Implementasi pembentukan objek dari kelas `Mahasiswa` baru dilakukan pada baris ke-10 dengan perintah `new`, setelah itu pemanggilan fungsi `getNama` dilakukan di baris ke-11 yang hasilnya ditampilkan di *browser*. Hasil tampilan dari kode di atas adalah seperti pada gambar 5.4 berikut :



Gambar 5.4: Hasil Keluaran Contoh Sederhana Implementasi Kelas

Sebuah kelas juga dapat diwariskan ke kelas yang baru, nantinya kelas baru akan memiliki properti dan *method* yang sama dengan kelas pendahulunya, mari perhatikan contoh kode berikut :

```
1 <?php
2 class Barang {
3     public function cetakKodeBarang() {
4         echo "Kode Barangnya : 12354";
5     }
6 }
7
```

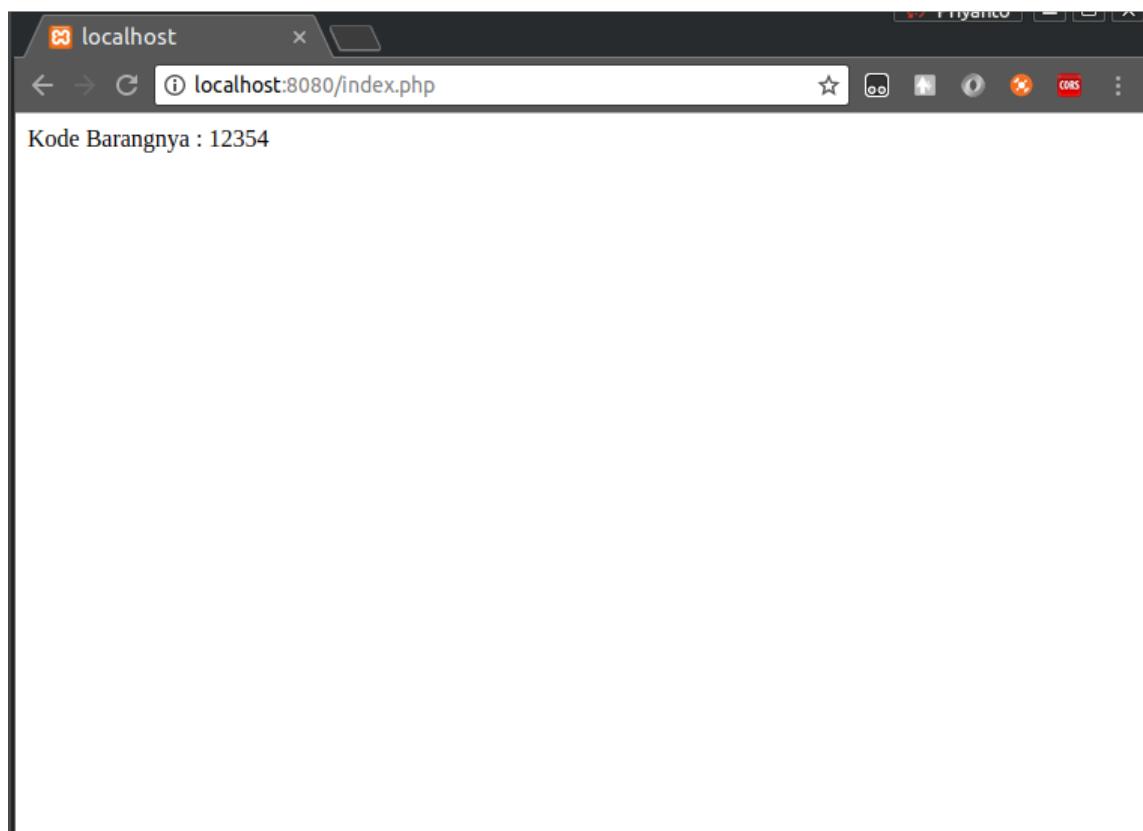
```

8 class Coklat extends Barang {}
9
10 $data = new Coklat();
11 $data->cetakKodeBarang();
12 ?>

```

Kelas **Coklat** di atas akan mewarisi kelas **Barang** dimana pada kelas **Barang** memiliki sebuah *method* bernama **cetakKodeBarang()**, karena kelas **Coklat** mewarisi kelas **Barang**, maka fungsi / *method* milik kelas **Barang** pun akan diwariskan ke kelas **Coklat**, sehingga *method* **cetakKodeBarang()** dapat dipanggil dari kelas **Coklat**.

Hasil keluaran dari kode program di atas adalah seperti pada gambar 5.5 berikut :



Gambar 5.5: Hasil Keluaran Contoh Sederhana Pewarisan

Pada kelas dapat memiliki deklarasi konstanta yang tujuannya adalah memberikan nilai atau data acuan untuk aplikasi, misalkan, kita akan mendeklarasikan nilai atau data acuan untuk server basis data yang digunakan pada alamat phb.ac.id, perhatikan contoh kode berikut ini :

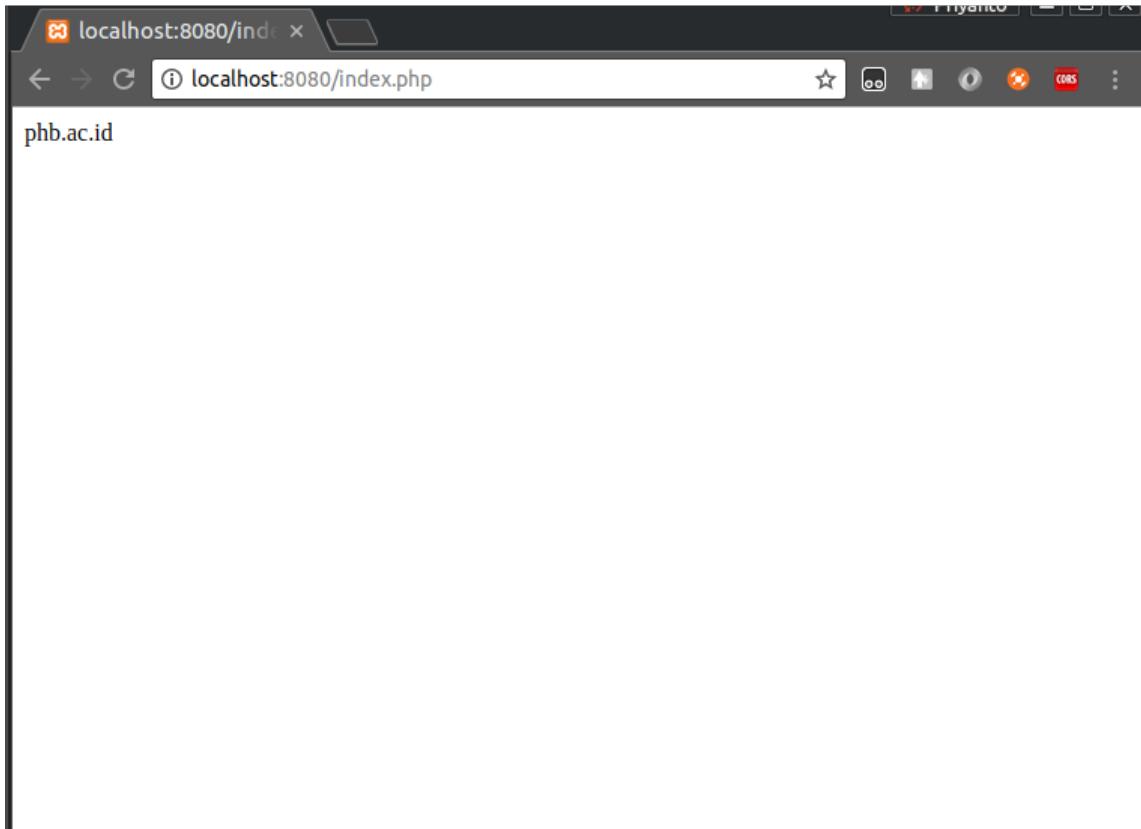
```

1 <?php
2 class DBConfig {
3     const HOST = 'phb.ac.id';
4 }
5

```

```
6 echo DBConfig ::HOST;  
7 ?>
```

Pada kelas DBConfig terdapat konstanta `HOST` dengan isi adalah alamat server basis data `phb.ac.id`, kita dapat memanggil isi dari konstanta ini seperti perintah pada baris ke-6, hasil keluaran dari kode di atas adalah seperti pada gambar 5.6 berikut :



Gambar 5.6: Hasil Keluaran Contoh Sederhana Penggunaan Konstanta

Sebetulnya pada setiap kelas pasti memiliki konstruktor baik disebutkan secara eksplisit maupun implicit, fungsinya yaitu tempat untuk melakukan inisialisasi pada saat objek kelas tersebut mulai dibentuk. Deklarasi dari konstruktor ini berbeda untuk tiap versi PHP, kita ikuti saja dari versi terbarunya, karena deklarasi pada versi lama akan dihapuskan. Berikut adalah contoh kode menggunakan konstruktor kelas secara sederhana :

```
1 <?php  
2 class Karyawan {  
3     private $nama = '';  
4  
5     function __construct($nama) {  
6         $this->nama = $nama;  
7     }  
8 }
```

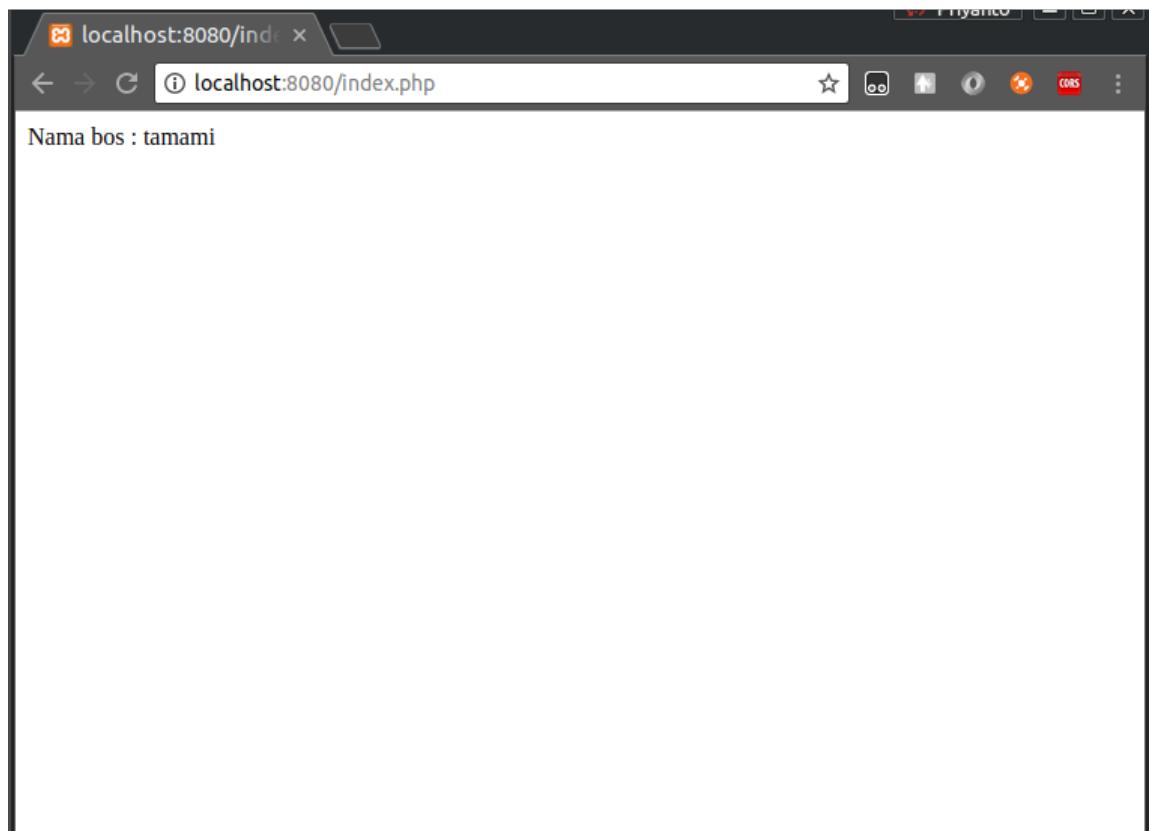
```

9  function getNama() {
10    return $this->nama;
11  }
12}
13
14 class Bos extends Karyawan {
15   function __construct($nama) {
16     parent::__construct($nama);
17   }
18}
19
20 $bos = new Bos("tamami");
21 echo "Nama bos : " . $bos->getNama();
22 ?>
```

Pada kode di atas, ada kelas **Bos** yang mewarisi kelas **Karyawan** dimana pada kelas **Karyawan** sudah memiliki properti **nama**, sebuah konstruktor dengan parameter **\$nama**, dan sebuah fungsi untuk mengambil nilai dari properti **nama**.

Pada kelas **Bos** hanya memiliki sebuah konstruktor dengan parameter **\$nama**. Isi dari konstruktor kelas **Bos** sendiri akan memanggil konstruktor milik kelas di atasnya. Karena semua properti dan fungsi telah diwariskan, maka kita dapat memanggilnya seperti pada baris ke-20 dan ke-21.

Hasil keluaran dari kode di atas adalah seperti pada gambar 5.7 berikut :



Gambar 5.7: Hasil Keluaran Contoh Penggunaan Konstruktor Kelas

Sebagian besar yang kita gunakan untuk kelas adalah seperti itu, pembahasan lebih lanjut bisa dilihat pada materi Pemrograman Berorientasi Objek.

5.3.3 Pemrosesan *File*

Pemrosesan *file* pada PHP harus dilakukan secara cermat, kesalahan yang sering terjadi biasanya pada kekeliruan merubah isi *file*, memproduksi *file* yang tidak begitu penting sehingga memenuhi *harddisk*, dan salah menghapus data secara tidak sengaja.05-03-03.file.tex

Membuat *File*

Untuk membuat sebuah *file* dengan PHP, kita dapat menggunakan fungsi `fopen` yang sudah disediakan, contoh kode sederhananya adalah seperti berikut ini :

```

1 <?php
2 $filenya = fopen("main.txt", "w+");
3 fclose($filenya);
4 ?>

```

Kode tersebut apabila dieksekusi akan menghasilkan sebuah *file* dengan nama `main.txt`, periksalah dibawah *folder htdocs* milik Xampp.

Jadi isi parameter pertama dari fungsi `fopen` ini adalah nama *file* yang akan dibuat atau dibaca, atau ditulis, sedangkan parameter kedua adalah mode yang akan digunakan untuk operasi pada *file*. Beberapa mode yang bisa kita gunakan dapat dilihat pada tabel 5.1 berikut :

Mode	Keterangan
r	Membuka <i>file</i> untuk dibaca saja. <i>Pointer</i> ada di awal <i>file</i>
w	Membuka <i>file</i> untuk menulis saja. Membuat <i>file</i> baru bila belum ada, atau menghapus seluruhnya
a	Membuka <i>file</i> untuk menulis saja. Membuat <i>file</i> baru bila belum ada, bila sudah ada <i>pointer</i> ditambahkan di akhirnya
x	Membuat <i>file</i> baru untuk menulis saja. Bila <i>file</i> sudah ada akan mengembalikan nilai false
r+	Membuka <i>file</i> untuk menulis dan dibaca. Akan munculkan warning saat <i>file</i> yang hendak ditulis belum dibuat
w+	Membuka <i>file</i> untuk menulis dan baca. Bila <i>file</i> tidak ada akan dibuatkan <i>file</i> baru, namun jika ada akan menghapus seluruhnya
a+	Membuka <i>file</i> untuk menulis dan baca. Bila <i>file</i> tidak ada akan dibuatkan <i>file</i> baru, posisi ditambahkan di akhirnya
x+	Membuat <i>file</i> untuk menulis dan baca. Bila <i>file</i> sudah ada akan mengembalikan nilai false

Tabel 5.1: Mode Operasi *File*

Menulis *File*

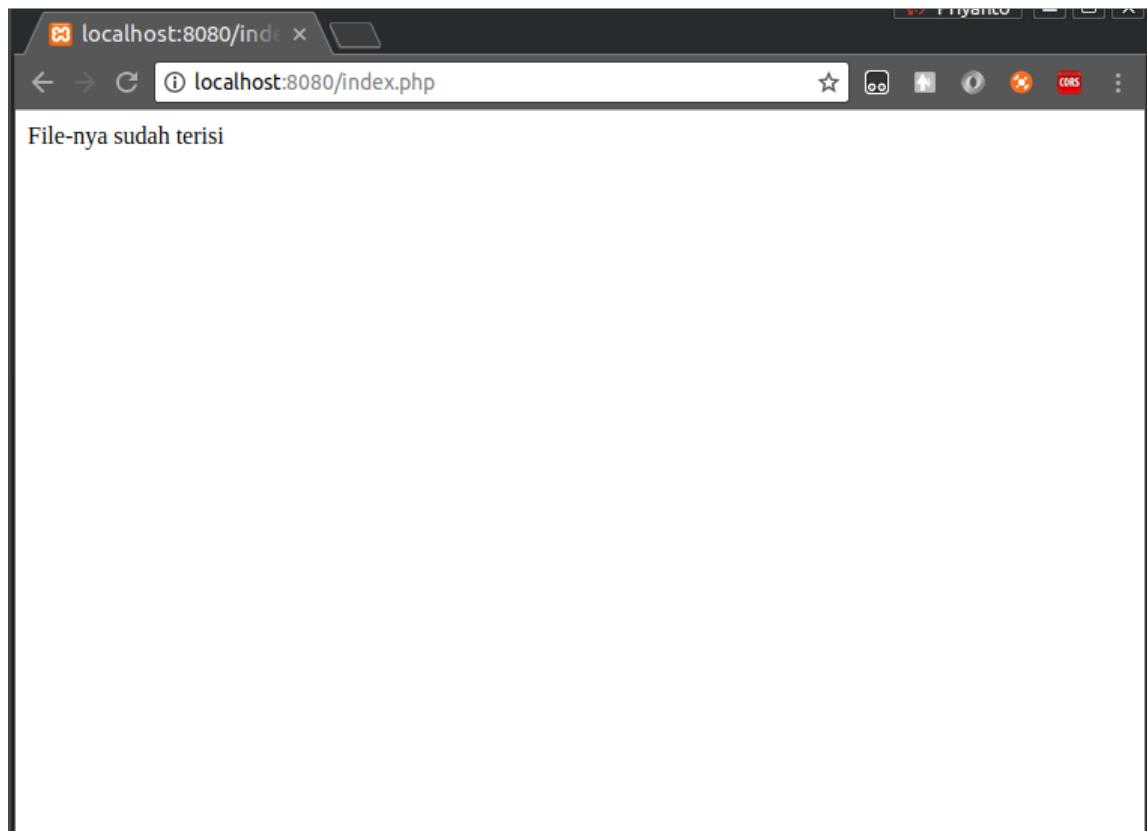
Untuk membuat sebuah *file* dan menuliskan isi ke dalamnya contoh kodennya adalah sebagai berikut :

```

1 <?php
2 $filenya = fopen("main.txt", "w+") or die("Tidak dapat membuka file!");
3
4 fwrite($filenya, "Ini nanti masuk ke file jadi isinya");
5 echo "File-nya sudah terisi";
6
7 fclose($filenya);
8 ?>

```

Fungsi **fwrite** yang melakukan penulisan isi ke dalam *file*. Hasil keluaran untuk kode di atas adalah seperti pada gambar 5.8 berikut :



Gambar 5.8: Hasil Keluaran Contoh Menulis Ke Dalam *File*

Isi dari *file* yang telah terbentuk adalah seperti berikut ini :

```
1 Ini nanti masuk ke file jadi isinya
```

Membaca *File*

Untuk membaca isi *file* lalu kita tampilkan hasilnya ke *browser* dapat kita lihat pada contoh kode berikut :

```

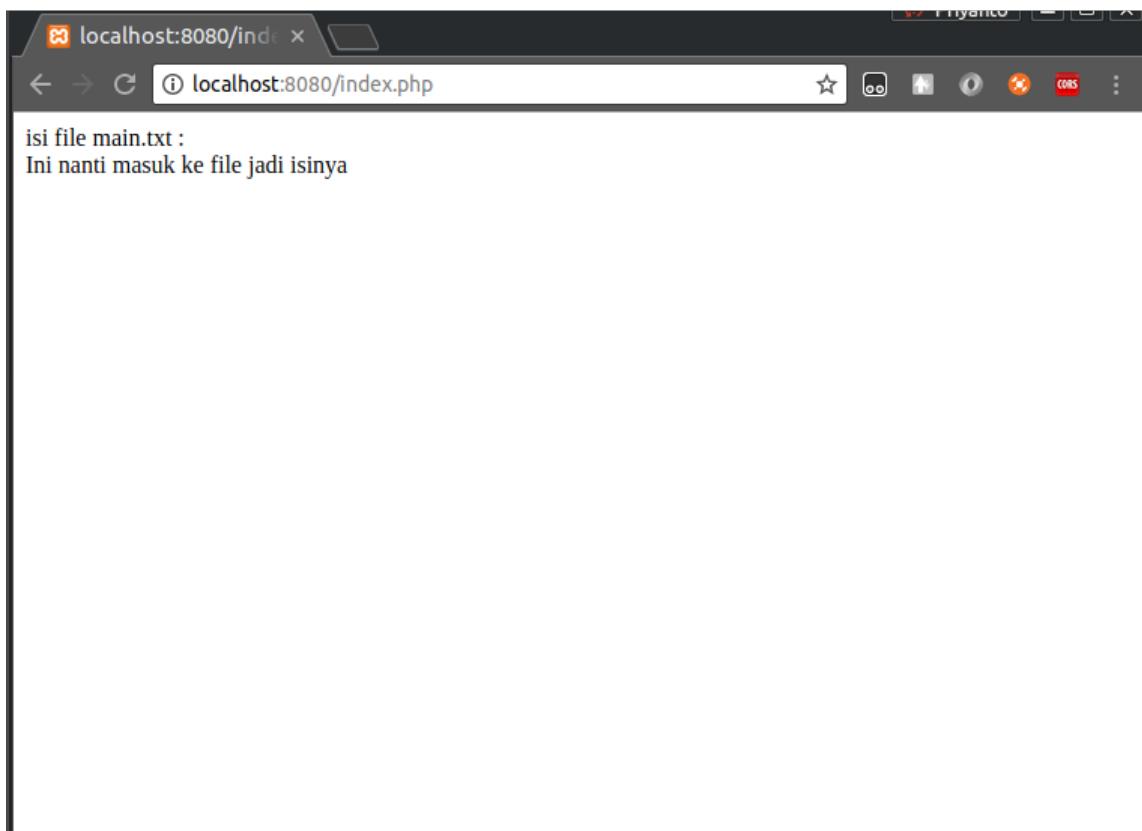
1 <?php
2 $filenya = fopen("main.txt", "r+");
3

```

```
4 echo 'isi file main.txt : <br>';
5 echo fread($filenya, filesize("main.txt"));
6
7 fclose($filenya);
8 ?>
```

Perhatikan bahwa mode yang digunakan adalah `r+`. Kali ini kita membaca dengan fungsi `fread()` dimana parameter pertamanya adalah variabel yang membuka *file* menggunakan fungsi `fopen()`, sedangkan parameter yang kedua adalah ukuran dalam *byte* dari *file* yang dibuka.

Hasil keluaran untuk kode tersebut adalah seperti pada gambar 5.9 berikut :



Gambar 5.9: Hasil Keluaran Contoh Sederhana Membaca *File*

Apabila isi *file* yang ada di *server* lebih dari 1 (satu) baris, kita pun dapat membacanya baris demi baris dengan fungsi `fgets()`. Misalkan ada *file* teks dengan nama `contoh.txt` dengan isi *file* seperti berikut :

```
1 ini isi baris ke-1
2 ini isi baris ke-2
3 ini isi baris ke-3
```

Contoh kode untuk membaca *file* tersebut baris demi baris adalah dengan kode berikut :

```
1 <?php
```

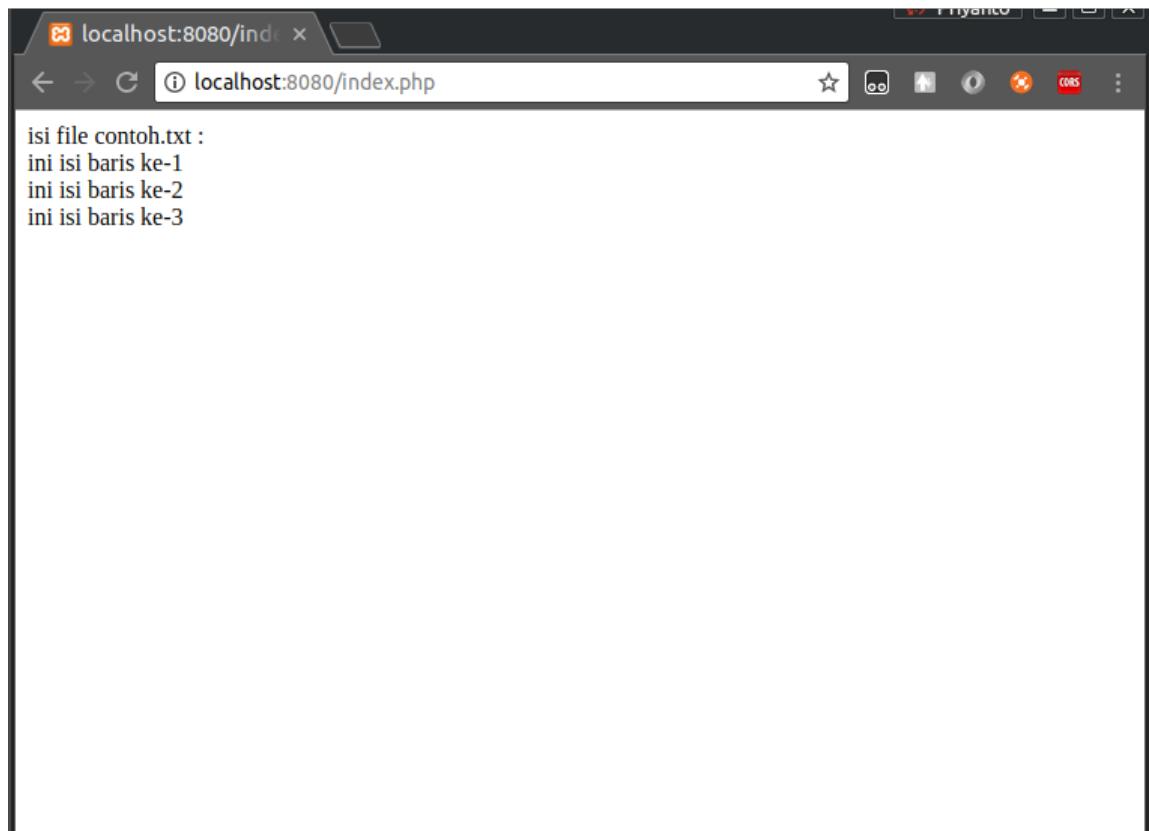
```

2 $filenya = fopen("contoh.txt", "r+");
3
4 echo 'isi file contoh.txt : <br>';
5
6 while (!feof($filenya)) {
7     echo fgets($filenya) . "<br>";
8 }
9
10 fclose($filenya);
11 ?>

```

Pertama kita buka *file contoh.txt* dengan fungsi `fopen()` pada baris ke-2, kemudian melakukan iterasi dengan `while`, selama *pointer* bukan berada di akhir *file* seperti pada baris ke-6, maka iterasi akan selalu membaca tiap baris dan menampilkannya ke *browser* dengan fungsi `fgets()`. Tiap kali fungsi `fgets()` dipanggil, *pointer* akan pindah ke baris berikutnya sampai *pointer* berada di akhir *file*.

Hasil keluaran dari kode di atas adalah seperti pada gambar 5.10 berikut :



Gambar 5.10: Hasil Keluaran Contoh Pembacaan *File* Tiap Baris

Selain cara di atas, kita juga dapat melakukan unggah *file* dari komputer lokal yang kita gunakan ke *server* dengan PHP, terutama untuk *file* berbentuk gambar dan caranya pun cukup mudah.

Yang pertama kita lakukan adalah memastikan bahwa PHP memperbolehkan pengguna untuk melakukan unggah data, *file* konfigurasi yang perlu kita periksa adalah *file* `php.ini`. *File* ini biasanya terdapat di dalam *folder* `etc` milik Xampp. Pastikan bahwa nilainya adalah seperti ini :

```
1 file_uploads = On
```

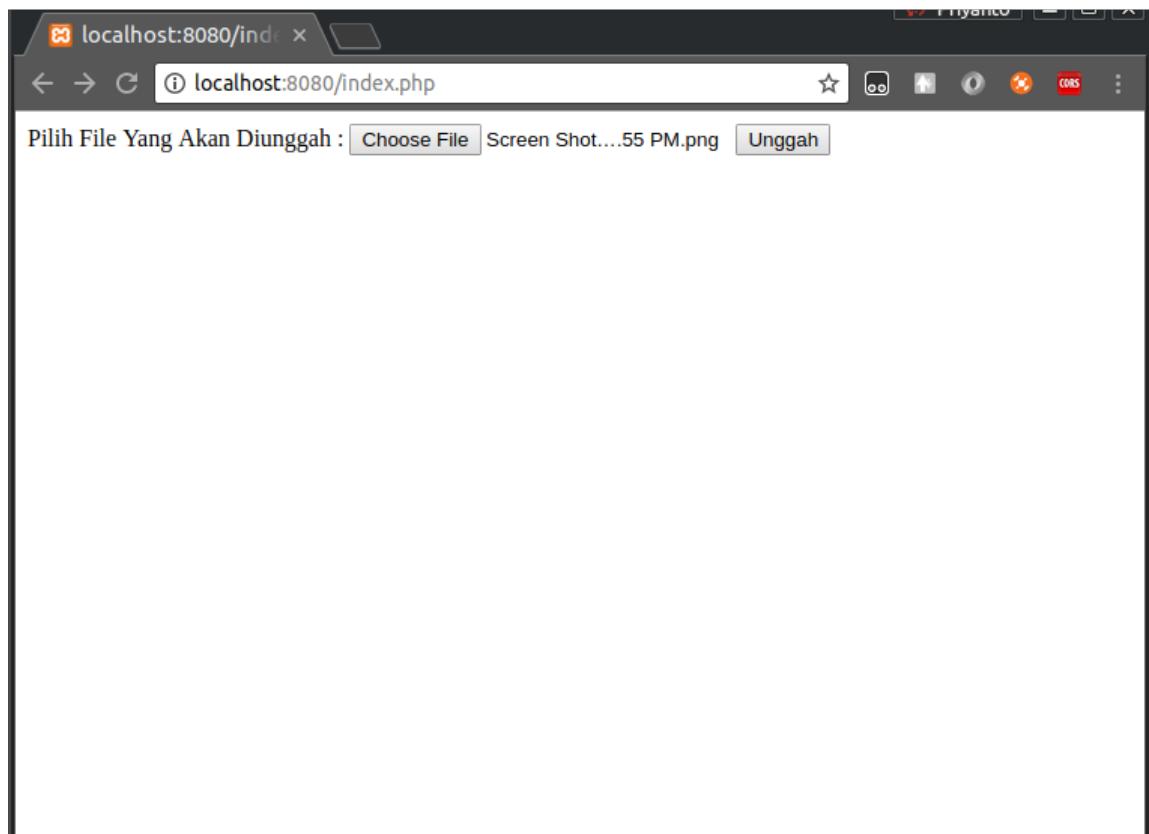
Kita memerlukan 2 (dua) *file*, yang pertama untuk menampilkan data masukkan yang dibutuhkan, *file* yang kedua untuk memproses unggah data. Isi kode dari *file* yang pertama (misalkan kita beri nama `index.php`) adalah sebagai berikut :

```
1 <form action="unggah.php" method="post" enctype="multipart/form-data">
2   Pilih File Yang Akan Diunggah :
3   <input type="file" name="fileUnggahan">
4   <input type="submit", value="Unggah" name="Kirim">
5 </form>
```

Yang perlu dicatat dari kode di atas adalah bahwa datanya nanti akan dikirim ke `unggah.php` dengan *method* `post` dan tipe *encoding* yang digunakan untuk menegir-imkan data ini adalah `multipart/form-data` yang biasanya digunakan untuk mengirimkan *raw* data biner.

Kemudian parameter yang jadi acuan di *file* `unggah.php` adalah atribut `name` milik *tag input*, yaitu `fileUnggahan`.

File tersebut hanya menampilkan sebuah *form* isian untuk memilih sebuah *file* yang akan diunggah. Hasil keluaran untuk kode tersebut adalah seperti pada gambar 5.11 berikut :



Gambar 5.11: Hasil Keluaran *Form* Masukkan Untuk Unggah *File*

Sekarang mari kita perhatikan isi kode dari *file unggah.php* berikut :

```

1 <?php
2 $target_file = basename($_FILES["fileUnggahan"]["name"]);
3 if( move_uploaded_file($_FILES["fileUnggahan"]["tmp_name"],
4   $target_file) ) {
5   echo "File " . basename($_FILES["fileUnggahan"]["name"]) . " telah
6   terunggah";
7 }
8 ?>

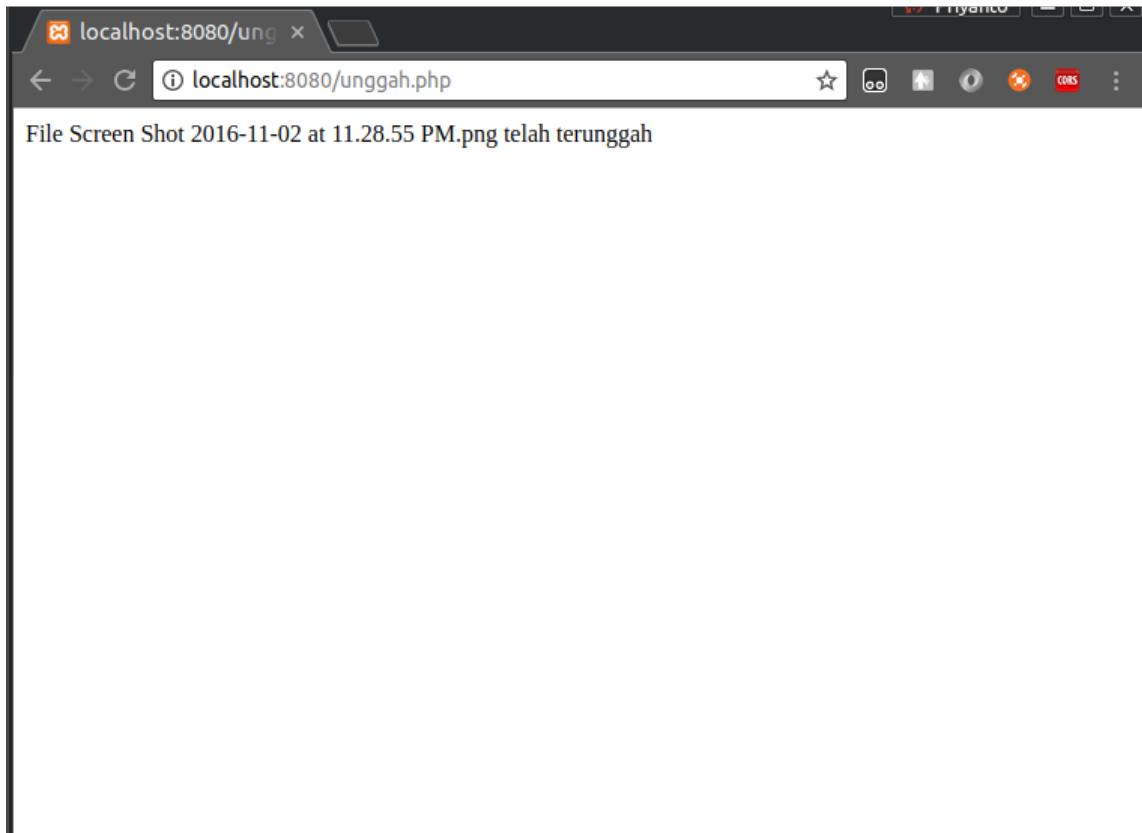
```

Data yang dikirim dari *file index.php* tadi akan disimpan pada variabel `$_FILES` yang bertipe *array* 2 (dua) dimensi, sehingga isinya dapat kita akses.

Key atau kunci pertama yang kita gunakan adalah isi dari parameter `name` di *file index.php*, yaitu `fileUnggahan`. Dan kunci kedua yang digunakan adalah `name`, yaitu nama *file* yang terpilih yang akan diunggah. Kita siapkan nama *file* di *server* sama seperti nama *file* di lokal komputer.

Begitu proses *file* selesai diunggah, *file* tidak otomatis berada pada `$target_file`, tetapi singgah dahulu pada `tmp_name`, atau *file* sementara di *server* kemudian dipindahkan menjadi nama aslinya dengan fungsi `move_uploaded_file()`.

Setelah proses unggah selesai, maka *browser* akan menampilkan informasi seperti pada gambar 5.12 berikut :



Gambar 5.12: Hasil Keluaran Akhir Hasil Unggah Data

Pastikan bahwa *file* yang terunggah ada pada *folder htdocs* di *server*.

5.4 Kesimpulan

Bahwa untuk membangun aplikasi yang lebih besar, akan mudah apabila dipecah ke dalam beberapa modul berupa kelas-kelas dan fungsi-fungsi yang memiliki tugas masing-masing.

Untuk melakukan unggah *file* menggunakan PHP, bisa memanfaatkan variabel `$_FILES` yang tipe datanya berupa *array*.

5.5 Tugas

Buatkan sebuah halaman yang nantinya menerima masukkan berupa :

- Nama Mahasiswa
- NIM
- Kelas

Hasil dari masukkan / *input* tersebut akan disimpan dalam sebuah *file* di *server*.

Bab 6

Konektivitas PHP dan Basis Data

6.1 Tujuan

Mampu untuk mengimplementasikan konektivitas antara PHP dengan sistem basis data dan melakukan operasi-operasi yang berhubungan dengan basis data.

6.2 Pengantar

Koneksi dengan basis data untuk membangun sebuah aplikasi pengolahan data adalah sebuah kewajiban, bila datanya tidak banyak, hanya ada beberapa buah mungkin bisa disimpan menggunakan beberapa *file*, namun bila datanya sudah ratusan dan antar data memiliki korelasi / hubungan, maka kita memerlukan sebuah sistem basis data untuk melakukan pengolahan data.

PHP sendiri mendukung konektivitas dan operasi-operasi yang berhubungan dengan basis data, terutama MySQL, karena dalam *bundle* yang kita pasang menggunakan Xampp, di dalamnya sudah terdapat sistem basis data MySQL yang siap kita gunakan, versi terbarunya mungkin akan menggunakan MariaDB, namun keduanya sama saja.

Dalam implementasinya, PHP sendiri dapat menggunakan MySQLi atau PDO (PHP Data Object) untuk terhubung dengan sistem basis data MySQL. Kita memilih salah satu.

MySQLi sesuai namanya, hanya bisa digunakan untuk terhubung dengan sistem basis data MySQL (atau MariaDB) saja, sedangkan penggunaan PDO dimungkinkan untuk melakukan koneksi ke selain sistem basis data MySQL.

6.3 Praktek

6.3.1 Memulai Koneksi

Agar PHP dapat berkomunikasi dengan sistem basis data, tentu saja kita harus melakukan konfigurasi untuk memulai koneksi. Berikut akan kita coba melakukan sambungan dengan sistem basis data MySQL baik dengan MySQLi maupun PDO.

Kode yang pertama kita akan coba menggunakan MySQLi, perhatikan kode program berikut :

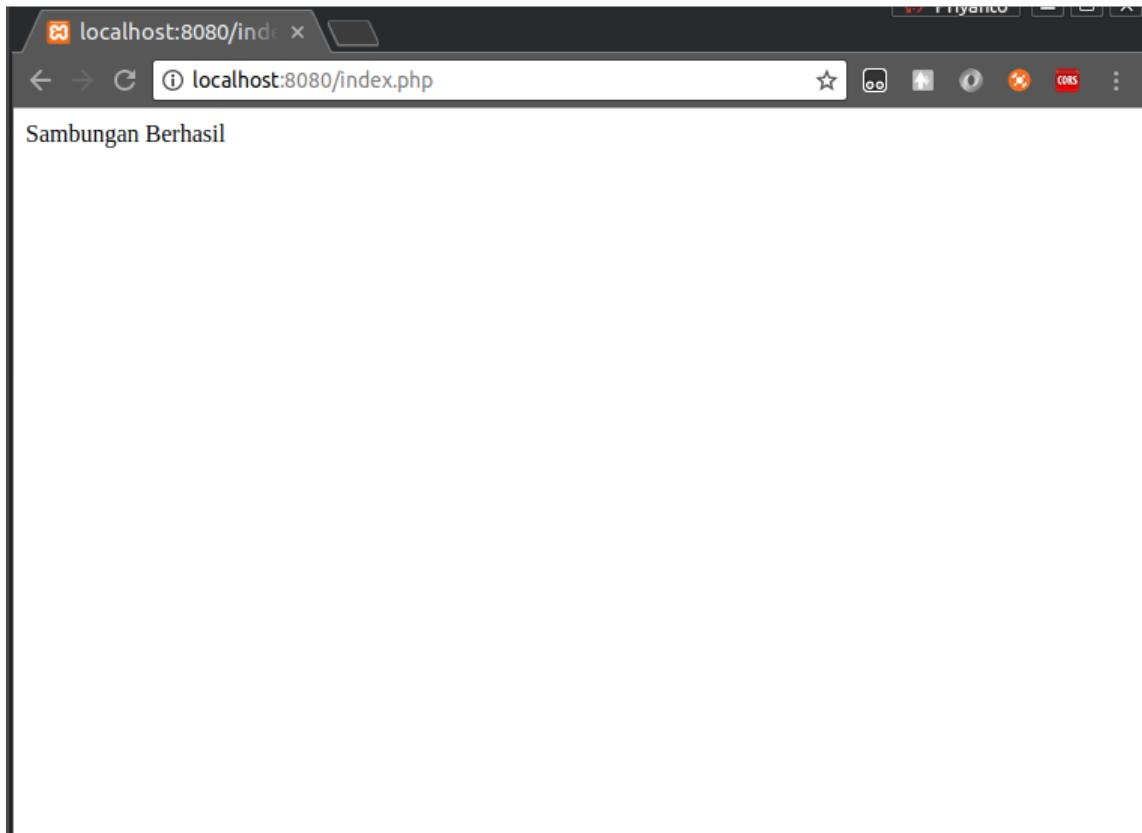
```
1 <?php
2 $server = "localhost";
3 $username = "dev";
4 $password = "rahasia";
5 $db = "phb";
6
7 // memulai sambungan
8 $koneksi = new mysqli($server, $username, $password, $db);
9
10 // periksa sambungan
11 if($koneksi->connect_error) {
12     die("Sambungan gagal: " . $koneksi->connect_error);
13 }
14 echo "Sambungan Berhasil";
15
16 ?>
```

Dari baris ke-2 sampai dengan baris ke-5 adalah parameter yang digunakan untuk melakukan koneksi ke sistem basis data.

Pada baris berikutnya adalah membuat objek dari `mysqli` dengan parameter yang sudah disiapkan, hasilnya ditaruh pada variabel `$koneksi`.

Pada blok baris berikutnya, akan dilakukan pemeriksaan, apakah ada kesalahan koneksi atau tidak dengan pemanggilan properti `connect_error`, bila tidak ada maka akan ditampilkan informasi ke *browser* bahwa *Sambungan Berhasil*.

Hasil keluaran dari kode di atas adalah seperti pada gambar 6.1 berikut :



Gambar 6.1: Hasil Keluaran Memulai Sambungan Ke Sistem Basis Data

Bila kita menggunakan PDO, kodennya mirip seperti terlihat berikut ini :

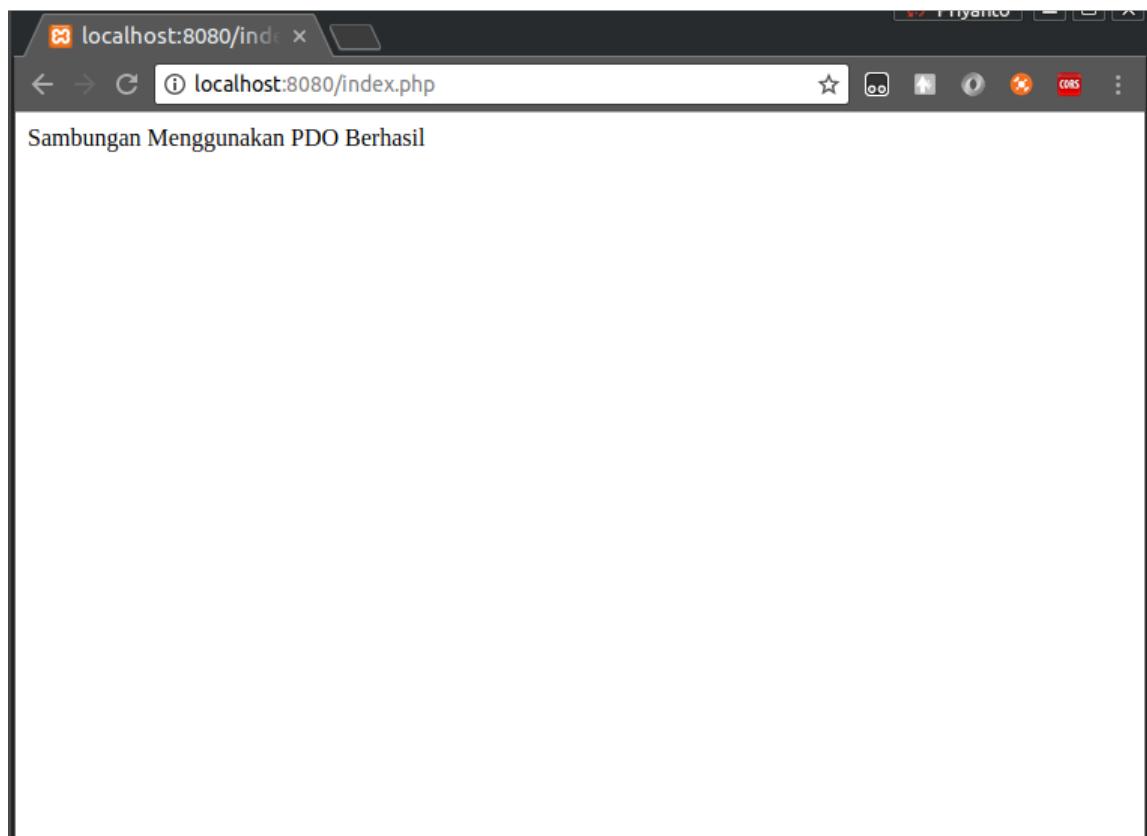
```
1 <?php
2 $server = "localhost";
3 $username = "dev";
4 $password = "rahasia";
5 $db = "phb";
6
7 try{
8     // memulai sambungan
9     $koneksi = new PDO("mysql:host=$server;dbname=$db", $username,
10                        $password);
11
12     // pindahkan mode error PDO ke exception
13     $koneksi->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
14     echo "Sambungan Menggunakan PDO Berhasil";
15 } catch(PDOException $e) {
16     echo "Sambungan gagal : " . $e->getMessage();
17 }
```

Parameter yang digunakan masih sama, hanya saja saat membentuk objek dari kelas PDO, parameter pertama berupa informasi *host* dan nama basis data, sedangkan parameter kedua dan ketiga berurutan adalah *username* dan *password* untuk

terhubung dengan sistem basis data.

Apabila koneksi ke sistem basis data gagal, maka PDO akan menghasilkan *exception* karena atribut *error* PDO telah dirubah ke bentuk *exception*.

Hasil dari kode di atas adalah seperti pada gambar 6.2 berikut :



Gambar 6.2: Hasil Keluaran Koneksi Sistem Basis Data Dengan PDO

6.3.2 Masukkan Data

Sekarang kita coba bagaimana caranya memasukkan data ke dalam sebuah tabel dengan PHP. Data yang dimasukkan sederhana, berupa stok barang yang ada di gudang. Tabelnya mungkin dapat kita beri nama **stok**, berikut struktur tabelnya :

Kolom	Tipe Data	Keterangan
kode	integer	primary key
nama_barang	varchar(50)	
stok	integer	

Tabel 6.1: Struktur Tabel Stok

Untuk mengimplementasikan tambah data, kita akan pecah menjadi beberapa *file*

yang nantinya pada akhir materi ini, aplikasi akan menjadi utuh memiliki beberapa operasi data standar.

Pertama kita buat dahulu *file* atau kelas konfigurasi basis data yang nantinya akan digunakan di tiap operasi tambah data, ambil data, ubah data dan hapus data.

Kita sebut saja nama *file* ini adalah **conf-db.php** dengan kode sebagai berikut :

```

1 <?php
2 class ConfigDb {
3     const SERVER = "localhost";
4     const USERNAME = "dev";
5     const PASSWORD = "rahasia";
6     const DB = "phb";
7
8     static function getKoneksi() {
9         return new mysqli(ConfigDb :: SERVER, ConfigDb :: USERNAME,
10            ConfigDb :: PASSWORD, ConfigDb :: DB);
11    }
12 }
13 ?>

```

Kelas ini akan menyimpan konfigurasi basis data yang dapat digunakan di beberapa halaman PHP yang membutuhkan, kelas ini pun memiliki fungsi **getKoneksi()** yang memudahkan untuk membuka koneksi dengan sistem basis data.

File yang kedua adalah halaman formulir pengisian data, kita sebut saja nama *file* ini adalah **input.php** yang isi kodennya adalah seperti berikut ini :

```

1 <h2>Form Entry Data Stok Barang</h2>
2 <hr>
3 <form action="simpan.php" method="post">
4     <table>
5         <tr>
6             <td>KODE</td>
7             <td><input type="text" name="kode"></td>
8         </tr>
9         <tr>
10            <td>NAMA BARANG</td>
11            <td><input type="text" name="namaBarang"></td>
12        </tr>
13        <tr>
14            <td>STOK</td>
15            <td><input type="text" name="stok"></td>
16        </tr>
17        <tr>
18            <td><input type="submit" value="Simpan"></td>
19        </tr>
20    </table>
21 </form>

```

File tersebut memang tidak ada kode dalam bahasa PHP, hanya menyiapkan saja bahan-bahan yang digunakan untuk memasukkan data oleh pengguna, kemudian hasilnya dikirim ke *file* **simpan.php**.

Yang perlu diperhatikan disini adalah isi dari atribut **name** milik tiap komponen **input**, karena nama-nama ini akan menjadi *key* atau kunci pada variabel **\$_POST** pada *file* **simpan.php**.

Tampilan dari kode di atas adalah seperti pada gambar 6.3 berikut ini :

The screenshot shows a web browser window with the URL `localhost:8080/input.php`. The page title is "Form Entry Data Stok Barang". It contains three text input fields and one button:

- KODE**: Value 1
- NAMA BARANG**: Value gula
- STOK**: Value 4

Below the inputs is a **Simpan** button.

Gambar 6.3: Hasil Keluaran Bentuk Formulir Masukkan Data

Sekarang mari kita lihat isi kode dari file `simpan.php` yang akan melakukan proses simpan data ke sistem basis data, berikut isi kodennya :

```

1 <?php
2 include "conf-db.php";
3
4 $koneksi = ConfigDb::getKoneksi();
5
6 if($koneksi->connect_error) {
7     die("Koneksi ke Basis Data gagal : " . $koneksi->connect_error);
8 }
9
10 $query = "insert into stok(kode, nama_barang, stok) values(" .
11     $_POST['kode'] . " ,'" . $_POST["namaBarang"] . "' ,'" . $_POST["stok"]
12     . ")";
13 if($koneksi->query($query) === TRUE) {
14     echo "Data " . $_POST["namaBarang"] . " telah ditambahkan";
15 } else {
16     echo "Error : " . $query . "<br>" . $koneksi->error;
17 }
18

```

```
19 $koneksi->close();  
20 ?>
```

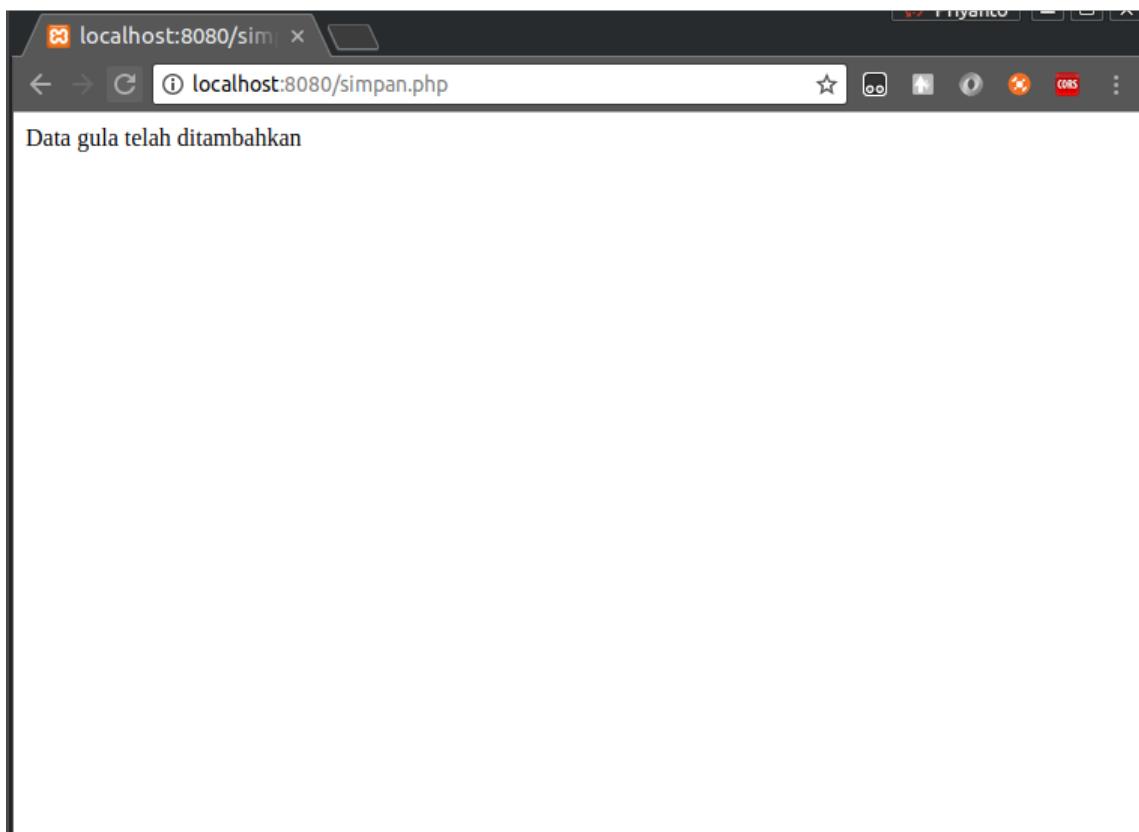
Pertama *file* ini akan memasukkan *file conf-db.php* agar konfigurasi basis data tidak perlu dideklarasikan ulang, tinggal gunakan saja seperti pada baris ke-4, dengan memanggil fungsi dengan tipe statis *getKoneksi()* milik kelas *ConfigDb*.

Berikutnya memastikan sambungan bahwa telah terhubung ke sistem basis data, perintahnya ada pada baris ke-6 sampai ke-8.

Baris ke-10 dan ke-11 menyiapkan *query* untuk menambahkan data ke sistem basis data, semua data yang dimasukkan ke dalam basis data akan diambil dari hasil kiriman *file input.php* melalui *method POST* dengan melakukan akses ke variabel *\$_POST*.

Pada baris ke-13, melakukan eksekusi terhadap *query* yang telah disiapkan sebelumnya, bila *query* berhasil maka berikan informasi ke pengguna melalui *browser* bahwa data telah tersimpan ke sistem basis data, apabila gagal maka berikan informasi kegagalannya.

Informasi ditampilkan seperti pada gambar 6.4 berikut :



Gambar 6.4: Hasil Keluaran Informasi Berhasil Simpan Ke Basis Data

Sampai sini bisa dipastikan bahwa data telah tersimpan dalam basis data, perhatikan isi tabelnya.

6.3.3 Ambil Data

Untuk melakukan pengambilan data, kita akan coba buatkan halaman yang menjadi halaman utama aplikasi yang kita bangun. Halaman ini kita beri nama *file main.php*, kodennya adalah seperti berikut ini :

```

1 <h2>Aplikasi Data Barang</h2>
2
3 <hr>
4 <a href="input.php">Tambah Data</a>
5 <table border="1">
6   <tr>
7     <th>KODE</th>
8     <th>NAMA BARANG</th>
9     <th>STOK</th>
10    </tr>
11    <?php
12      include "conf-db.php";
13      $koneksi = ConfigDb::getKoneksi();
14
15      if ($koneksi->connect_error) {
16          die("Koneksi ke Basis Data gagal : " . $koneksi->
17          connect_error);
18      }
19
20      $query = "select * from stok";
21
22      $result = $koneksi->query($query);
23
24      if ($result->num_rows > 0) {
25          while ($data = $result->fetch_assoc()) {
26              echo "<tr>";
27              echo "<td>" . $data["kode"] . "</td>";
28              echo "<td>" . $data["nama_barang"] . "</td>";
29              echo "<td>" . $data["stok"] . "</td>";
30              echo "</tr>";
31      } else {
32          echo "<tr>";
33          echo '<td colspan="3">Data NIHIL</td>';
34          echo "</tr>";
35      }
36
37      $koneksi->close();
38    ?>
39  </table>
40
41 <?php
42 ?>
```

File ini menampilkan daftar data yang ada pada tabel **stok** dalam basis data, perhatikan pada baris ke-4 bahwa kita menambahkan sebuah *link* ke formulir masukkan data baru yang sebelumnya telah kita buat.

Pembuatan tabel seperti pada umumnya dengan HTML, namun pada baris ke-2

dari tabel, kita akan memasukkan kode PHP untuk melakukan pengambilan data ke basis data dan menampilkannya dalam tabel.

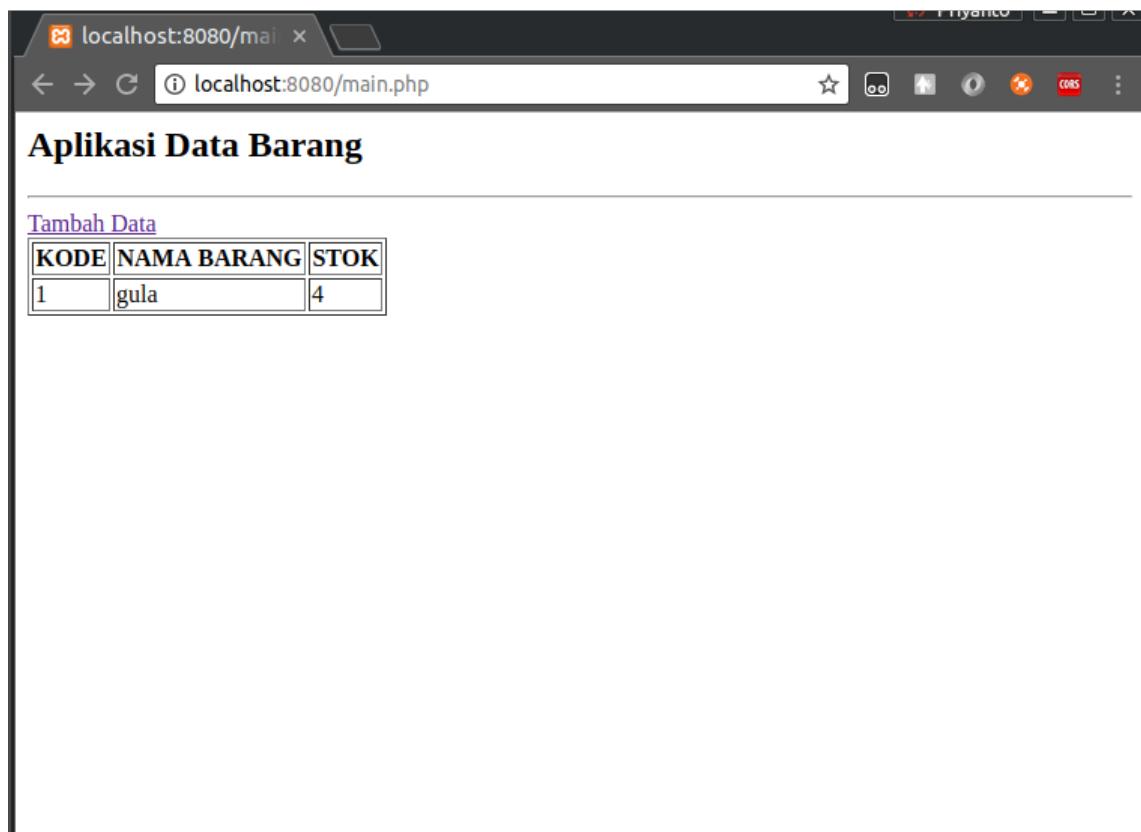
Seperti *file* sebelumnya, untuk melakukan koneksi ke sistem basis data kita memerlukan sebuah koneksi yang konfigurasinya tersimpan di *file conf-db.php*, kita gunakan perintah *include* seperti pada baris ke-12, kemudian langsung mengambil koneksinya dan disimpan pada variabel *\$koneksi* seperti pada baris ke-13.

Setelah melakukan koneksi, kita perlu memastikan dahulu apakah koneksi telah berhasil tersambung, kondisi ini dilakukan oleh blok baris ke-15 sampai dengan baris ke-17.

Baris ke-19 akan menyiapkan skrip sql-nya, yang kemudian dieksekusi pada baris ke-21 yang hasil eksekusinya akan disimpan pada variabel *\$result*.

Seleksi berikutnya adalah memastikan bahwa basis data memiliki data yang akan ditampilkan, seleksi ini ada pada baris ke-23. Blok baris ke-24 sampai dengan baris ke-30, kita melakukan iterasi semua data yang ada pada basis data untuk ditampilkan ke *browser*. Bila basis data tidak memiliki data, maka blok baris ke-32 sampai dengan baris ke-34 yang akan dikerjakan.

Hasil keluaran dari kode di atas tampilannya akan terlihat seperti pada gambar 6.5 berikut :



Gambar 6.5: Hasil Keluaran Daftar Stok Barang

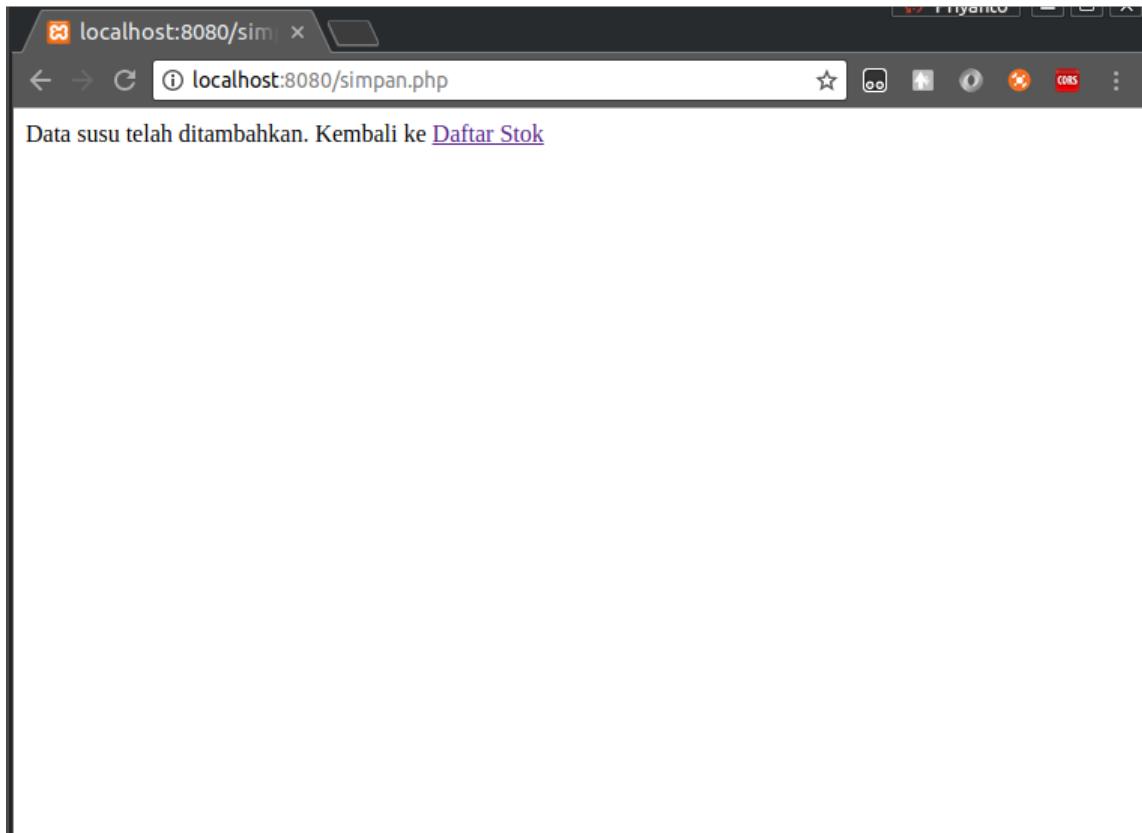
Kita perlu merubah kode pada *file simpan.php* agar memiliki *link* untuk kembali

ke halaman daftar stok barang, berikut kode lengkap dari *file simpan.php* yang baru :

```

1 <?php
2 include "conf-db.php";
3
4 $koneksi = ConfigDb::getKoneksi();
5
6 if($koneksi->connect_error) {
7     die("Koneksi ke Basis Data gagal : " . $koneksi->connect_error);
8 }
9
10 $query = "insert into stok(kode, nama_barang, stok) values(" .
11     $_POST['kode'] . ",'" . $_POST["namaBarang"] . "','" . $_POST["stok"]
12     "] . ")";
13 if($koneksi->query($query) == TRUE) {
14     echo "Data " . $_POST["namaBarang"] . ' telah ditambahkan. Kembali
15     ke <a href="main.php">Daftar Stok</a>';
16 } else {
17     echo "Error : " . $query . "<br>" . $koneksi->error;
18 }
19 $koneksi->close();
20 ?>
```

Penambahan kode yang kita lakukan ada pada baris ke-14. Hasil keluaran dari kode tersebut adalah apabila kita melakukan tambah data baru, kemudian mengisikan data yang diminta, setelah menekan tombol simpan, *file simpan* ini akan dipanggil dan memberikan tampilan seperti pada gambar 6.6 berikut :



Gambar 6.6: Hasil Keluaran Perubahan File simpan.php

6.3.4 Perbarui Data

Untuk perubahan data, langkah yang dilakukan mirip dengan penambahan data baru, hanya saja pada saat formulir ditampilkan, data lama akan ikut ditampilkan yang kemudian memberikan kesempatan bagi pengguna untuk melakukan perubahan data.

Pertama kita lakukan perubahan dahulu pada file `main.php` agar data yang ditampilkan dapat kita lakukan perubahan data. Kode setelah perubahan seperti berikut ini :

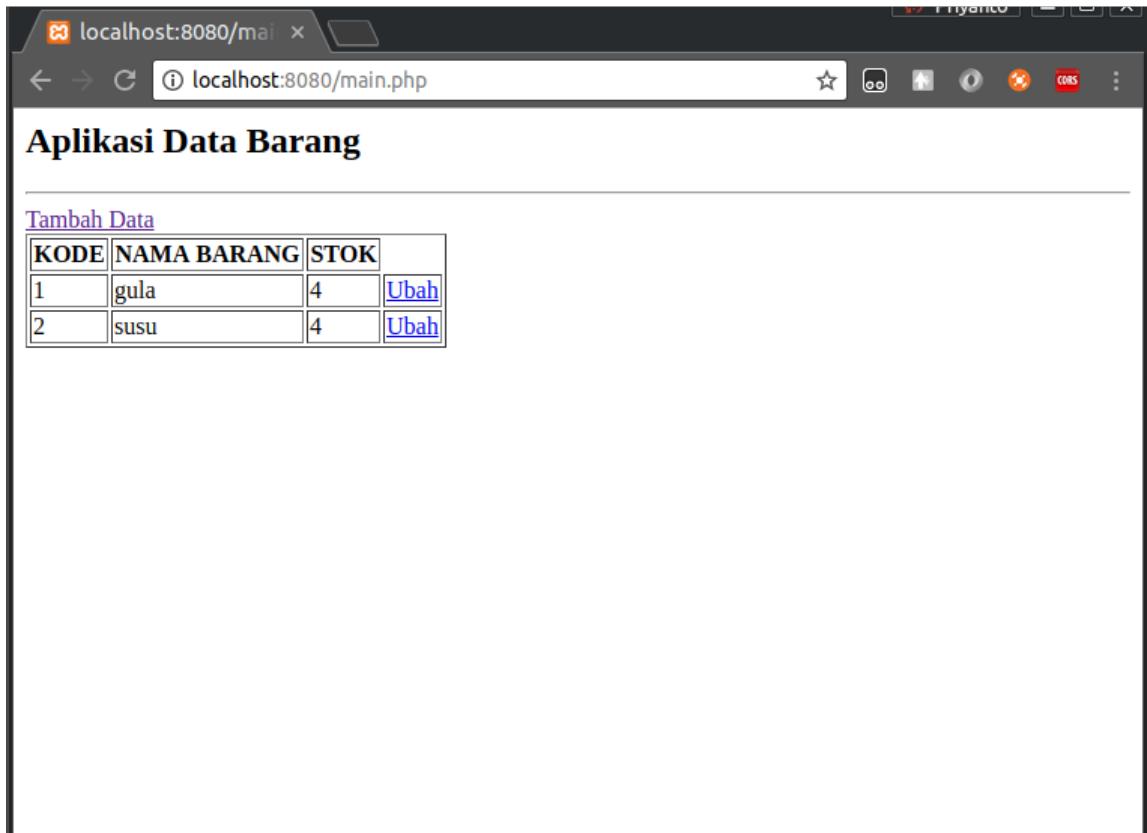
```
1 <h2>Aplikasi Data Barang</h2>
2
3 <hr>
4 <a href="input.php">Tambah Data</a>
5 <table border="1">
6   <tr>
7     <th>KODE</th>
8     <th>NAMA BARANG</th>
9     <th>STOK</th>
10    </tr>
11    <?php
12      include "conf-db.php";
13      $koneksi = ConfigDb::getKoneksi();
```

```

14     if ($koneksi->connect_error) {
15         die("Koneksi ke Basis Data gagal : " . $koneksi->
16             connect_error);
17     }
18
19     $query = "select * from stok";
20
21     $result = $koneksi->query($query);
22
23     if ($result->num_rows > 0) {
24         while ($data = $result->fetch_assoc()) {
25             echo "<tr>";
26             echo "<td>" . $data["kode"] . "</td>";
27             echo "<td>" . $data["nama_barang"] . "</td>";
28             echo "<td>" . $data["stok"] . "</td>";
29             echo '<td><a href="ubah.php?kode=' . $data["kode"] . '',
30                  "Ubah</a></td>';
31             echo "</tr>";
32         }
33     } else {
34         echo "<tr>";
35         echo '<td colspan="3">Data NIHIL</td>';
36         echo "</tr>";
37     }
38     $koneksi->close();
39     ?>
40 </table>

```

Penambahan baris kode untuk *file* ini ada pada baris ke-29. *Link* tersebut digunakan untuk membuka halaman baru untuk pengisian perubahan data. Tampilannya akan terlihat seperti pada gambar 6.7 berikut :



Gambar 6.7: Hasil Keluaran Penambahan Fasilitas Ubah Data

Begitu tombol / link diklik, maka akan muncul halaman dari file ubah.php dengan isi kode seperti berikut ini :

```

1 <?php
2 include "conf-db.php";
3
4 $koneksi = ConfigDb::getKoneksi();
5
6 if($koneksi->connect_error) {
7     die("Koneksi ke Basis Data gagal : " . $koneksi->connect_error);
8 }
9
10 $query = "select kode, nama_barang, stok from stok where kode = " .
11     $_GET["kode"];
12
13 $result = $koneksi->query($query);
14 ?>
15
16 <h2>Formulir Ubah Data</h2>
17 <hr>
18 <?php
19 if($result->num_rows > 0) {
20     $data = $result->fetch_assoc();
21 ?>
22 <form action="simpan-ubah.php" method="post">
```

```

22 <table>
23     <tr>
24         <td>KODE</td>
25         <td><input type="text" name="kode" value="<?php echo $data["
26             "kode"]; ?>" readonly="true" ></td>
27     </tr>
28     <tr>
29         <td>NAMA BARANG</td>
30         <td><input type="text" name="namaBarang" value="<?php echo
31             $data["nama_barang"]; ?>"></td>
32     </tr>
33     <tr>
34         <td>STOK</td>
35         <td><input type="text" name="stok" value="<?php echo $data["
36             "stok"]; ?>"></td>
37     </tr>
38 </table>
39 <input type="submit" value="Simpan">
40 </form>
41 <?php
42 } else {
43     echo "Anda salah memilih, data tidak ada dalam basis data<br>";
44     echo '<a href="main.php">Kembali</a>';
45 }
46 $koneksi->close();
47 ?>
```

Seperti biasanya kita membutuhkan *file conf-db.php* yang menyimpan parameter koneksi ke sistem basis data seperti terlihat pada baris ke-2, kemudian kita gunakan parameter koneksinya dengan memanggil fungsi *getKoneksi()* seperti pada baris ke-4, setelah itu kita periksa status koneksinya apakah berhasil atau gagal pada blok baris ke-6 sampai dengan baris ke-8.

Pada baris ke-10, kita menyiapkan *query* untuk mengambil data dari sistem basis data untuk ditampilkan dalam formulir perubahan data. Eksekusi *query* terjadi pada baris ke-12 yang hasilnya disimpan pada variabel *\$result*.

Sebelum menampilkan formulir perubahan, kita cek dulu apakah ada data dengan kode yang telah di *request*. Bila ada maka tampilkan formulirnya, bila tidak ada, tampilkan informasinya pada *browser*. Hal ini terjadi pada blok baris ke-18 sampai dengan baris ke-42.

Formulir perubahan ada pada blok baris ke-21 sampai dengan baris ke-37. Data dari formulir ini akan dikirimkan ke *file simpan-ubah.php* dengan metode *post*. Seperti sebelumnya, tanda yang dijadikan *key* pada *array \$_POST* adalah pada atribut *name* dari tiap komponen *input*.

Hasil keluaran dari kode di atas adalah seperti pada gambar 6.8 berikut :

The screenshot shows a web browser window with the URL `localhost:8080/ubah.php?kode=2`. The page title is "Formulir Ubah Data". There are three input fields: "KODE" with value "2", "NAMA BARANG" with value "susu", and "STOK" with value "3". Below the inputs is a "Simpan" button.

Gambar 6.8: Hasil Keluaran Formulir Perubahan Data

Proses selanjutnya adalah mengirimkan data ini ke file `simpan-ubah.php` apabila tombol Simpan di klik seperti deklarasi pada baris ke-21. Isi kodenya adalah seperti berikut ini :

```

1 <?php
2 include "conf-db.php";
3
4 $koneksi = ConfigDb::getKoneksi();
5
6 if ($koneksi->connect_error) {
7     die("Koneksi ke Basis Data gagal : " . $koneksi->connect_error);
8 }
9
10 $query = "update stok set nama_barang = '" . $_POST["namaBarang"] . "' ,
11      'stok = ' . $_POST["stok"] . ' where kode = ' . $_POST["kode"]
12 ;
13 if ($koneksi->query($query) === TRUE) {
14     echo "Data " . $_POST["namaBarang"] . ' telah diubah. Kembali ke <a'
15     href="main.php">Daftar Stok</a>';
16 } else {
17     echo "Error : " . $query . "<br>" . $koneksi->error;
18 }
```

```

18 $koneksi->close();
19 ?>
20

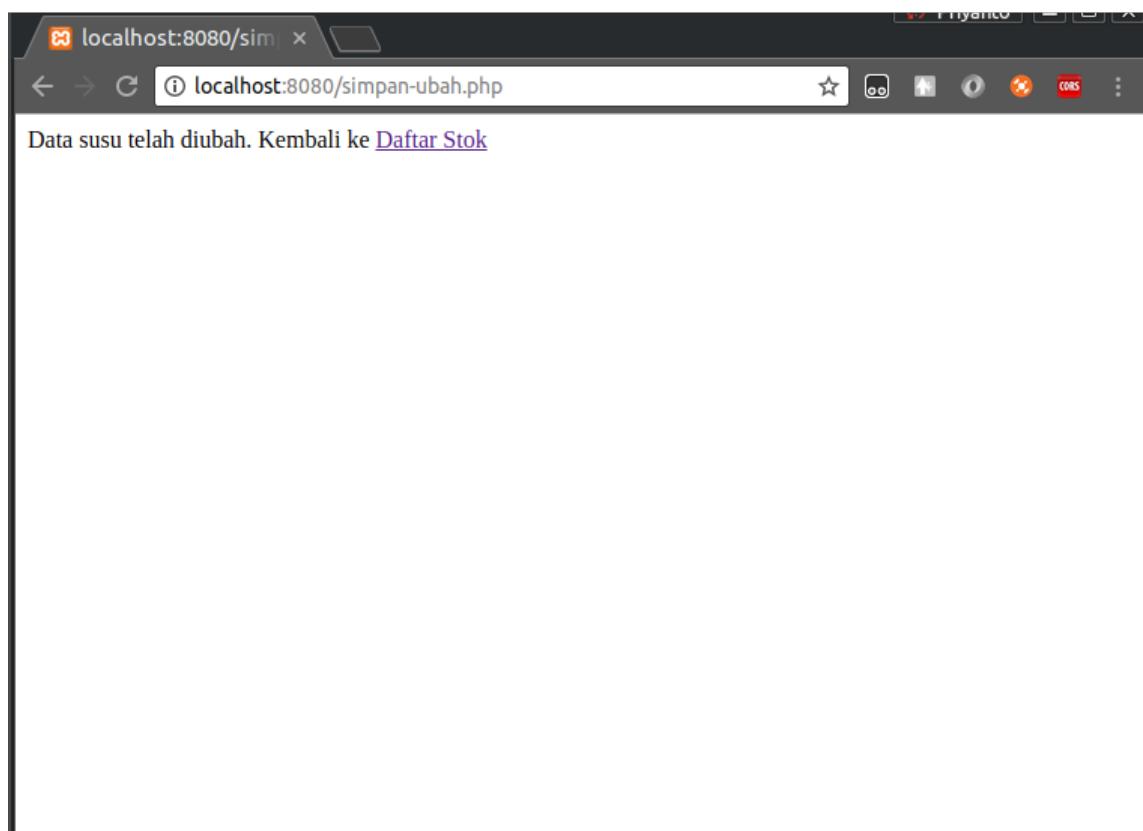
```

Karena *file* ini bertugas menyimpan perubahan data yang terjadi, maka akan menggunakan *file* `conf-db.php` seperti deklarasi pada baris ke-2.

Kemudian mengaktifkan koneksi seperti pada baris ke-4, dan melakukan periksaan status koneksi seperti pada blok baris ke-6 sampai baris ke-8 .

Query untuk *update* data disiapkan pada baris ke-10, yang kemudian dieksekusi sekaligus diperiksa apakah eksekusi *query* berhasil dijalankan atau tidak seperti pada blok baris ke-13 sampai baris ke-17.

Hasil keluaran dari kode di atas adalah seperti pada gambar 6.9 berikut :



Gambar 6.9: Hasil Keluaran Halaman Hasil Perubahan Data

6.3.5 Delete Data

Untuk memasangkan modul hapus data, kita perbarui dulu halaman utama kita (halaman `main.php` agar dapat menerima perintah hapus. Berikut adalah kode lengkap hasil perubahan dari *file* `main.php` :

```

1 <h2>Aplikasi Data Barang</h2>
2
3 <hr>

```

```

4 <a href="input.php">Tambah Data</a>
5 <table border="1">
6     <tr>
7         <th>KODE</th>
8         <th>NAMA BARANG</th>
9         <th>STOK</th>
10    </tr>
11    <?php
12        include "conf-db.php";
13        $koneksi = ConfigDb::getKoneksi();
14
15        if ($koneksi->connect_error) {
16            die("Koneksi ke Basis Data gagal : " . $koneksi->
17            connect_error);
18        }
19
20        $query = "select * from stok";
21
22        $result = $koneksi->query($query);
23
24        if ($result->num_rows > 0) {
25            while ($data = $result->fetch_assoc()) {
26                echo "<tr>";
27                echo "<td>" . $data["kode"] . "</td>";
28                echo "<td>" . $data["nama_barang"] . "</td>";
29                echo "<td>" . $data["stok"] . "</td>";
30                echo '<td><a href="ubah.php?kode=' . $data["kode"] . ,
31                    ">Ubah</a></td>';
32                echo '<td><a href="hapus.php?kode=' . $data["kode"] . ,
33                    ">Hapus</a></td>';
34                echo "</tr>";
35            }
36        } else {
37            echo "<tr>";
38            echo '<td colspan="3">Data NIHIL</td>';
39            echo "</tr>";
40        }
41    $koneksi->close();
42    ?>
43 </table>
```

Penambahan ada pada baris ke-30, yang nanti akan membuka halaman `hapus.php` yang akan melakukan proses penghapusan data pada sistem basis data.

Isi kode dari file `hapus.php` ini adalah seperti berikut :

```

1 <?php
2 include "conf-db.php";
3
4 $koneksi = ConfigDb::getKoneksi();
5
6 if ($koneksi->connect_error) {
7     die("Koneksi ke Basis Data gagal : " . $koneksi->connect_error);
8 }
```

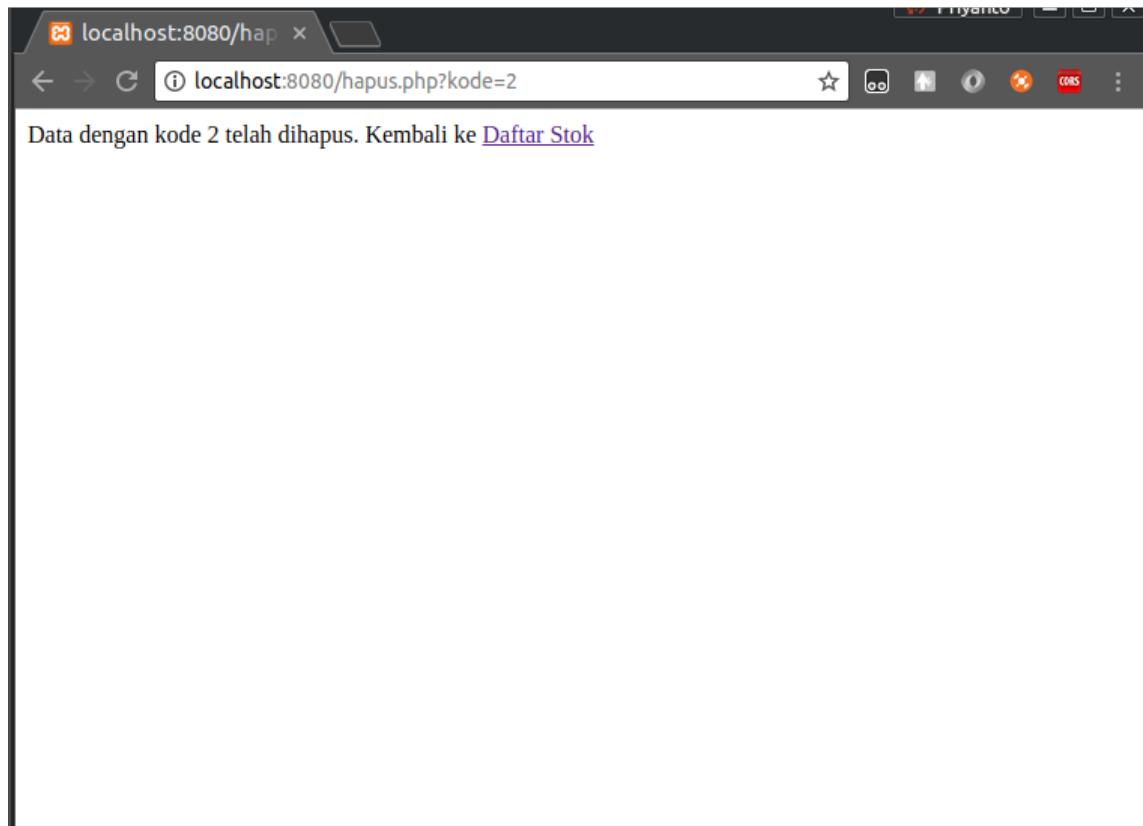
```

10 $query = "delete from stok where kode = " . $_GET["kode"];
11
12 if ($koneksi->query($query) === TRUE) {
13     echo "Data dengan kode " . $_GET["kode"] . ' telah dihapus. Kembali
14     ke <a href='main.php'>Daftar Stok</a>';
15 } else {
16     echo "Error : " . $query . "<br>" . $koneksi->error;
17 }
?>
```

Seperti halaman lainnya yang berinteraksi dengan sistem basis data, kita siapkan koneksinya seperti pada baris ke-2 dan ke-4, kemudian melakukan pemeriksaan koneksi pada blok baris ke-6 sampai dengan ke-8, *query* untuk hapus data disiapkan pada baris ke-10.

Terakhir adalah melakukan eksekusi *query* dan memeriksa eksekusinya, apakah berhasil atau tidak, bila berhasil maka berikan informasi bahwa kode barang tersebut telah dihapus dari sistem basis data, sedangkan apabila gagal berikan informasi kegagalan yang dialami.

Hasil dari kode *hapus.php* di atas akan tampil seperti berikut ini :



Gambar 6.10: Hasil Keluaran Halaman Hapus Data

Sampai disini lengkap sudah aplikasi sederhana yang kita bangun.

6.4 Kesimpulan

Bahwa untuk membangun sebuah aplikasi biasanya diperlukan sebuah sistem basis data untuk menyimpan hasil masukkan data atau pengolahan datanya. Untuk urusan ini, PHP dapat terkoneksi ke sistem basis data (terutama MySQL) melalui 2 (dua) alat, yang pertama sistem *mysqli* yang berikutnya adalah PDO (PHP Data Object).

6.5 Tugas

Buatlah aplikasi sederhana CRUD untuk menyimpan data mahasiswa yang terdiri dari NIM, Nama, dan Jurusan.

Harus menampilkan daftar mahasiswa, memiliki fitur untuk menambahkan data mahasiswa, merubah data mahasiswa, dan menghapus data mahasiswa.

Bab 7

Session

7.1 Tujuan

Memahami dan mampu mengimplementasikan penggunaan *session* di PHP.

7.2 Pengantar

Session adalah sebuah cara untuk menyimpan beberapa variabel untuk digunakan antar halaman *web* yang dibuat. Biasanya *session* dibuat agar *server* tahu kapan seorang *user* memulai aktivitasnya, dan kapan mengakhirinya. Karena keterbatasan protokol *HTTP* yang *stateless*, maka digunakanlah model *session* seperti ini.

Cara kerja dari *session* ini adalah sebetulnya di tiap *browser* yang mengakses, akan dititipkan sebuah *key*, bila *user* melakukan akses ke halaman yang lain, maka akan dilakukan pemeriksaan terlebih dahulu terhadap *key* ini, bila ada, maka *session* yang telah ditetapkan dapat diakses, namun bila tidak ada *browser* akan memulai *session* baru dengan nilai yang masih kosong.

Berbeda dengan *cookie*, bahwa *session* hanya dapat diakses melalui kode *session* yang telah ditempelkan di *browser* melalui *cookie*, yang biasanya akan ada jangka waktunya atau bila *user* telah *logout* dari aplikasi kode *session* ini akan berubah.

Untuk lebih mudahnya, mari kita pelajari kode implementasi *session* di bagian praktek.

7.3 Praktek

7.3.1 Memulai Session dan Membuat Variabel *Session*

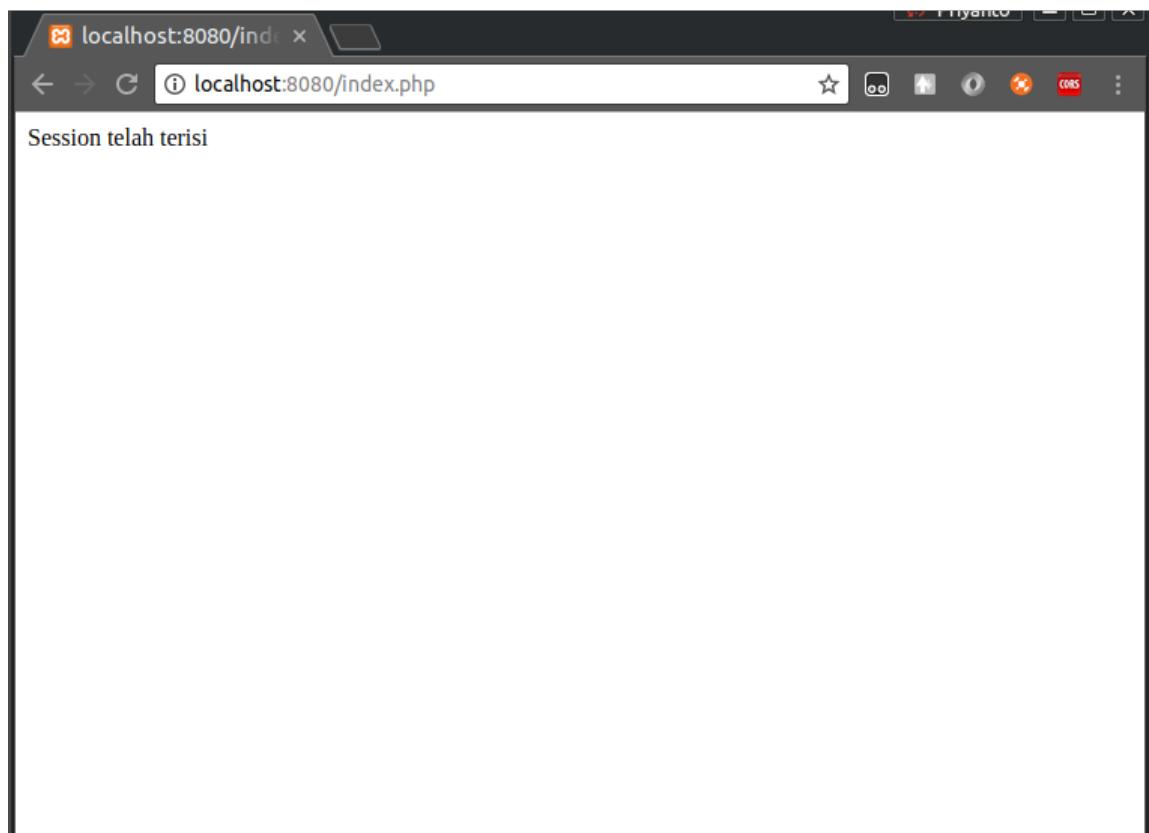
Untuk memulai sebuah *session* caranya cukup mudah , hanya memanggil fungsi `session_start()` sebelum semua fungsi atau skrip dijalankan, setelah itu tempatkan variabel yang akan disampaikan ke halaman lain melalui array `$_SESSION`. Berikut contoh kode untuk menyimpan sebuah nama pada *session*.

```
1 <?php  
2 session_start();
```

```
3 ?>
4
5 <html>
6 <body>
7 <?php
8 $_SESSION["nama"] = "tamami";
9 $_SESSION["jurusan"] = "teknik informatika";
10 echo "Session telah terisi";
11 ?>
12 </body>
13 </html>
```

Tampak bahwa pada baris ke-2 fungsi `session_start()` dijalankan sebelum halaman *HTML* dijalankan. Kemudian pada baris ke-8 dan baris ke-9 , variabel dari *session* terisi dengan nilai masing-masing.

Hasil keluaran untuk kode di atas adalah seperti pada gambar 7.1 berikut ini :



Gambar 7.1: Hasil Keluaran Pembuatan *Session*

7.3.2 Mengakses Variabel *Session*

Untuk mengambil isi dari *session* yang ada, kita harus mengaktifkan atau membuka dahulu dengan perintah `session_start()`, jadi isi dari variabel *session* sebenarnya tidak otomatis dilewatkan ke halaman berikutnya, tetapi dengan aktif kita akan melakukan pembukaan *session* di tiap halaman.

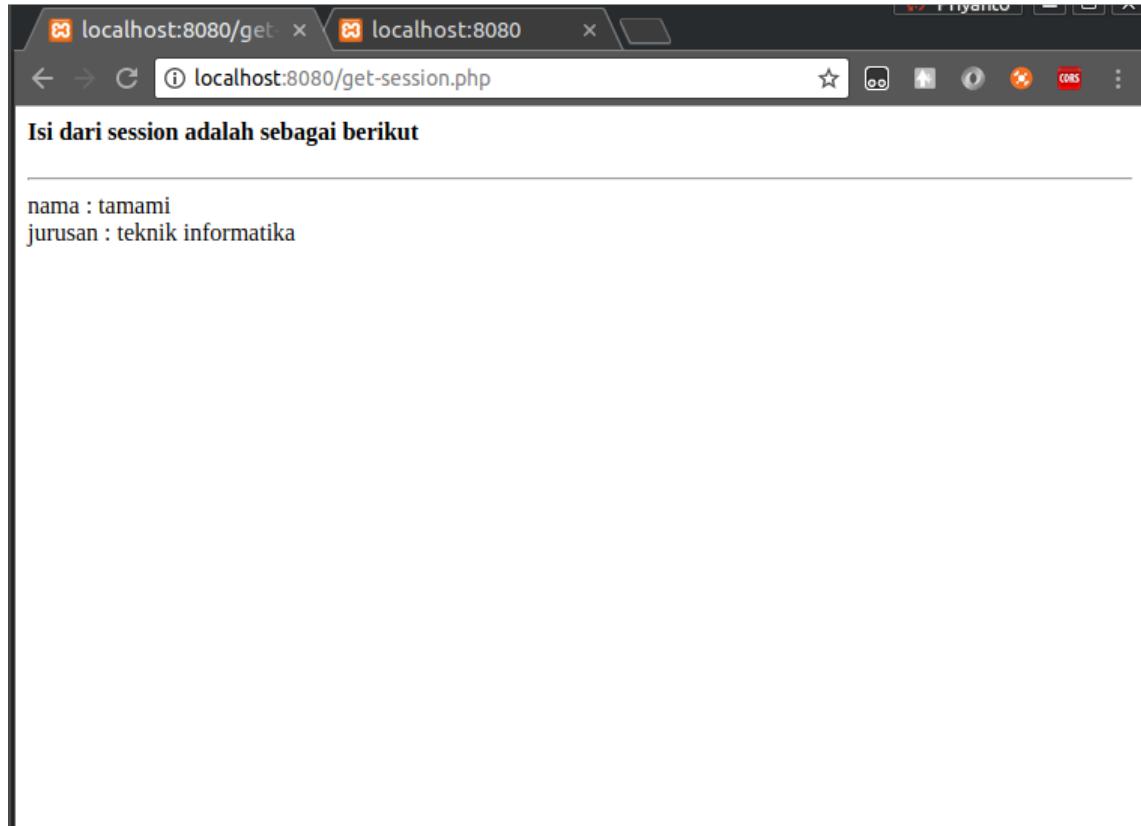
Kode untuk melakukan akses terhadap *session* yang telah terisi di halaman *index.php* adalah seperti berikut :

```
1 <?php
2 session_start();
3 ?>
4
5 <html>
6 <body>
7 <h4>Isi dari session adalah sebagai berikut</h4>
8 <hr>
9 <?php
10 echo "nama : " . $_SESSION["nama"] . "<br>";
11 echo "jurusan : " . $_SESSION["jurusan"];
12 ?>
13 </body>
14 </html>
```

Seperti petunjuk di atas, bahwa tiap halaman harus melakukan pembukaan terhadap *session* maka pada halaman ini pun demikian, pada baris ke-2 kita melakukan pemanggilan fungsi *session_start()*.

Kemudian akses untuk tiap data yang ada pada *session* dipanggil langsung dari *array* *\$_SESSION* seperti pada baris ke-10 dan baris ke-11.

Hasil keluaran dari kode di atas adalah seperti pada gambar 7.2 berikut ini :



Gambar 7.2: Hasil Keluaran Pemanggilan *Session*

7.3.3 Merubah Variabel *Session*

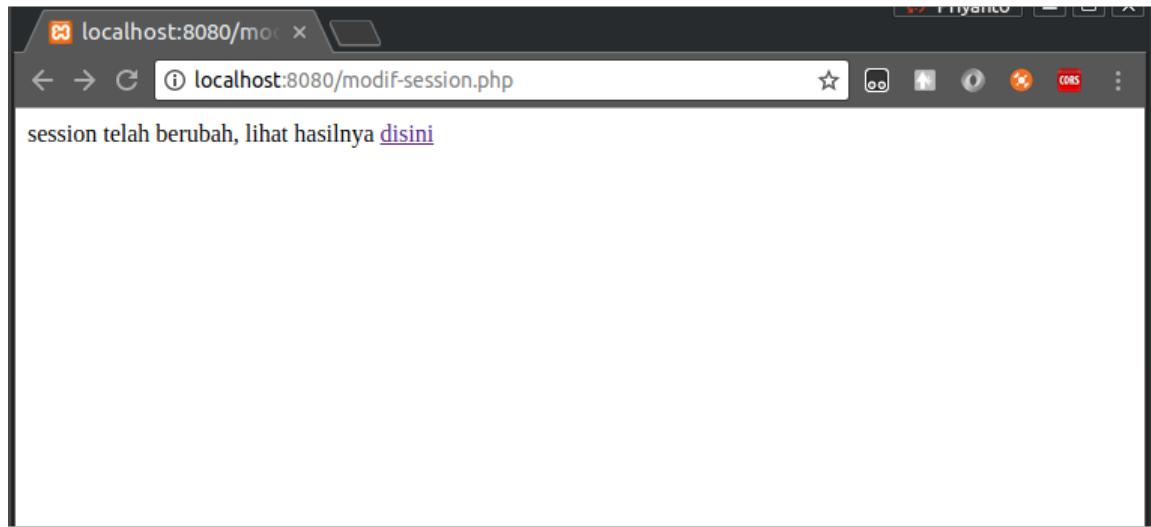
Untuk mengubah isi dari variabel *session* cukup mudah, tinggal ubah saja isi variabel tersebut dengan pemanggilan namanya, contoh kodennya akan lebih mudah untuk menjelaskan seperti berikut ini :

```

1 <?php
2 session_start();
3 ?>
4
5 <html>
6 <body>
7 <?php
8     $_SESSION["nama"] = "mas tamami";
9     echo 'session telah berubah, lihat hasilnya <a href='get-session.
10    php">disini</a>';
11 ?>
12 </body>
13 </html>
```

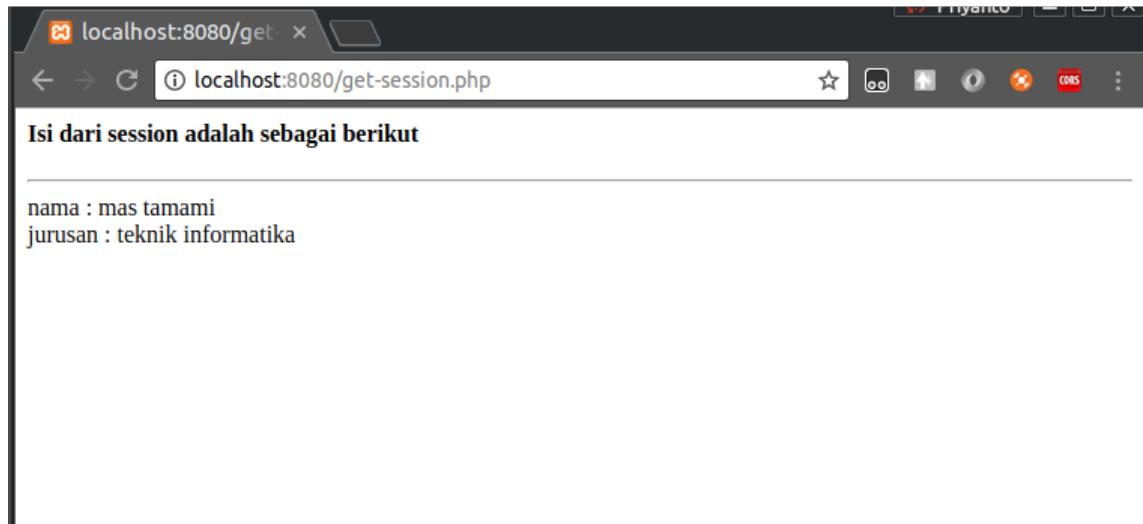
Pada kode tersebut kita mengganti isi variabel *session* `nama` dengan `mas tamami` seperti pada baris ke-8, kemudian akan dicetak ke *browser* informasinya bahwa variabel *session* telah berubah.

Sebelum menjalankan kode ini, terlebih dahulu jalankan kode pada bagian 7.3.1, setelah itu baru akses kode ini melalui *browser*, hasil yang akan dikeluarkan dari kode ini adalah seperti pada gambar 7.3 berikut ini :



Gambar 7.3: Hasil Keluaran Halaman Modifikasi Variabel *Session*

Bila kita mengikuti *link* yang disediakan, kita akan dibawa kembali ke halaman yang mencetak variabel *session* dan melihat perubahan isinya seperti pada gambar 7.4 berikut ini :



Gambar 7.4: Hasil Keluaran Perubahan Variabel *Session*

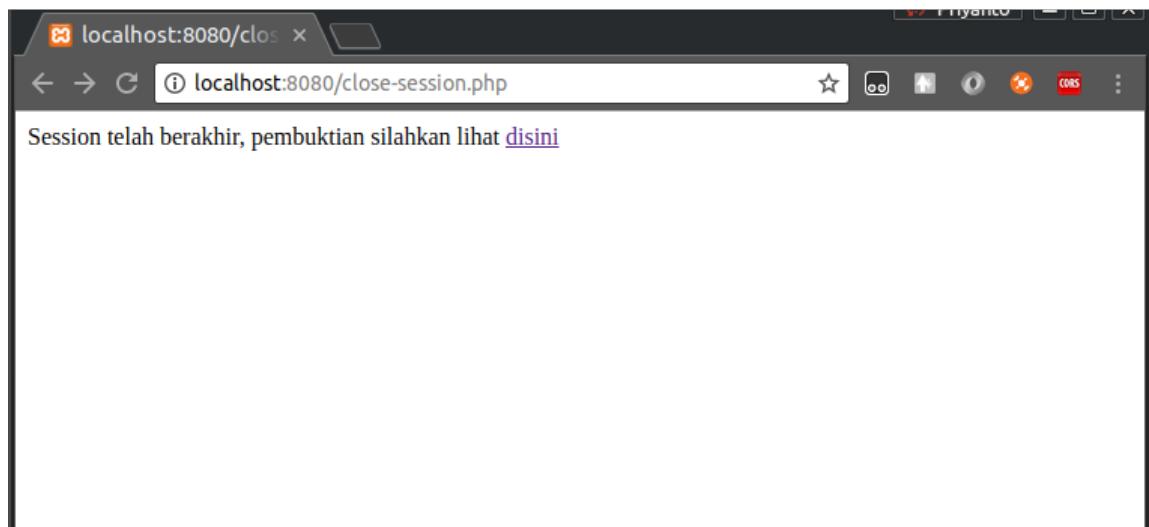
7.3.4 Menutup *Session*

Untuk menutup atau mengakhiri sebuah *session* kita perlu untuk menghapus seluruh variabel *session* dengan memanggil fungsi `session_unset()` kemudian mengakhiri *session* dengan memanggil fungsi `session_destroy()`. Berikut adalah contoh kode untuk melakukan tugas-tugas tersebut :

```
1 <?php
2 session_start();
3 ?>
4
5 <html>
6 <body>
7 <?php
8 session_unset();
9 session_destroy();
10 echo 'Session telah berakhir, pembuktian silahkan lihat <a href="get-
     session.php">disini</a>';
11 ?>
12 </body>
13 </html>
```

Seperti biasa, *session* harus kita aktifkan dengan pemanggilan fungsi seperti pada baris ke-2. Setelah itu halaman ini akan membersihkan variabel *session* pada baris ke-8, dan mengakhiri *session* pada baris ke-9.

Pada baris ke-10, aplikasi akan menampilkan informasi ke *browser* bahwa *session* telah diakhiri. Hasil tampilannya akan terlihat seperti pada gambar 7.5 berikut :



Gambar 7.5: Hasil Keluaran Penutupan Sebuah *Session*

Kemudian apabila kita mengikuti instrukssinya untuk melihat pembuktian bahwa *session* yang kita lalui telah berakhir, kita akan melakukan klik pada *link* [disini](#) kemudian menghasilkan halaman seperti pada gambar 7.6 berikut :



Gambar 7.6: Hasil Keluaran Variabel *Session* Yang Tidak Ada

7.4 Kesimpulan

Karena sifat protokol HTTP yang *stateless*, maka *session* digunakan agar *server* atau aplikasi mengetahui kapan seorang *user* melakukan akses ke aplikasi dan kapan mengakhirinya.

Session pun dapat digunakan untuk menyimpan beberapa variabel yang dapat diakses antar halaman.

7.5 Tugas

Buatlah 2 (dua) buah variabel *session* yang diisi dari *input user*, kemudian datanya ditampilkan di halaman PHP yang lain dengan mengambil data dari *session*.

Bab 8

Penggunaan Form

8.1 Tujuan

Pada akhir bab ini mampu untuk mengimplementasikan bagaimana PHP menangani *form* yang kemudian datanya dapat dioleh sesuai kebutuhan.

8.2 Pengantar

Untuk melakukan pengisian data oleh *user*, maka biasanya akan disediakan *form*. Untuk menangani pengiriman data pada protokol HTTP, yang biasa digunakan adalah metode *get* dan *post*.

Keduanya akan menghasilkan sebuah variabel bertipe data *array* dengan pasangan kunci (*key*) dan isi (*value*) di dalamnya. Perbedaan metode ini adalah apabila kita menggunakan *get* maka data akan dikirimkan melalui *URL*, sedangkan bila menggunakan *post* maka data akan dikirimkan dalam struktur *body* dari metode *post*.

Untuk lebih jelasnya mari kita lihat contoh implementasinya.

8.3 Praktek

8.3.1 Dengan Metode *GET*

Untuk mengirimkan data dengan metode *get*, kita akan mencoba menyiapkan tampilan *form* kemudian membuat kode PHP untuk menerima data dengan metode *get* ini, berikut adalah kode yang dibuat :

```
1 <form action="index.php" method="get">
2 <table>
3     <tr>
4         <td>NAMA</td>
5         <td><input type="text" name="nama"></td>
6     </tr>
7     <tr>
8         <td>BIDANG</td>
9         <td><input type="text" name="bidang"></td>
```

```

10   </tr>
11   <tr>
12     <td>AKSES</td>
13     <td><input type="text" name="akses"></td>
14   </tr>
15   <tr>
16     <td><input type="submit" value="Proses"></td>
17   </tr>
18 </table>
19 </form>
20
21 <?php
22 if(array_key_exists("nama", $_GET) == null && array_key_exists("bidang",
23   , $_GET) == null &&
24   array_key_exists("akses", $_GET) == null) return;
25 echo "Datanya telah diterima sebagai berikut:<br>";
26 echo 'NAMA : ' . $_GET["nama"] . "<br>";
27 echo 'BIDANG : ' . $_GET["bidang"] . "<br>";
28 echo 'AKSES : ' . $_GET["akses"] . "<br>";
29 ?>
```

Langkah awal kita definisikan dahulu *form* dengan tag **form** yang datanya akan dikirim ke *file* ini sendiri, yaitu **index.php** dengan metode *get*.

Kemudian kita menggunakan **table** agar tampilan *form* menjadi lebih rapih. Pembuatan *form* ada pada baris ke-1 sampai baris ke-19.

Proses dengan kode PHP sendiri ada pada baris ke-21 sampai baris ke-29, pada baris ke-22 kita periksa terlebih dahulu isi data dari *variabel* **\$_GET** apakah semua *key* telah terisi data atau belum, bila ada 1 (satu) yang masih kosong, maka program akan berhenti melanjutkan proses.

Dari baris ke-25 sampai dengan baris ke-28, aplikasi akan menampilkan hasil masukkan yang dilakukan oleh pengguna dengan memanggil *array* **\$_GET**.

Isi tampilan dari kode di atas sebelum terisi adalah seperti pada gambar 8.1 berikut :

A screenshot of a web browser window. The address bar shows the URL `localhost:8080/index.php`. The page contains a form with three input fields labeled "NAMA", "BIDANG", and "AKSES", each with a corresponding empty text input box. Below the inputs is a "Proses" button. The browser interface includes standard navigation buttons (back, forward, search) and a toolbar.

Gambar 8.1: Hasil Keluaran *Form* Sebelum Dikirim

Setelah pengisian data dan tombol **Proses** di klik, maka tampilan akan menjadi seperti pada gambar 8.2 berikut :

A screenshot of a web browser window showing the result of a form submission. The address bar now shows the URL `localhost:8080/index.php?nama=tamami&bidang=pendataan...`. The page content includes a message "Datanya telah diterima sebagai berikut:" followed by the submitted data: "NAMA : tamami", "BIDANG : pendataan", and "AKSES : admin". The rest of the page is blank.

Gambar 8.2: Hasil Keluaran Setelah Dikirim Menggunakan Metode *Get*

Terlihat bahwa proses penanganan *form* dengan menggunakan metode *get* akan melalui pengisian parameter pada URL, yaitu pada bagian *address bar* di *browser*.

8.3.2 Dengan Metode *POST*

Pengiriman data atau penanganan *form* dengan metode *post* tidak jauh berbeda dengan metode *get*, perbedaanya adalah bahwa dengan metode *post*, data yang dikir-

imkan akan masuk dalam *body* paket. Mari perhatikan kondisi modifikasi pada *file* yang menggunakan metode *get*, kita ubah jadi menggunakan metode *post* berikut :

```

1 <form action="index.php" method="post">
2 <table>
3     <tr>
4         <td>NAMA</td>
5         <td><input type="text" name="nama"></td>
6     </tr>
7     <tr>
8         <td>BIDANG</td>
9         <td><input type="text" name="bidang"></td>
10    </tr>
11    <tr>
12        <td>AKSES</td>
13        <td><input type="text" name="akses"></td>
14    </tr>
15    <tr>
16        <td><input type="submit" value="Proses"></td>
17    </tr>
18 </table>
19 </form>
20
21 <?php
22 if(array_key_exists("nama", $_POST) == null && array_key_exists("bidang",
23     $_POST) == null &&
24         array_key_exists("akses", $_POST) == null) return;
25
26 echo "Datanya telah diterima sebagai berikut:<br>";
27 echo 'NAMA : ' . $_POST["nama"] . "<br>";
28 echo 'BIDANG : ' . $_POST["bidang"] . "<br>";
29 echo 'AKSES : ' . $_POST["akses"] . "<br>";
?>
```

Perbedaan yang dilakukan tentu saja ada pada deklarasi *form* pada parameter *method*, sekarang menggunakan *post*.

Kemudian pada kode PHP yang memproses data masukkan, variabel *array* yang digunakan bukan lagi *\$_GET* melainkan diganti menjadi *\$_POST*. Hasil keluarannya di *browser* akan tampil sama seperti pada gambar 8.1.

Sedangkan pada saat pengguna menekan tombol *proses*, maka akan tampak perubahannya seperti pada gambar 8.3 berikut ini :

The screenshot shows a browser window with the URL `localhost:8080/index.php`. The page contains a form with three input fields: 'NAMA' (Name), 'BIDANG' (Subject), and 'AKSES' (Access). Below the form is a button labeled 'Proses' (Process). To the right of the form, a message displays the submitted data: 'Datanya telah diterima sebagai berikut:' followed by 'NAMA : tamami', 'BIDANG : pendataan', and 'AKSES : admin'.

Gambar 8.3: Hasil Keluaran Dengan Menggunakan Metode Post

Tampak bahwa informasi data yang diisikan tidak lagi muncul pada *address bar*, tetapi data tetap terkirim dan tetap dapat ditampilkan.

8.4 Kesimpulan

Bahwa penanganan *form* pada bahasa pemrograman PHP dapat dilakukan beberapa cara, cara yang umum adalah dengan 2 (dua) metode *get* dan *post*.

Pada metode *get*, maka data yang akan dikirimkan dilewatkan melalui URL atau *address bar* pada *browser*, hal ini dapat dilakukan untuk data-data yang sifatnya tidak terlalu rahasia dan panjang parameternya tidak terlalu banyak.

Sedangkan pada metode *post*, data akan dikirim dan di tempatkan pada paket *body*, sehingga data-data yang sifatnya rahasia hendaknya dikirim dengan metode ini.

8.5 Tugas

Buatlah sebuah halaman *login* yang terdiri dari *username* dan *password*, kemudian kirimkan datanya ke halaman lain dan tampilkan ucapan selamat datang pada *username* tersebut dengan menyebutkan namanya.

Bab 9

Operasi Aritmatika

9.1 Tujuan

Mampu melakukan operasi aritmatika menggunakan fungsi lanjutan pada bahasa pemrograman PHP.

9.2 Pengantar

PHP sebagai bahasa pemrograman telah memberikan fungsi-fungsi yang cukup banyak berkaitan dengan operasi aritmatika, selain fungsi, PHP pun telah menyediakan beberapa konstanta yang biasa digunakan pada bidang aritmatika seperti *pi*, *euler*, dan banyak lagi.

Langsung saja kita coba beberapa fungsi aritmatika yang biasanya digunakan.

9.3 Praktek

9.3.1 Fungsi abs()

Fungsi ini digunakan untuk mengubah angka pada parameter menjadi nilai absolut atau nilai positif. Berikut adalah contoh kodenya :

```
1 <?php
2 echo abs(4) . "<br>" .
3     abs(-6) . "<br>" .
4     abs(-3.2);
5 ?>
```

Hasil keluaran dari kode tersebut adalah seperti pada gambar 9.1 berikut ini :



Gambar 9.1: Hasil Keluaran Fungsi **abs**

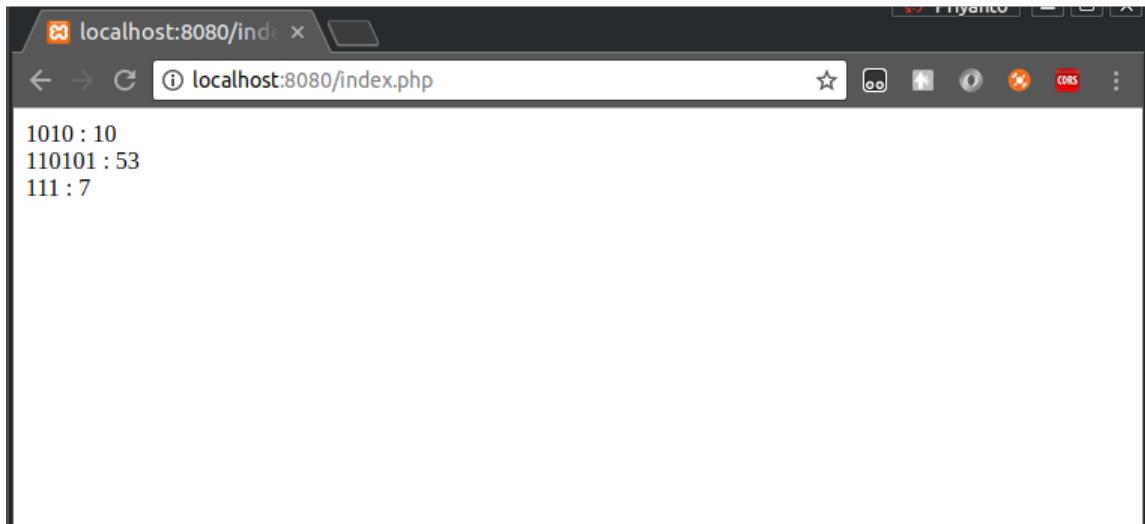
Terlihat bahwa hasil yang ditampilkan adalah bilangan positif dari nilai yang ada pada parameter fungsi **abs()**.

9.3.2 Fungsi bindec()

Fungsi ini digunakan untuk merubah angka biner yang ada pada parameter menjadi bilangan desimal. Contoh kodennya adalah seperti berikut :

```
1 <?php
2 echo "1010 : " . bindec("1010") . "<br>" .
3     "110101 : " . bindec("110101") . "<br>" .
4     "111 : " . bindec("111");
5 ?>
```

Hasil keluaran untuk kode tersebut adalah seperti pada gambar 9.2 berikut ini :



Gambar 9.2: Hasil Keluaran Fungsi `bindec()`

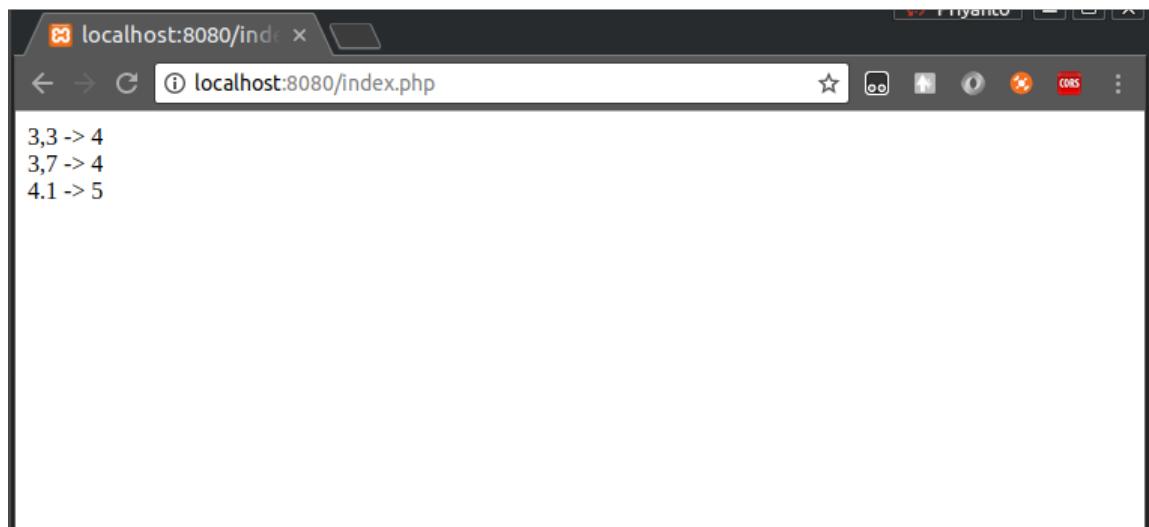
Terlihat bahwa parameter angka biner yang ada pada fungsi `bindec()` akan dikonversi ke bilangan desimal.

9.3.3 Fungsi `ceil()`

Fungsi `ceil()` berfungsi untuk membulatkan angka di belakang koma ke bilangan atas. Artinya, selama ada angka berapapun di belakang koma, maka nilainya akan dibulatkan ke atas. Berikut contoh kode penggunaan fungsi `ceil()` ini :

```
1 <?php
2 echo "3,3 -> " . ceil(3.3) . "<br>" .
3     "3,7 -> " . ceil(3.7) . "<br>" .
4     "4,1 -> " . ceil(4.1);
5 ?>
```

Hasil keluaran dari kode tersebut adalah seperti pada gambar 9.3 berikut ini :



Gambar 9.3: Hasil Keluaran Penggunaan Fungsi `ceil()`

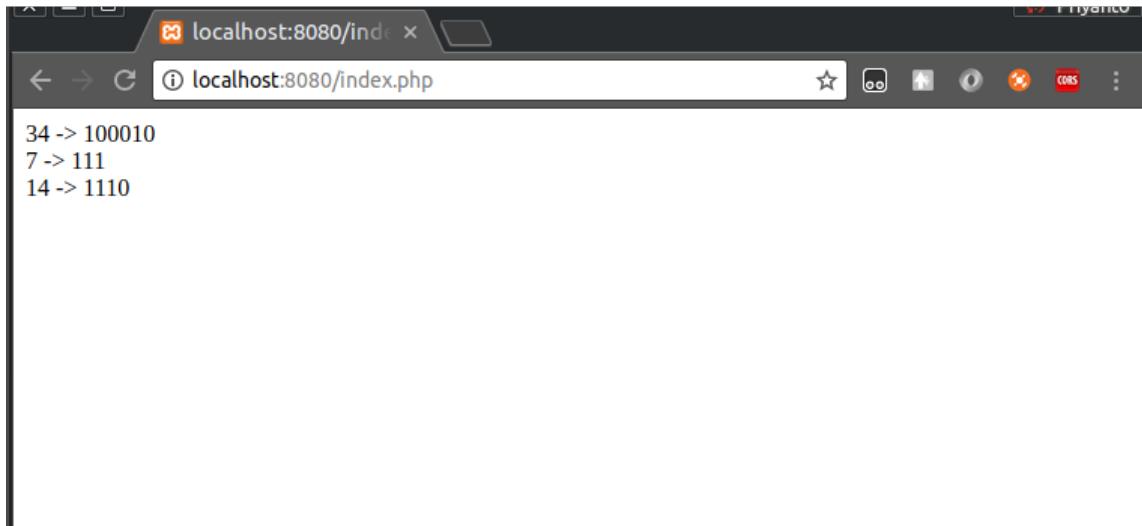
Tampak bahwa apabila muncul angka berapapun di belakang koma, maka angka akan dibulatkan ke atas.

9.3.4 Fungsi `decbin()`

Fungsi ini akan mengubah nilai desimal pada parameter menjadi nilai biner. Contoh penggunaan fungsi ini adalah seperti berikut ini :

```
1 <?php
2 echo "34 -> " . decbin(34) . "<br>" .
3     "7 -> " . decbin(7) . "<br>" .
4     "14 -> " . decbin(14);
5 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 9.4 berikut ini :



Gambar 9.4: Hasil Keluaran Penggunaan Fungsi `decbin()`

Tampak bahwa fungsi `decbin` akan mengubah bilangan desimal pada parameter menjadi bilangan biner.

9.3.5 Fungsi `dechex()`

Seperti fungsi sebelumnya, fungsi `dechex` ini akan mengubah bilangan desimal yang ada pada parameter menjadi bilangan hexa, berikut contoh penggunaannya :

```
1 <?php
2 echo "34 -> " . dechex(34) . "<br>" .
3     "7 -> " . dechex(7) . "<br>" .
4     "14 -> " . dechex(14);
5 ?>
```

Hasil keluaran untuk kode di atas adalah seperti pada gambar 9.5 berikut ini :



Gambar 9.5: Hasil Keluaran Penggunaan Fungsi `decbin()`

9.3.6 Fungsi `decoct()`

Fungsi ini akan mengubah bilangan desimal pada parameter menjadi bilangan dalam oktal. Contoh penggunaan kodennya adalah seperti berikut ini :

```
1 <?php
2 echo "34 -> " . decoct(34) . "<br>" .
3     "8 -> " . decoct(8) . "<br>" .
4     "14 -> " . decoct(14);
5 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 9.6 berikut ini :



Gambar 9.6: Hasil Keluaran Penggunaan Fungsi `decoct()`

9.3.7 Fungsi deg2rad()

Fungsi ini akan mengubah nilai derajat pada parameter menjadi nilai radian. Contoh penggunaannya adalah seperti berikut ini :

```
1 <?php
2 echo "90 -> " . deg2rad("90") . "<br>" .
3     "45 -> " . deg2rad("45") . "<br>" .
4     "170 -> " . deg2rad("170");
5 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 9.7 berikut ini :



Gambar 9.7: Hasil Keluaran Penggunaan Fungsi deg2rad()

9.3.8 Fungsi floor()

Fungsi ini digunakan untuk membulatkan angka di belakang koma ke bilangan bulat dibawahnya. Contoh penggunaan kodennya adalah seperti berikut ini :

```
1 <?php
2 echo "3,7 -> " . floor(3.7) . "<br>" .
3     "2,3 -> " . floor(2.3) . "<br>" .
4     "0,9 -> " . floor(0.9);
5 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 9.8 berikut ini :



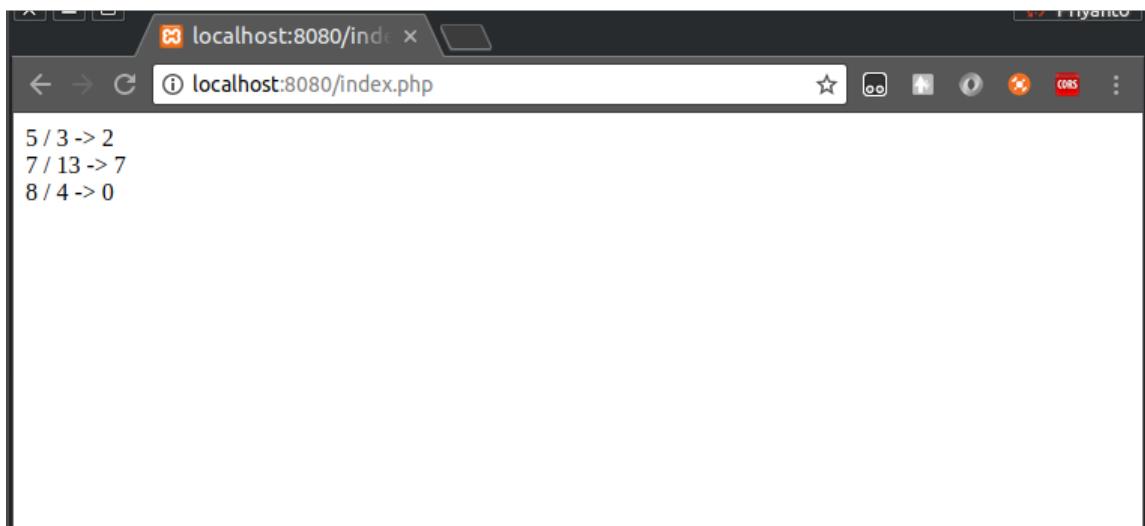
Gambar 9.8: Hasil Keluaran Penggunaan Fungsi `floor()`

9.3.9 Fungsi `fmod()`

Fungsi ini akan menghasilkan nilai sisa hasil bagi dari 2 (dua) bilangan yang menjadi parameternya, contoh penggunaan kodennya adalah seperti berikut ini :

```
1 <?php
2 echo "5 / 3 -> " . fmod(5,3) . "<br>" .
3     "7 / 13 -> " . fmod(7,13) . "<br>" .
4     "8 / 4 -> " . fmod(8,4);
5 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 9.9 berikut ini :



Gambar 9.9: Hasil Keluaran Penggunaan Fungsi `fmod()`

9.3.10 Fungsi getrandmax()

Fungsi ini akan menghasilkan nilai terbesar yang mungkin dihasilkan oleh fungsi `rand()`. Contoh penggunaan kodennya adalah seperti berikut ini :

```
1 <?php
2 echo " hasil -> " . getrandmax() . "<br>" ;
3 ?>
```

Hasil keluaran dari fungsi tersebut adalah seperti pada gambar 9.10 berikut ini :



Gambar 9.10: Hasil Keluaran Penggunaan Fungsi `getrandmax()`

9.3.11 Fungsi hexdec()

Fungsi ini seperti terlihat pada namanya, akan mengubah bilangan hexa pada parameteranya menjadi bilangan desimal. Contoh penggunaan kodennya adalah seperti berikut ini :

```
1 <?php
2 echo "4e -> " . hexdec("4e") . "<br>" .
3     "ca -> " . hexdec("ca") . "<br>" .
4     "10 -> " . hexdec("10");
5 ?>
```

Hasil keluaran dari kode tersebut adalah seperti pada gambar 9.11 berikut ini :



Gambar 9.11: Hasil Keluaran Penggunaan Fungsi `hexdec()`

9.3.12 Fungsi `is_finite()`

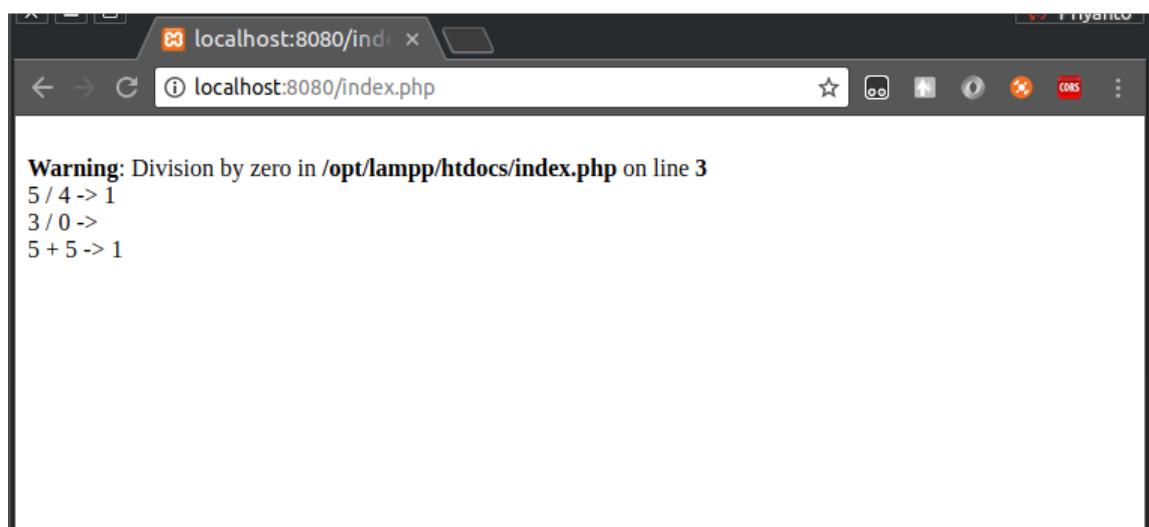
Fungsi ini akan memeriksa apakah sebuah nilai pada parameter masuk dalam jenis angka terhingga, atau tak terhingga. Contoh kodenya adalah seperti berikut ini :

```

1 <?php
2 echo "5 / 4 -> " . is_finite(5/4) . "<br>" .
3     "3 / 0 -> " . is_finite(3/0) . "<br>" .
4     "5 + 5 -> " . is_finite(5 + 5);
5 ?>

```

Hasil keluaran pada kode di atas adalah seperti pada gambar 9.12 berikut ini :



Gambar 9.12: Hasil Keluaran Penggunaan Fungsi `is_finite()`

Hasil pembagian dengan 0 (nol) akan menghasilkan nilai *false* atau kosong, karena masuk dalam kategori nilai tak terhingga, namun tetap muncul peringatan bahwa ada operasi dengan pembagi nilai 0 (nol).

9.3.13 Fungsi `is_infinite()`

Fungsi ini kebalikan dari sebelumnya, yang akan memastikan apakah nilainya masuk dalam kategori tak terhingga atau terhingga. Bila hasil nilai pada parameter adalah tak terhingga, maka akan menghasilkan nilai *true* atau 1 (satu). Contoh kodennya adalah seperti berikut ini :

```

1 <?php
2 echo "5 / 4 -> " . is_infinite(5/4) . "<br>" .
3     "3 / 0 -> " . is_infinite(3/0) . "<br>" .
4     "5 + 5 -> " . is_infinite(5 + 5);
5 ?>

```

Hasil keluaran dari kode di atas adalah seperti pada gambar 9.13 berikut ini :



Gambar 9.13: Hasil Keluaran Penggunaan Fungsi `is_infinite()`

9.3.14 Fungsi `max()`

Fungsi ini akan mencari nilai terbesar dari seluruh parameter yang disertakan, baik bentuknya dalam baris parameter, atau dalam larik. Contoh penggunaannya adalah seperti berikut ini :

```

1 <?php
2 echo "1,9,5,4,7 -> " . max(1,9,5,4,7) . "<br>" .
3     "[2,4,3,8,1] -> " . max(array(2,4,3,8,1)) . "<br>" ;
4 ?>

```

Hasil keluaran dari kode di atas adalah seperti pada gambar 9.14 berikut ini :

Gambar 9.14: Hasil Keluaran Penggunaan Fungsi `max()`

9.3.15 Fungsi `min()`

Fungsi ini sesuai namanya, akan mencari nilai paling kecil dari parameter yang disertakan atau dari larik yang disertakan pada parameternya. Contoh kodenya adalah seperti berikut ini :

```
1 <?php
2 echo "1,9,5,4,7 -> " . min(1,9,5,4,7) . "<br>" .
3     "[2,4,3,8,1] -> " . min(array(2,4,3,8,1)) . "<br>" ;
4 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 9.15 berikut ini :

Gambar 9.15: Hasil Keluaran Penggunaan Fungsi `min()`

9.3.16 Fungsi octdec()

Fungsi ini akan melakukan konversi terhadap nilai oktal yang ada pada parameternya untuk diubah ke sistem desimal. Contoh penggunaan kodennya adalah seperti berikut ini :

```

1 <?php
2 echo "32 -> " . octdec(32) . "<br>" .
3     "17 -> " . octdec(17) . "<br>" .
4     "11 -> " . octdec(11);
5 ?>

```

Hasil keluaran untuk kode di atas adalah seperti pada gambar 9.16 berikut ini :



Gambar 9.16: Hasil Keluaran Penggunaan Fungsi octdec()

9.3.17 Fungsi pow()

Fungsi ini akan menghasilkan pangkat dari parameter pertama di pangkat parameter yang kedua. Contoh kodennya adalah seperti berikut ini :

```

1 <?php
2 echo "3 ^ 2 -> " . pow(3,2) . "<br>" .
3     "4 ^ -2 -> " . pow(4,-2) . "<br>" .
4     "-3 ^ 3 -> " . pow(-3,3);
5 ?>

```

Hasil keluaran dari kode di atas adalah seperti pada gambar 9.17 berikut ini :



Gambar 9.17: Hasil Keluaran Penggunaan Fungsi `pow()`

9.3.18 Fungsi `rad2deg()`

Fungsi ini akan mengubah parameter radian ke bentuk derajat. Contoh penggunaan kodenya adalah seperti berikut ini :

```

1 <?php
2 echo "0.3 -> " . rad2deg(0.3) . "<br>" .
3     "2.7 -> " . rad2deg(2.7) . "<br>" .
4     "pi / 4 -> " . rad2deg(pi() / 4);
5 ?>

```

Hasil keluaran dari kode tersebut adalah seperti pada gambar 9.18 berikut ini :



Gambar 9.18: Hasil Keluaran Penggunaan Fungsi `rad2deg()`

9.3.19 Fungsi rand()

Fungsi ini akan menghasilkan bilangan bulat secara acak. Contoh penggunaannya adalah seperti pada kode berikut ini :

```

1 <?php
2 echo " -> " . rand() . "<br>" .
3     " -> " . rand() . "<br>" .
4     "1 - 100 -> " . rand(1,100) . "<br>" .
5     "1 - 100 -> " . rand(1,100);
6
7 ?>

```

Hasil keluaran dari kode di atas adalah seperti pada gambar 9.19 berikut ini :



Gambar 9.19: Hasil Keluaran Penggunaan Fungsi `rand()`

Pemberian 2 (dua) buah parameter pada fungsi `rand()` ini akan menghasilkan bilangan bulat secara acak tapi di antara 2 (dua) angka tersebut.

9.3.20 Fungsi round()

Fungsi ini akan membulatkan nilai pada parameter ke nilai terdekat berdasarkan skala tertentu. Contoh penggunaan kodenya adalah seperti berikut ini :

```

1 <?php
2 echo " 5,4 -> " . round(5,4) . "<br>" .
3     " 1,8 -> " . round(1.8) . "<br>" .
4     " -5,3 -> " . round(-5.3) . "<br>" .
5     " 5,243 -> " . round(5.243,1);
6 ?>

```

Hasil keluaran untuk kode tersebut adalah seperti pada gambar 9.20 berikut ini :



Gambar 9.20: Hasil Keluaran Penggunaan Fungsi `round()`

Pada pembulatan yang terakhir, kita menentukan banyak digit dibelakang koma pada parameter kedua, sehingga akan dibulatkan ke 1 (satu) angka di belakang koma.

9.4 Kesimpulan

Bahwa PHP sebetulnya mampu digunakan untuk membangun sebuah aplikasi yang membutuhkan perhitungan aritmatika lanjutan, atau pun untuk membangun aplikasi-aplikasi statistik dan geometri karena ketersediaan fungsi aritmatikanya cukup lengkap.

9.5 Tugas

Buatlah aplikasi kalkulator sederhana yang memasukkan hanya 2 (dua) parameter, dengan pilihan operasi dapat dipilih pada *combo box*.

Bab 10

Operasi String

10.1 Tujuan

Agar mampu melakukan operasi *string* / teks pada PHP.

10.2 Pengantar

PHP memiliki fungsi yang disediakan untuk manipulasi *string* atau teks, fungsi-fungsi ini sudah disediakan sebagai *core* atau paket inti dari PHP, jadi tidak perlu *install* sebagai tambahan.

10.3 Praktek

10.3.1 Fungsi bin2hex()

Fungsi ini digunakan untuk mengubah karakter ASCII dari teks menjadi bilangan hexa. Berikut contoh penggunaan kode untuk fungsi ini :

```
1 <?php  
2 echo " test kode -> " . bin2hex("test kode") . "<br>";  
3 ?>
```

Hasil keluaran untuk kode di atas adalah seperti pada gambar 10.1 berikut ini :



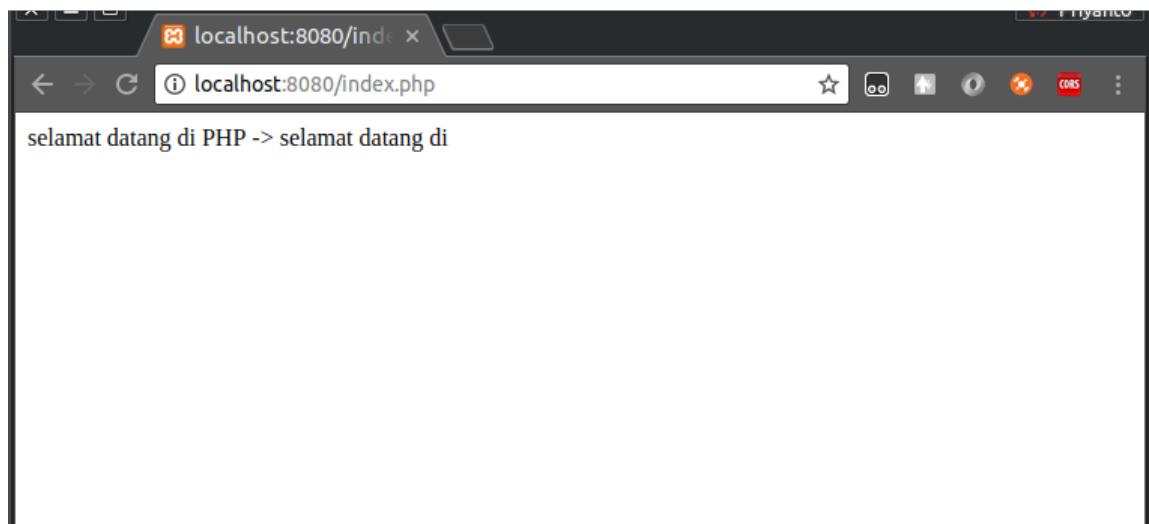
Gambar 10.1: Hasil Keluaran Penggunaan Fungsi `bin2hex()`

10.3.2 Fungsi `chop()`

Fungsi ini akan menghapus karakter dari paling sebelah kanan. Contoh penggunaan fungsi ini adalah seperti berikut :

```
1 <?php
2 echo " selamat datang di PHP -> " . chop(" selamat datang di PHP" , "PHP"
     ) . "<br>" ;
3 ?>
```

Hasil dari kode di atas adalah seperti pada gambar 10.2 berikut ini :



Gambar 10.2: Hasil Keluaran Penggunaan Fungsi `chop()`

10.3.3 Fungsi chr()

Fungsi ini digunakan untuk menghasilkan karakter dari nilai ASCII. Parameter yang dimasukkan dapat berupa angka desimal, oktal, atau hexa seperti kode berikut :

```

1 <?php
2 echo " Desimal 47 -> " . chr(47) . "<br>" .
3     " Oktal 47 -> " . chr(047) . "<br>" .
4     " Hex 47 -> " . chr(0x47);
5 ?>

```

Hasil keluaran dari kode di atas adalah seperti pada gambar 10.3 berikut ini :



Gambar 10.3: Hasil Keluaran Penggunaan Fungsi chr()

10.3.4 Fungsi chunk_split()

Fungsi ini akan menghasilkan teks pada parameter pertama terpotong menjadi beberapa bagian sebanyak bilangan pada parameter kedua, dengan diberikan karakter pemisah pada parameter ketiga. Parameter kedua dan parameter ketiga sebetulnya opsional pada fungsi ini. Berikut contoh penggunaan kodennya :

```

1 <?php
2 echo " selamat datang di PHP -> " . chunk_split(" selamat datang di PHP"
3     , 5, "-") . "<br>";
4 ?>

```

Hasil keluaran untuk kode di atas adalah seperti pada gambar 10.4 berikut ini :



Gambar 10.4: Hasil Keluaran Penggunaan Fungsi `chunk_split()`

10.3.5 Fungsi `count_chars()`

Fungsi ini akan menghasilkan data berupa informasi kemunculan sebuah karakter pada teks yang menjadi parameter. Contoh penggunaannya seperti kode berikut ini :

```

1 <?php
2 $data = "selamat datang di PHP";
3 $result = count_chars($data, 1);
4 echo " selamat datang di PHP -> <br>";
5 foreach($result as $key => $value) {
6     echo chr($key) . " : " . $value . "<br>";
7 }
8 ?>

```

Pada baris ke-3, fungsi `count_chars()` dipanggil dengan mode 1 (satu) seperti pada parameter ke-2, mode ini dapat berisi opsi sebagai berikut :

- 0, secara *default* nilainya adalah 0 (nol) ini, yang akan menghasilkan *array*, dengan nilai ASCII sebagai *key* dan angka kemunculannya sebagai *value*.
- 1, ini sama dengan opsi 0 (nol), tetapi hanya akan menampilkan data yang nilai kemunculannya minimal 1 (satu) kali.
- 2, ini sama dengan opsi 0 (nol), tetapi yang ditampilkan adalah data yang nilai kemunculannya sama dengan 0 (nol), atau tidak pernah muncul sama sekali.
- 3, akan menghasilkan *string* yang isinya adalah huruf-huruf yang dimunculkan secara berbeda, bila hurufnya pernah muncul, maka tidak akan ditampilkan lagi.

- 4, akan menghasilkan *string* yang berisi karakter yang tidak pernah digunakan sama sekali.

Hasil keluaran dari kode di atas adalah seperti pada gambar 10.5 berikut ini :



```
selamat datang di PHP ->
:3
H:1
P:2
a:4
d:2
e:1
g:1
i:1
l:1
m:1
n:1
s:1
t:2
```

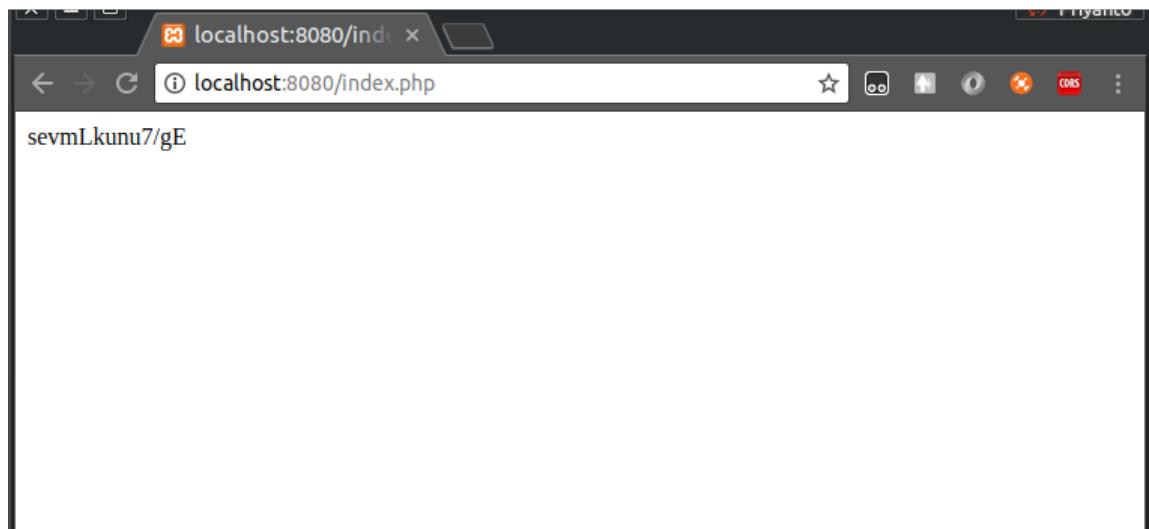
Gambar 10.5: Hasil Keluaran Penggunaan fungsi `count_chars()`

10.3.6 Fungsi `crypt()`

Fungsi ini untuk menghasilkan nilai *hash* dari parameter pertama dengan menggunakan *salt* dari parameter kedua. Contoh penggunaan kodennya adalah seperti berikut ini :

```
1 <?php
2 $data = "selamat datang di PHP";
3 echo crypt($data, 'secret');
4 ?>
```

Hasil keluaran dari kode tersebut adalah seperti pada gambar 10.6 berikut ini :



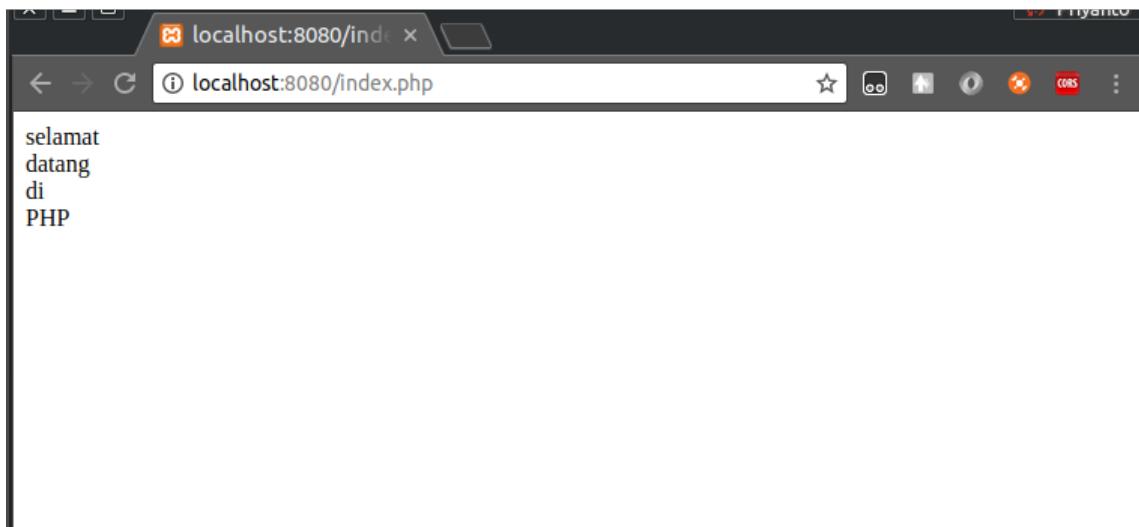
Gambar 10.6: Hasil Keluaran Penggunaan Fungsi `crypt()`

10.3.7 Fungsi `explode()`

Fungsi ini akan menghasilkan *array* dari *string* yang disertakan pada parameter kedua yang dipecah berdasarkan karakter tertentu pada parameter pertama. Contoh kodennya adalah seperti berikut ini :

```
1 <?php
2 $data = "selamat datang di PHP";
3 $result = explode(" ", $data);
4 foreach($result as $item) {
5     echo $item . "<br>";
6 }
7 ?>
```

Hasil keluaran dari kode tersebut adalah seperti pada gambar 10.7 berikut ini :



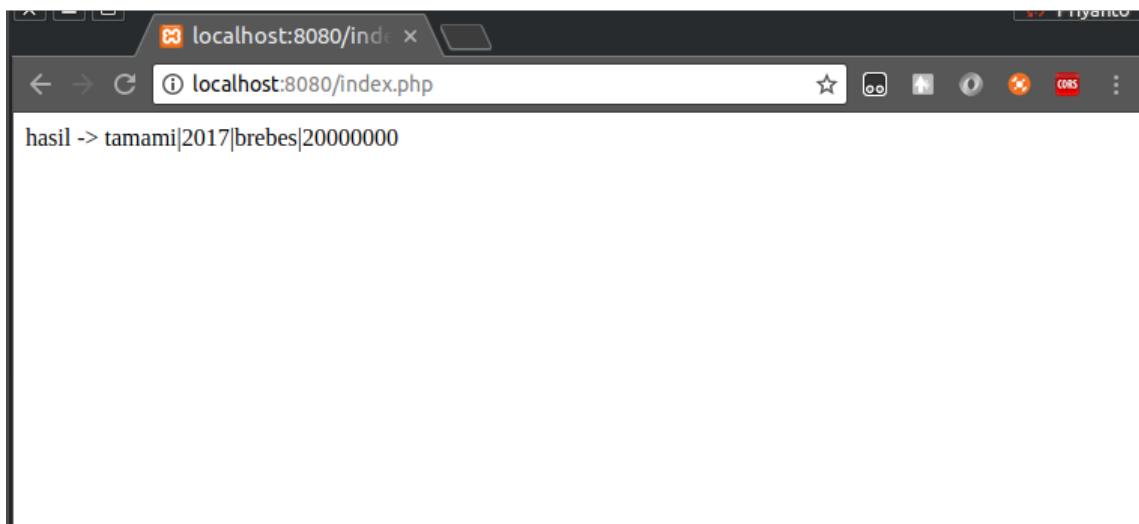
Gambar 10.7: Hasil Keluaran Penggunaan Fungsi `explode()`

10.3.8 Fungsi `implode()`

Fungsi ini kebalikan dari `explode()`, yaitu menghasilkan *string* dari *array* yang disertakan pada parameter kedua, parameter pertama akan berisi karakter pemisah antar data saat menjadi *string*. Contoh kodenya adalah seperti berikut ini :

```
1 <?php
2 $data = array("tamami", "2017", "brebes", 20000000);
3 $result = implode("|", $data);
4 echo "hasil -> " . $result;
5 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 10.8 berikut :



Gambar 10.8: Hasil Keluaran Penggunaan Fungsi `implode()`

10.3.9 Fungsi lcfirst()

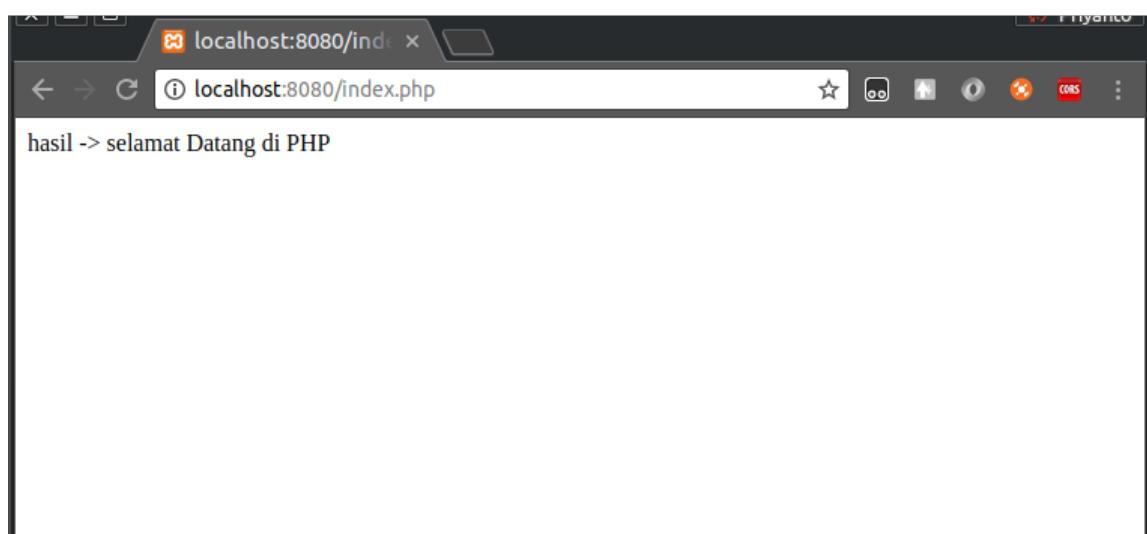
Fungsi ini akan membuat karakter pertama jadi huruf kecil. Contoh penggunaannya adalah seperti kode berikut :

```

1 <?php
2 $data = "Selamat Datang di PHP";
3
4 echo "hasil -> " . lcfirst($data);
5 ?>

```

Hasil keluaran dari kode tersebut adalah seperti pada gambar 10.9 berikut ini :



Gambar 10.9: Hasil Keluaran Penggunaan Fungsi lcfirst()

10.3.10 Fungsi ltrim()

Fungsi ini akan menghapus spasi yang ada pada sebelah kiri teks. Contoh kodennya adalah seperti berikut ini :

```

1 <?php
2 $data = "           Selamat Datang di PHP";
3 echo "sebelum menggunakan ltrim -> " . $data . "<br>";
4 echo "hasil -> " . ltrim($data);
5 ?>

```

Hasil kode tersebut bila ditampilkan pada *browser* tidak akan terlihat bedanya, namun bila kita buka *source code*, hasil yang ditampilkan akan terlihat seperti pada gambar 10.10 berikut ini :



A screenshot of a web browser window titled "view-source:localhost:8080/index.php". The address bar shows "localhost:8080/index.php". The page content displays two lines of text: "sebelum menggunakan ltrim ->" followed by "Selamat Datang di PHP
hasil -> Selamat Datang di PHP". The first line is preceded by a line number "1" and a colon. The browser interface includes standard navigation buttons (back, forward, search) and a toolbar.

Gambar 10.10: Hasil Keluaran Penggunaan Fungsi `ltrim()`

10.3.11 Fungsi `md5()`

Fungsi ini akan menghasilkan *string hash* dari algoritma MD5. Contoh penggunaannya adalah seperti berikut ini :

```
1 <?php
2 $data = "Selamat Datang di PHP";
3 echo " hasil -> " . md5($data);
4 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 10.11 berikut ini :



A screenshot of a web browser window titled "localhost:8080/index.php". The address bar shows "localhost:8080/index.php". The page content displays the text "hasil -> 2281f287b630ef531eac403dfbde2d79". The browser interface includes standard navigation buttons and a toolbar.

Gambar 10.11: Hasil Keluaran Penggunaan Fungsi `md5()`

10.3.12 Fungsi number_format()

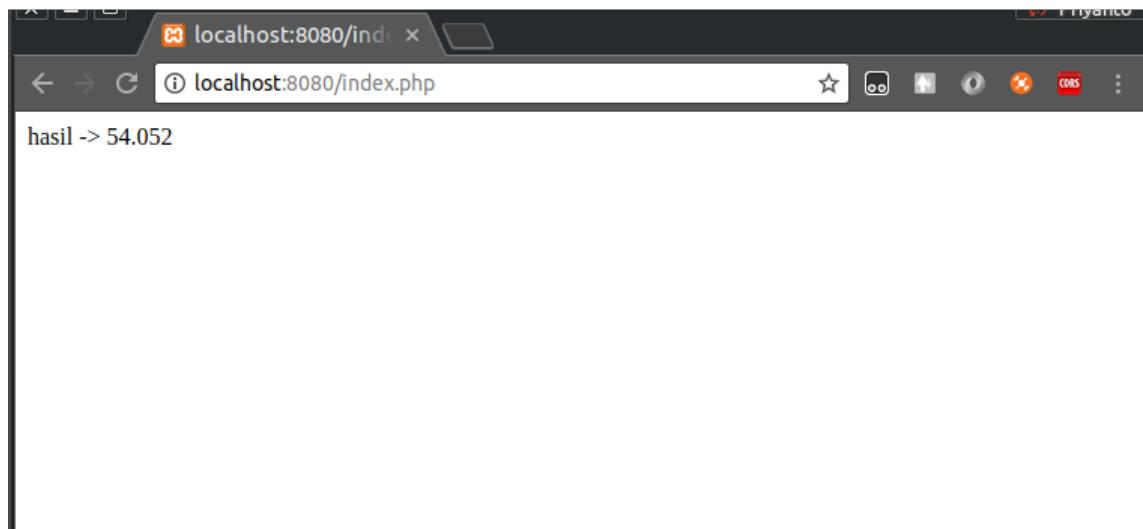
Fungsi ini akan menjadikan angka yang ditampilkan terformat dalam grup ribuan. Contoh kodennya adalah seperti berikut ini :

```
1 <?php
2 echo "hasil -> " . number_format(54052.32, 0, ',', ',');
3 ?>
```

Fungsi ini memiliki 4 (empat) parameter dengan keterangan seperti berikut ini :

- Parameter pertama, adalah angka yang nantinya akan *di format*.
- Parameter kedua, menunjukan jumlah digit di belakang koma (desimal) yang akan ditampilkan
- Parameter ketiga, adalah karakter yang digunakan untuk memisahkan bilangan desimal
- Parameter keempat adalah karakter yang digunakan untuk memisahkan bilangan ribuan ke atas.

Hasil keluaran dari kode tersebut adalah seperti pada gambar 10.12 berikut ini :



Gambar 10.12: Hasil Keluaran Penggunaan Fungsi number_format()

10.3.13 Fungsi rtrim()

Fungsi ini seperti fungsi `ltrim()` sebelumnya, perbedaannya adalah bahwa fungsi ini akan menghilangkan karakter spasi di sebelah kanan teks. Contoh penggunaan kodennya adalah seperti berikut ini :

```
1 <?php  
2 $data = "sebelah kanan teks ini spasi" ;  
3 echo "sebelum -> " . $data . " | ";  
4 echo "sesudah -> " . rtrim($data) . " | ";  
5 ?>
```

Hasil keluaran untuk kode di atas tidak akan terlihat bedanya saat menggunakan *browser* namun saat kita melihat *source code*, kita akan melihat perbedaannya seperti pada gambar 10.13 berikut ini :



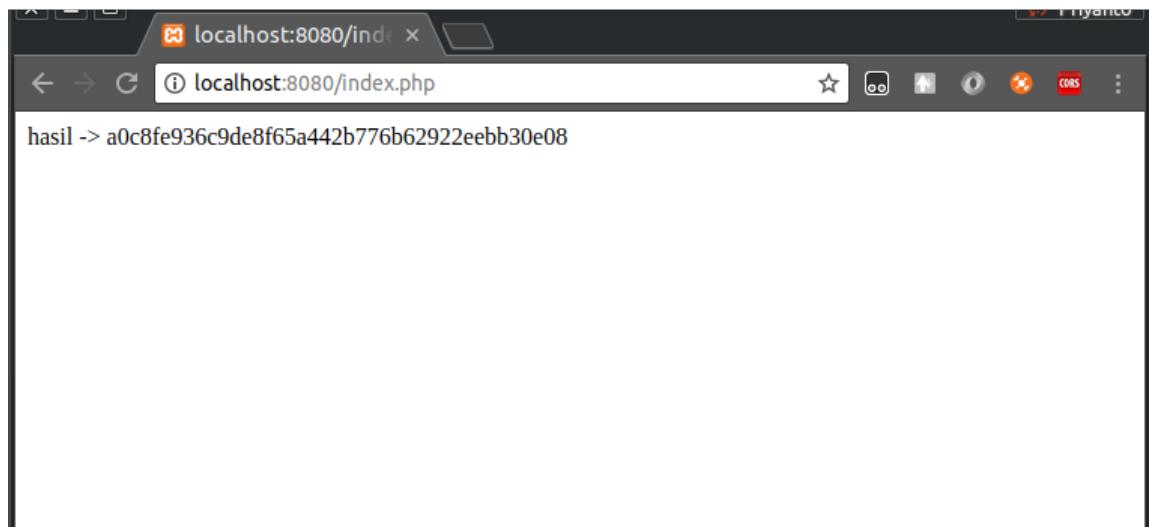
Gambar 10.13: Hasil Keluaran Penggunaan Fungsi `rtrim()`

10.3.14 Fungsi `sha1()`

Fungsi ini akan menghasilkan *string hash* dengan algoritma SHA1 dari parameter yang diberikan. Contoh kodennya adalah seperti berikut ini :

```
1 <?php  
2 $data = "sebelah kanan teks ini spasi";  
3 echo "hasil -> " . sha1($data);  
4 ?>
```

Hasil keluaran untuk kode tersebut adalah seperti pada gambar 10.14 berikut ini :



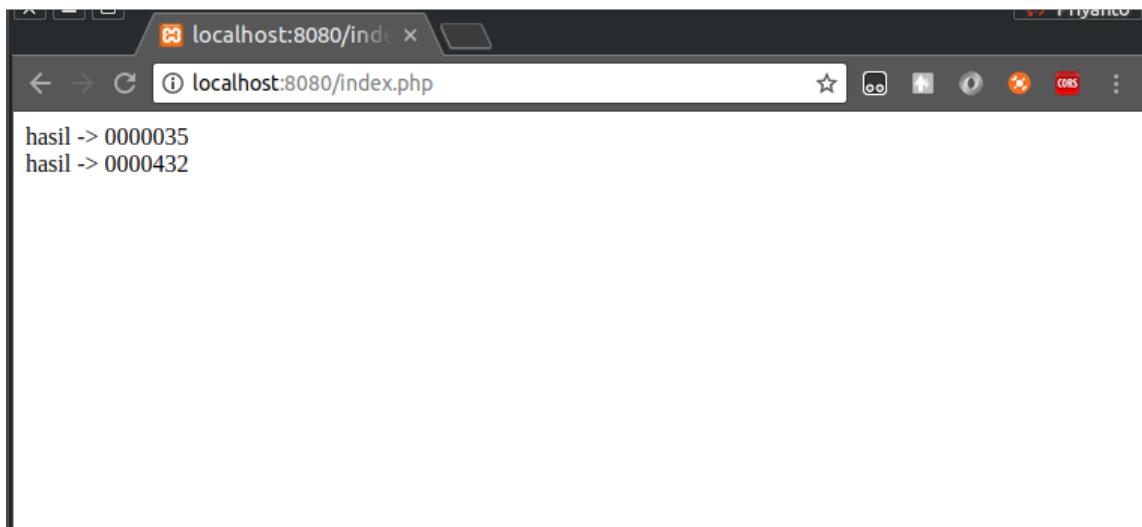
Gambar 10.14: Hasil Keluaran Penggunaan Fungsi `sha1()`

10.3.15 Fungsi `str_pad()`

Fungsi ini akan menghasilkan *string* baru dengan panjang tertentu yang apabila jumlah karakternya kurang, akan diisi dengan karakter yang ditentukan. Contoh penggunaan kodennya adalah seperti berikut ini :

```
1 <?php
2 $data = "35";
3 echo "hasil -> " . str_pad($data, 7, "0", STR_PAD_LEFT) . "<br>";
4 echo "hasil -> " . str_pad("432", 7, "0", STR_PAD_LEFT);
5 ?>
```

Hasil keluaran untuk kode tersebut adalah seperti pada gambar 10.15 berikut ini :



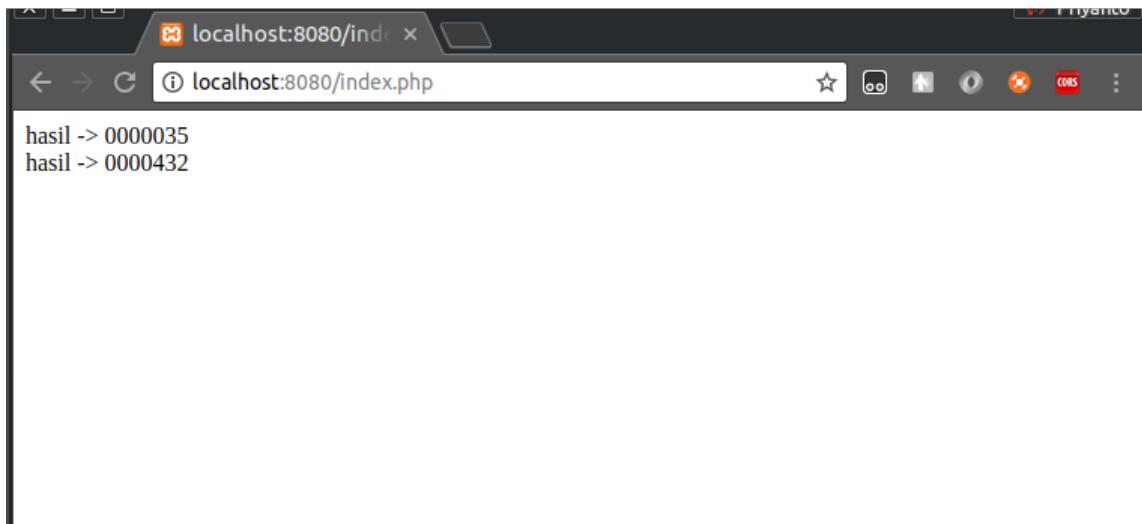
Gambar 10.15: Hasil Keluaran Penggunaan Fungsi `str_pad()`

10.3.16 Fungsi `str_split()`

Fungsi ini akan membagi *string* yang menjadi parameter ke beberapa bagian, hasil dari fungsi ini adalah *array*. Contoh penggunaannya adalah seperti kode berikut ini :

```
1 <?php
2 $data = "35";
3 echo " hasil -> " . str_pad($data, 7, "0", STR_PAD_LEFT) . "<br>";
4 echo " hasil -> " . str_pad("432", 7, "0", STR_PAD_LEFT);
5 ?>
```

Hasil keluaran untuk kode tersebut adalah seperti pada gambar 10.16 berikut :



Gambar 10.16: Hasil Keluaran Penggunaan Fungsi `str_split()`

10.3.17 Fungsi str_word_count()

Fungsi ini akan menghasilkan jumlah kata pada sebuah *string*. Contoh penggunaanya adalah seperti kode berikut ini :

```

1 <?php
2 $data = "selamat datang di PHP";
3 echo "hasil -> " . str_word_count($data) . "<br>";
4 ?>

```

Hasil keluaran dari kode di atas adalah seperti pada gambar 10.17 berikut ini :



Gambar 10.17: Hasil Keluaran Penggunaan Fungsi str_word_count()

10.3.18 Fungsi strlen()

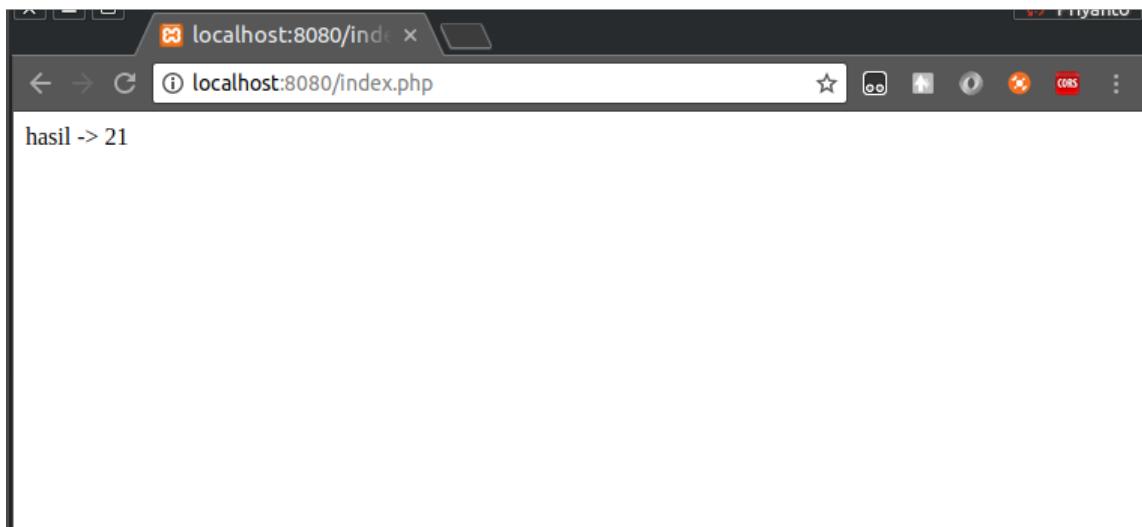
Fungsi ini akan menghasilkan nilai panjang dari *string* yang ada pada parameter. Contoh penggunaan kodennya adalah seperti berikut ini :

```

1 <?php
2 $data = "selamat datang di PHP";
3 echo "hasil -> " . strlen($data) . "<br>";
4 ?>

```

Hasil kode di atas adalah seperti pada gambar 10.18 berikut ini :



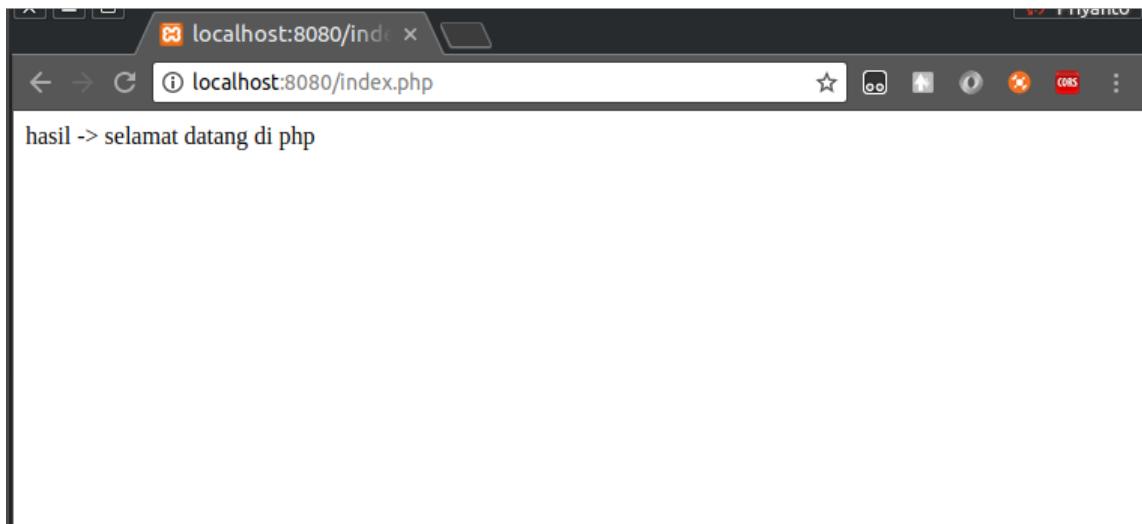
Gambar 10.18: Hasil Keluaran Penggunaan Fungsi `strlen()`

10.3.19 Fungsi `strtolower()`

Fungsi ini akan menghasilkan *string* mengubah data pada parameter yang dimasukkan menjadi karakter huruf kecil seluruhnya. Contoh penggunaan kodennya adalah seperti berikut ini :

```
1 <?php
2 $data = "selamat datang di PHP";
3 echo "hasil -> " . strtolower($data) . "<br>";
4 ?>
```

Hasil keluaran dari kode di atas adalah seperti pada gambar 10.19 berikut ini :



Gambar 10.19: Hasil Keluaran Penggunaan Fungsi `strtolower()`

10.3.20 Fungsi strtoupper()

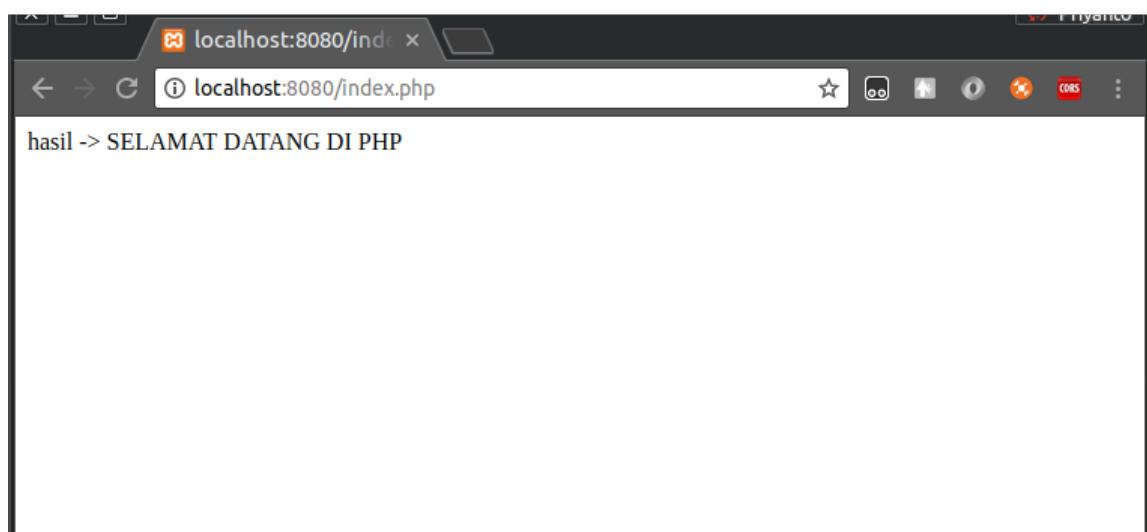
Fungsi ini kebalikan dari fungsi sebelumnya, fungsi ini akan menghasilkan sebuah *string* dengan karakter kapital seluruhnya. Contoh kodenya adalah seperti berikut ini :

```

1 <?php
2 $data = "selamat datang di PHP";
3 echo "hasil -> " . strtoupper($data) . "<br>";
4 ?>

```

Hasil keluaran dari kode di atas adalah seperti pada gambar 10.20 berikut ini :



Gambar 10.20: Hasil Keluaran Penggunaan Fungsi 10-03-020-hasil20

10.3.21 Fungsi trim()

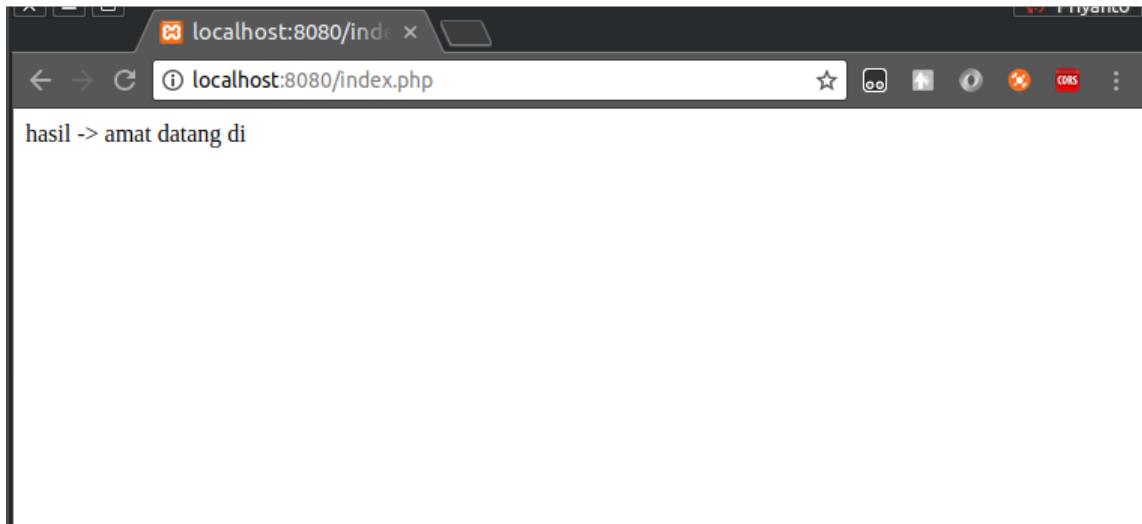
Fungsi ini menghilangkan baris karakter yang secara *default* adalah spasi, namun bisa diubah pada parameter kedua. Contoh penggunaan kodenya adalah seperti berikut ini :

```

1 <?php
2 $data = " selamat datang di PHP ";
3 echo " hasil -> " . trim($data, " selHP") . "<br>";
4 ?>

```

Pada kode di atas, *string* \$data akan dipotong berdasarkan parameter kedua, sehingga bagian sel di awal dan HP di akhir *string* akan dipotong, sehingga hasilnya akan terlihat seperti pada gambar 10.21 berikut ini :



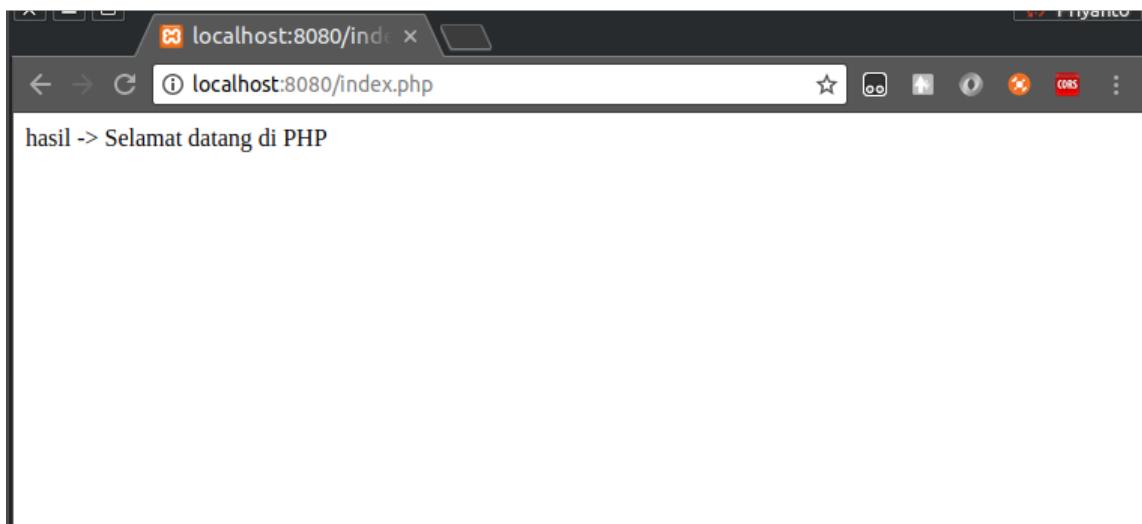
Gambar 10.21: Hasil Keluaran Penggunaan Fungsi 10-03-021-hasil21

10.3.22 Fungsi ucfirst()

Fungsi ini akan mengubah karakter pertama menjadi huruf besar. Contoh penggunaan fungsi ini adalah seperti berikut :

```
1 <?php
2 $data = "selamat datang di PHP";
3 echo "hasil -> " . ucfirst($data) . "<br>";
4 ?>
```

Hasil keluaran untuk kode di atas adalah seperti pada gambar 10.22 berikut ini :



Gambar 10.22: Hasil Keluaran Penggunaan Fungsi ucfirst()

10.3.23 Fungsi ucwords()

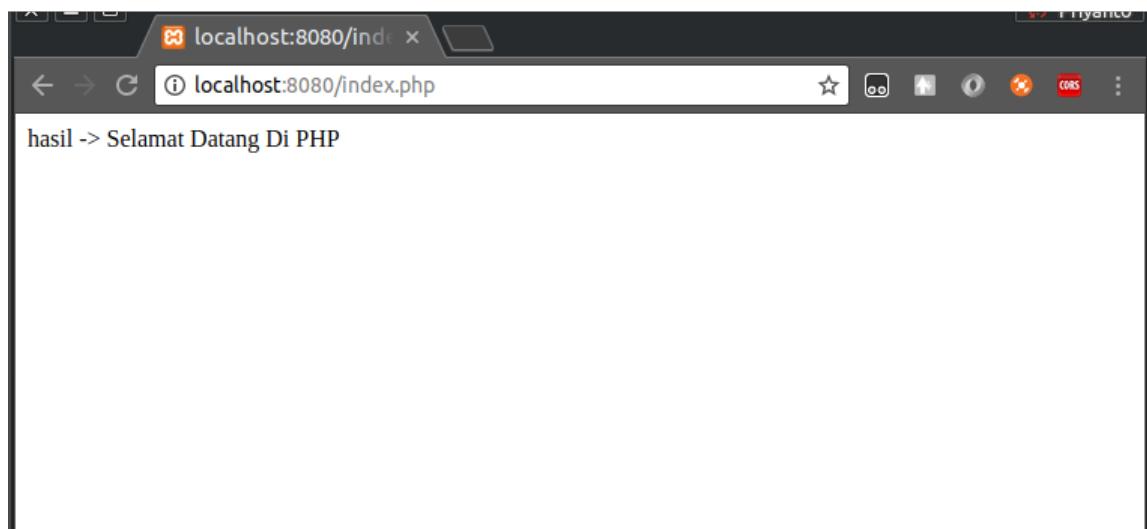
Fungsi ini akan menjadikan seluruh huruf pertama dari tiap kata menjadi huruf besar. Contoh penggunaan kode untuk fungsi ini adalah seperti berikut :

```

1 <?php
2 $data = "selamat datang di PHP";
3 echo "hasil -> " . ucwords($data) . "<br>";
4 ?>

```

Hasil keluaran untuk kode tersebut adalah seperti pada gambar 10.23 berikut :



Gambar 10.23: Hasil Keluaran Penggunaan Fungsi ucwords

10.3.24 Fungsi wordwrap()

Fungsi ini akan memotong *string* pada parameter pertama, untuk panjang yang telah ditentukan pada parameter kedua dan memindahkan ke baris bawahnya dengan memberikan karakter *break* pada parameter ketiga. Contoh kodennya adalah seperti berikut ini :

```

1 <?php
2 $data = "selamat datang di PHP, ini teks paling panjang yang pernah
3 dibuat karena tujuannya untuk " .
4 "membuktikan wrapping";
5 echo "hasil -> " . wordwrap($data, 20, "<br>") . "<br>";
6 ?>

```

Pada kode tersebut, teks yang ada di `$data` akan terpotong setiap 20 karakter, sehingga hasilnya akan terlihat seperti pada gambar 10.24 berikut :



Gambar 10.24: Hasil Keluaran Penggunaan Fungsi `wordwrap()`

10.4 Kesimpulan

Fungsi yang digunakan untuk manipulasi data *string* di PHP sudah cukup melimpah dan terdapat pada *core* program, sehingga tidak perlu modul tambahan lagi untuk menggunakannya. Kegunaan fungsi juga cukup beragam sehingga memudahkan kita untuk membuat aplikasi.

10.5 Tugas

Buatlah sebuah form yang menerima input data berupa angka, nantinya angka ini akan menjadi Nomor Induk Mahasiswa dengan format seperti berikut :

YY 00000

dimana :

- YY adalah 2 (dua) digit dari tahun penerimaan mahasiswa
- 00000 adalah 5 (lima) digit nomor urut penerimaan mahasiswa. Apabila angka yang dimasukkan dalam *form* kurang dari 5 (lima) digit, maka aplikasi akan otomatis melakukan penambahan digit sehingga formatnya menjadi standar dan seragam.

Bab 11

HTML, XHTML, dan DHTML

11.1 Tujuan

Mampu memahami perbedaan implementasi antara HTML, XHTML, dan DHTML.

11.2 Pengantar

HTML sendiri adalah bahasa yang dikenal sebagai standar untuk membangun sebuah halaman *web* yang dapat diakses melalui *browser*, sedangkan XHTML adalah modifikasi dari bahasa HTML yang menggabungkan kemampuan XML dan HTML. Sedangkan DHTML adalah cara pandang baru yang memungkinkan sebuah halaman *web* dapat berinteraksi lebih dinamis dengan pengguna.

DHTML ini bukan sebuah bahasa, melainkan sekumpulan teknologi yang digunakan untuk membangun sebuah aplikasi *web* menjadi lebih interaktif.

11.3 Praktek

11.3.1 HTML

Ini adalah bentuk paling dasar dari halaman sebuah *website* yang dapat kita bangun. Strukturnya sederhana, mari perhatikan kode berikut untuk membuat sebuah halaman :

```
1 <html>
2 <head>
3     <title>Halaman Statis</title>
4 </head>
5 <body>
6     <h2>Selamat datang di HTML</h2>
7     <p>
8         Ini isi dari paragraf
9     </p>
10 </body>
11 </html>
```

Hasil yang muncul untuk kode tersebut adalah seperti pada gambar 11.1 berikut ini :

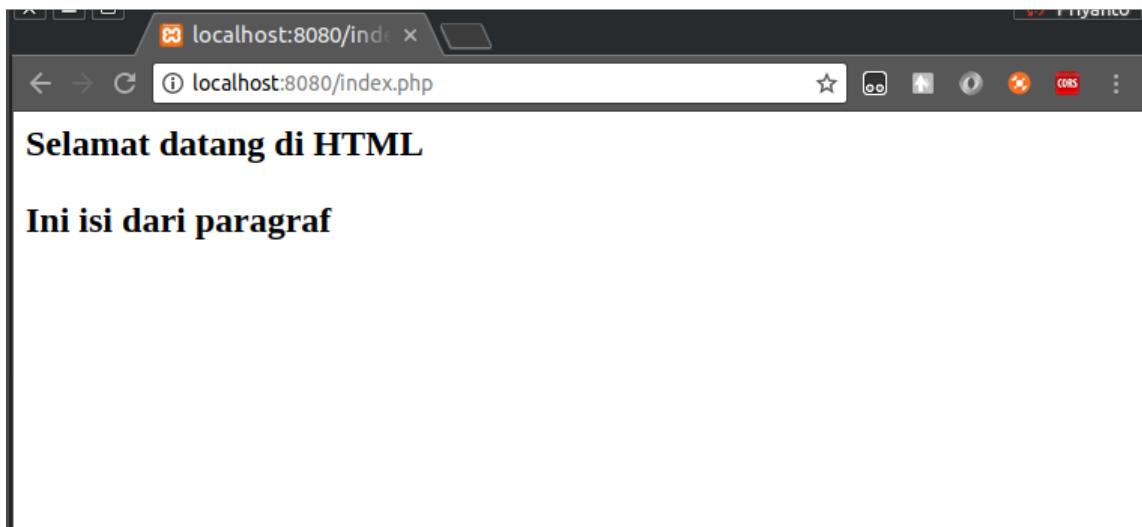


Gambar 11.1: Hasil Keluaran Kode HTML

Namun ada satu yang menjadi masalah adalah pada kode HTML yang sekarang dapat ditampilkan bukan hanya pada *browser* di PC atau Laptop, sekarang aksesnya pun sudah dapat dilakukan melalui *smartphone* seperti Android atau iOS, mari kita lihat apa yang terjadi saat kode di atas kita ubah atau kita hilangkan beberapa bagian kodenya seperti berikut :

```
1 <h2>Selamat datang di HTML
2 <p>Ini isi dari paragraf
```

Maka kodenya pun masih dapat diakses, menjadi seperti pada gambar 11.2 berikut ini :



Gambar 11.2: Hasil Keluaran Kode HTML Yang Buruk

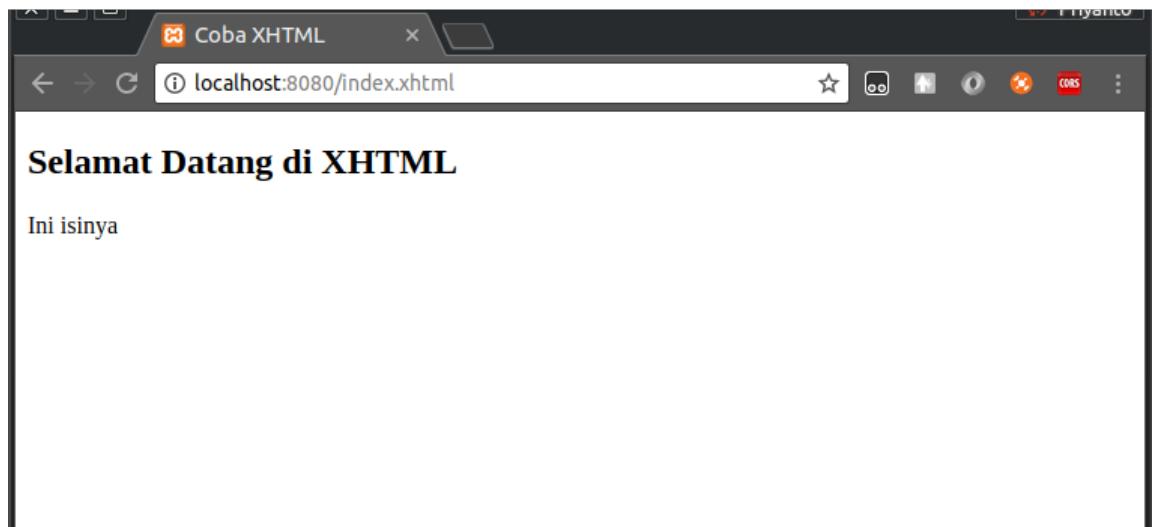
Kode tersebut masih dapat dieksekusi dan berjalan padahal struktur kodennya berantakan, *browser* pada perangkat *smartphone* memiliki keterbatasan sumber daya yang seharusnya tidak perlu untuk memeriksa kode kotor tersebut, maka munculah XHTML yang kodennya lebih ketat dalam penerapan struktur kode.

11.3.2 XHTML

Seperti HTML, namun XHTML ini lebih ketat dalam perlakuan kodennya, perhatikan kode XHTML berikut :

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 " http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3
4 <html xmlns=" http://www.w3.org/1999/xhtml">
5
6 <head>
7   <title>Coba XHTML</title>
8 </head>
9
10 <body>
11   <h2>Selamat Datang di XHTML</h2>
12   <p>Ini isinya a</p>
13 </body>
14
15 </html>
```

Hasil keluaran dari kode tersebut adalah seperti pada gambar 11.3 berikut :

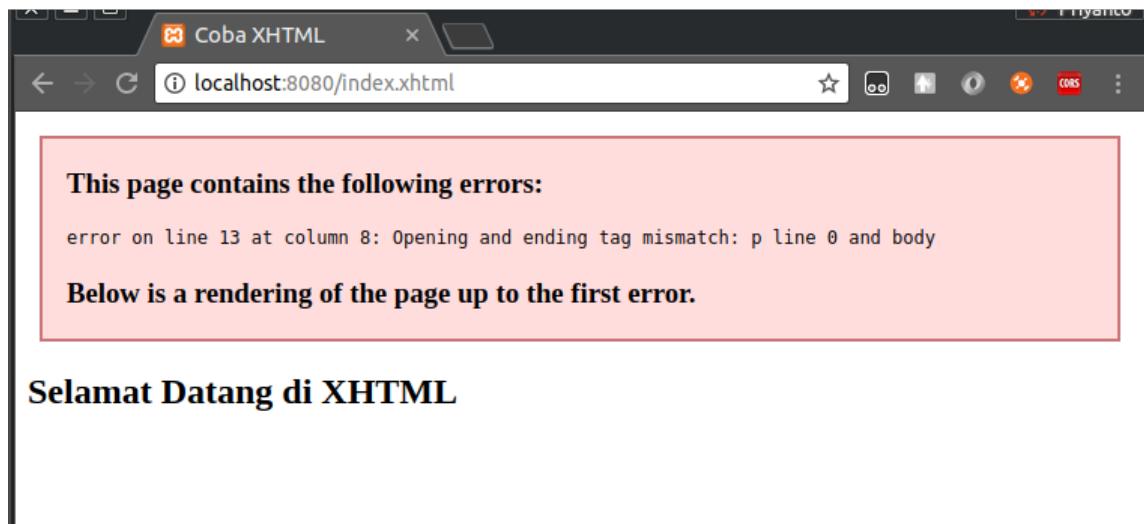


Gambar 11.3: Hasil Keluaran Kode XHTML

Namun bila kita ubah kodennya menjadi seperti ini :

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3
4 <html xmlns="http://www.w3.org/1999/xhtml">
5
6 <head>
7   <title>Coba XHTML</title>
8 </head>
9
10 <body>
11   <h2>Selamat Datang di XHTML</h2>
12   <p>Ini isinya
13 </body>
14
15 </html>
```

Hasil keluarannya akan menampilkan pesan kesalahan seperti pada gambar 11.4 berikut ini :



Gambar 11.4: Hasil keluaran *Bad Code XHTML*

11.3.3 DHTML

Harap diingat bahwa ini bukanlah sebuah bahasa pemrograman, melainkan sebuah teknologi yang memungkinkan sebuah halaman *web* dapat berinteraksi secara langsung dengan pengguna.

Perhatikan kode berikut ini :

```

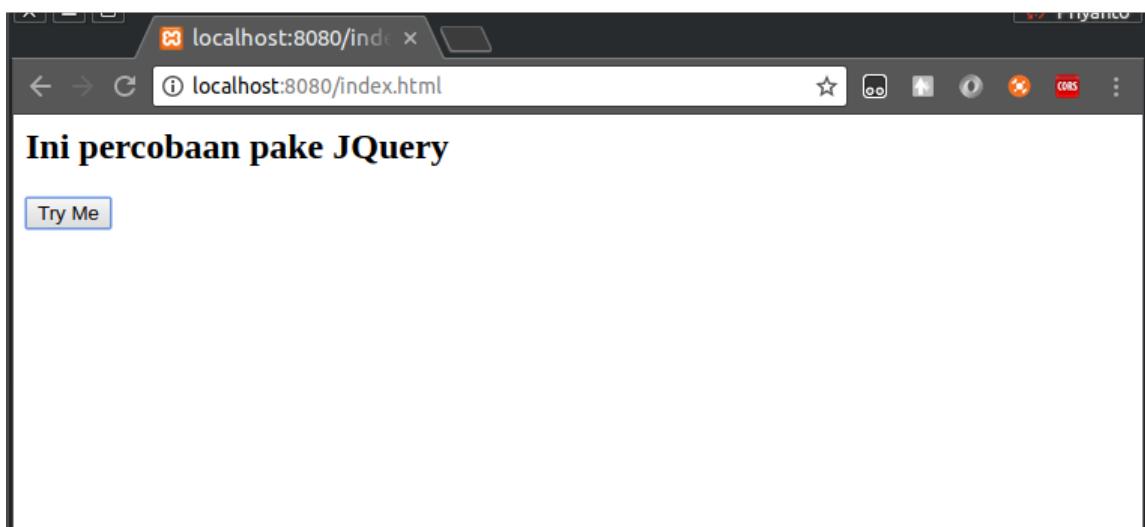
1 <html>
2   <head>
3     <script src="https://ajax.googleapis.com/ajax/libs/jquery
4       /3.3.1/jquery.min.js"></script>
5   </head>
6   <body>
7     <h2>Ini percobaan pake JQuery</h2>
8     <p>Ini teks dalam sebuah paragraf</p>
9     <button>Try Me</button>
10    <script>
11      $(document).ready(function() {
12        $("button").click(function() {
13          $("p").fadeOut("slow");
14        });
15      });
16    </script>
17  </body>
18 </html>
```

Kita melihat ada kode *JavaScript* di atas yang sebetulnya sudah menggunakan *jquery*. Komponen *button* di atas apabila di klik akan melakukan proses *fadeout* untuk komponen *p*. Tampilannya sebelum tombol tersebut di klik adalah seperti pada gambar 11.5 berikut ini :



Gambar 11.5: Hasil Tampilan DHTML Sebelum Tombol Diklik

Namun setelah diklik maka akan terjadi animasi menghilangkan komponen p menjadi seperti pada gambar 11.6 berikut :



Gambar 11.6: Hasil Tampilan DHTML Setelah Tombol Diklik

11.4 Kesimpulan

Bahwa untuk membangun sebuah halaman yang berisi informasi statis mungkin akan lebih sederhana bila menggunakan HTML saja, namun apabila kita ingin membangun sebuah aplikasi *web* yang lebih baik untuk ditampilkan pada perangkat yang lebih kecil seperti *smartphone* mungkin akan lebih baik menggunakan XHTML, sedangkan

apabila diperlukan adanya interaksi yang menarik maka digunakan model DHTML pada halaman *web*.

11.5 Tugas

Dengan menggunakan teknik DHTML, buatlah sebuah permintaan *input* nama kepada pengguna, lalu tampilkan teks "selamat datang, nama" setelah ditekan sebuah tombol di halaman tersebut.

Bab 12

CSS

12.1 Tujuan

Mampu mengimplementasikan CSS untuk merapihkan tampilan sebuah halaman *web*.

12.2 Pengantar

CSS atau dengan kata panjangnya adalah *Cascading Style Sheets* dimana kode di-dalamnya akan melakukan perubahan pengaturan terhadap tampilan sebuah elemen HTML.

Kode CSS sendiri sebetulnya dapat berada pada sebuah *file* HTML, atau berada di *file* terpisah sehingga memudahkan untuk mengaturan tampilannya.

Kode CSS sendiri sebetulnya hanya terdiri dari *selector*, yaitu deklarasi komponen yang akan diatur propertinya, dan bagian deklarasi. Untuk deklarasi sendiri akan terdiri dari nama properti dan isinya.

Penggunaan CSS sendiri sebetulnya sangat banyak sekali untuk dijabarkan dalam satu bagian dari buku ini, kita akan mencoba beberapa hal pada praktek kali ini.

12.3 Praktek

Contoh sederhana dari pembuatan CSS ini adalah seperti kode berikut :

```
1 <h2 style="color: red;">Selamat Datang di PHP</h2>
2 <p>Isinya ada di sini</p>
```

Hasil keluaran untuk kode di atas adalah seperti pada gambar 12.1 berikut :



Gambar 12.1: Hasil Keluaran CSS Dalam Elemen

CSS sendiri dapat dibuat dalam blok tersendiri pada satu halaman HTML, kode di atas dapat kita modifikasi menjadi seperti berikut :

```
1 <style>
2 h2 {
3     color: red;
4 }
5
6 p {
7     color: yellow;
8     background-color: blue;
9 }
10</style>
11<h2>Selamat Datang di PHP</h2>
12<p>Isinya ada di sini</p>
```

Hasil keluaran untuk kode di atas adalah seperti gambar 12.2 berikut ini :



Gambar 12.2: Hasil Keluaran Blok CSS

Pada kode di atas, `h2` kita sebut sebagai *selector*, sedangkan `color` adalah properti, dan `red` adalah *value* atau nilainya.

Untuk *selector* sendiri kita dapat menggantinya, bukan hanya menggunakan nama elemennya, tetapi bisa menggunakan *id* atau *class*, atau *selector* kita masukkan dalam sebuah grup. Contoh kodennya adalah seperti berikut ini :

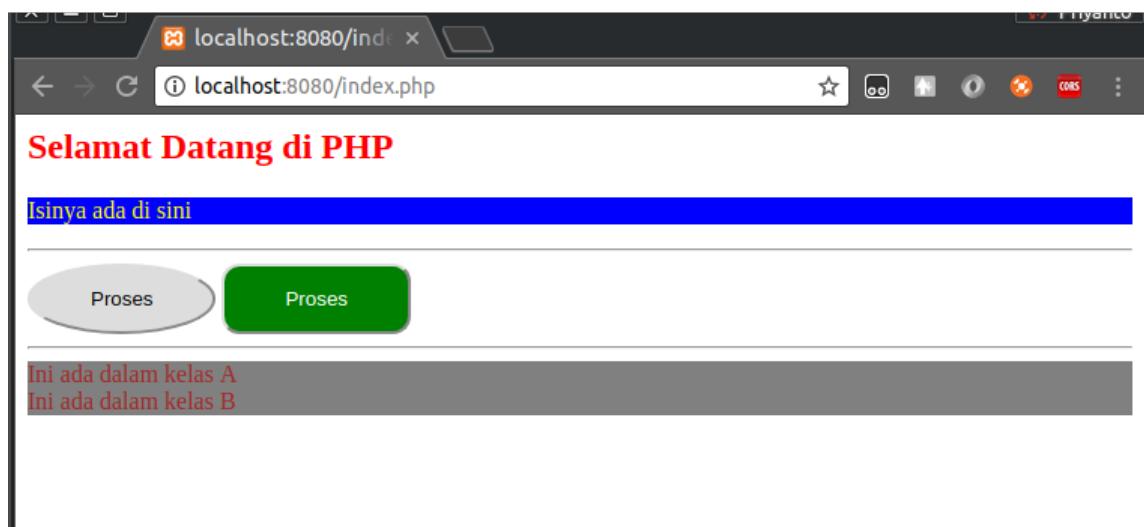
```
1 <style>
2 h2 {
3     color: red;
4 }
5
6 p {
7     color: yellow;
8     background-color: blue;
9 }
10
11 button {
12     padding: 14px 40px;
13 }
14
15 #pakeId {
16     border-radius: 50%;
17 }
18
19 .pakeClass {
20     border-radius: 12px;
21     background-color: green;
22     color: white;
23 }
24
25 .kelasA, .kelasB {
26     background-color: grey;
27     color: brown;
```

```

28 }
29 </ style>
30 < h2 > Selamat Datang di PHP < / h2 >
31 < p > Isinya ada di sini < / p >
32 < hr >
33 < button id = " pakeId " > Proses < / button >
34 < button class = " pakeClass " > Proses < / button >
35 < hr >
36 < div class = " kelasA " > Ini ada dalam kelas A < / div >
37 < div class = " kelasB " > Ini ada dalam kelas B < / div >

```

Hasil keluaran kode tersebut adalah seperti pada gambar 12.3 berikut ini :



Gambar 12.3: Hasil Keluaran Contoh *Selector*

12.4 Kesimpulan

Dengan menggunakan CSS, kita dapat membangun sebuah halaman dengan lebih menarik dan lebih efisien, karena kita dapat mengelola bentuk atau pengaturan tampilan ada pada satu tempat blok atau bahkan pada file terpisah.

12.5 Tugas

Buatlah sebuah tabel yang terdiri dari 3 (kolom) dengan jumlah data minimal 3 (3 baris), setelah itu gunakan CSS untuk merubah tampilannya dengan ketentuan seperti berikut ini :

- *Background* nama kolom diubah
- Setiap baris dengan kelipatan ganjil memiliki warna *background* yang berbeda dengan baris genap.

Bab 13

Production

13.1 Tujuan

Mampu untuk melakukan *publish* aplikasi *web* ke *server hosting*.

13.2 Pengantar

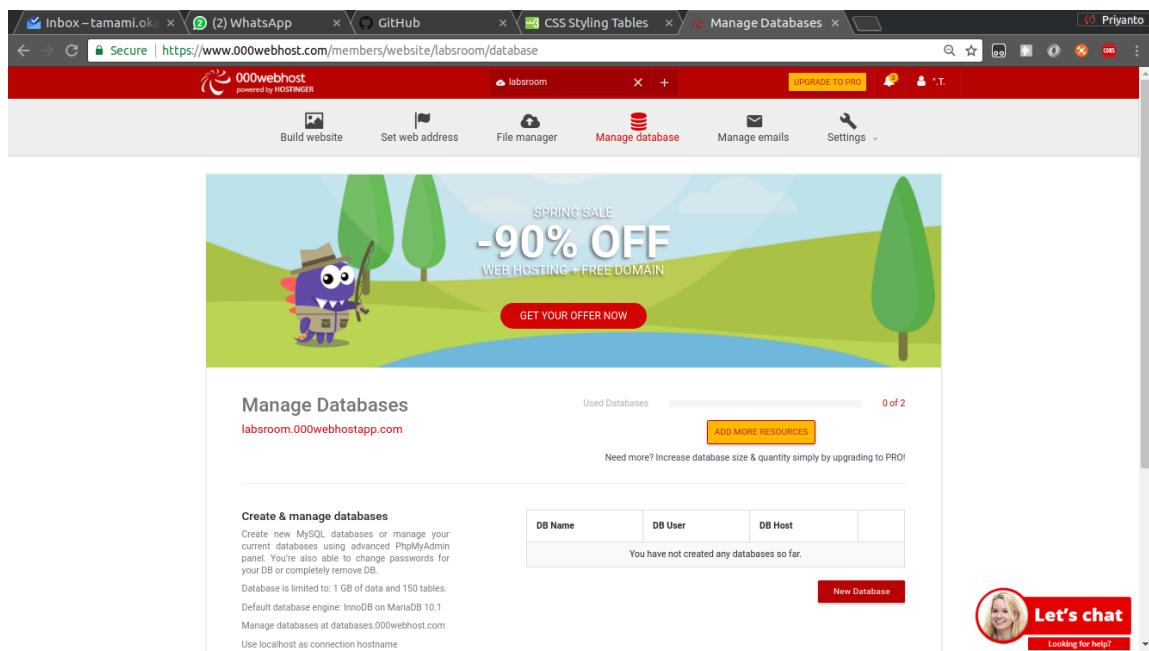
Pada saat kita ingin melakukan *publish* aplikasi *web* yang telah dibuat, kita memerlukan tempat untuk menaruh aplikasi kita yang dapat diakses oleh publik. Tempat ini biasa kita kenal dengan *web hosting*, tentunya apabila kita membangun sebuah aplikasi dengan PHP dan basis data MySQL, kita membutuhkan tempat *hosting* yang mendukung kedua sistem ini.

Tempat *hosting* sendiri ada yang menyediakannya secara gratis, ada pula yang berbayar, kali ini kita akan coba untuk melakukan unggah *project* ke tempat *hosting* yang gratis saja, yaitu di www.000webhost.com.

13.3 Praktek

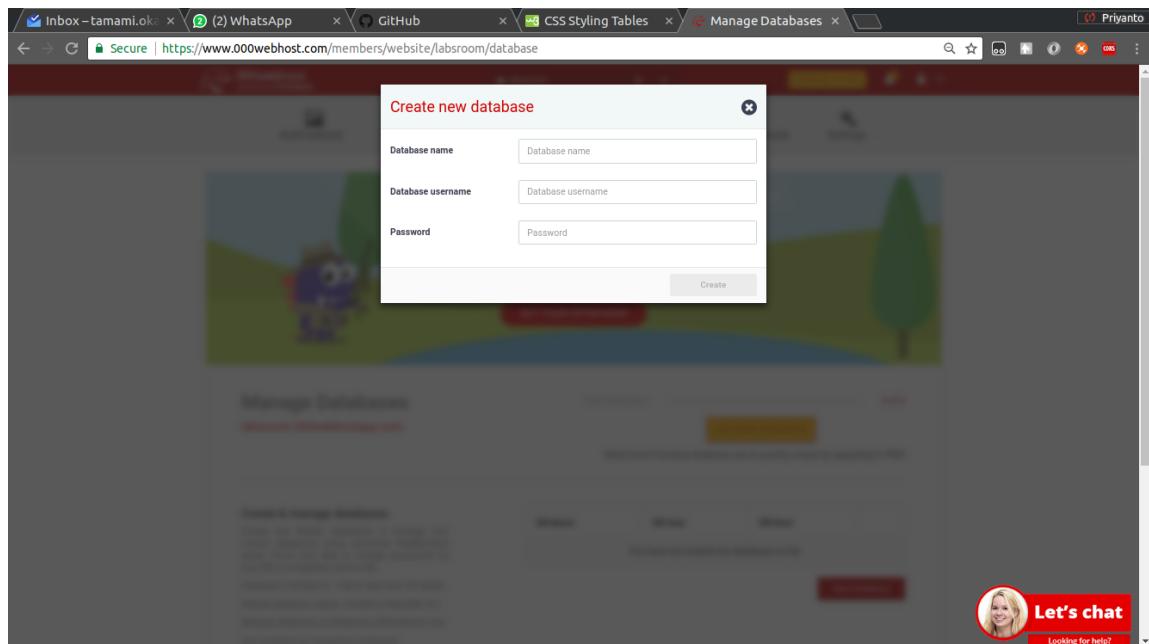
Pada bagian awal dari modul ini kita telah melakukan registrasi dan membuat sebuah alamat *web* pada 000webhost.com, sekarang kita coba unggah *project* CRUD yang telah kita buat dengan langkah seperti berikut ini :

1. Membuat tabel pada 000webhost.com terlebih dahulu, *project* yang akan diunggah kali ini adalah aplikasi data buku. Setelah *login* kita memilih menu **Manage Database** di bagian atas halaman sehingga muncul halaman berikut :



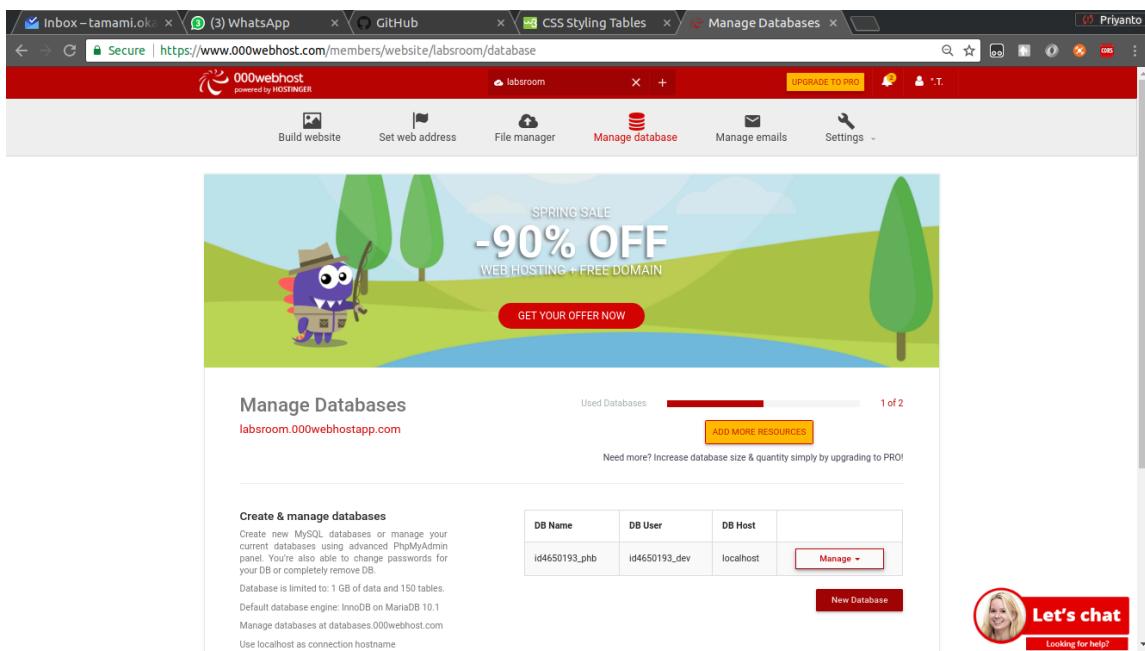
Gambar 13.1: Halaman Konfigurasi Basis Data 000webhost.com

2. Pada bagian kanan bawah, ada tombol untuk membuat basis data baru, mari klik tombol **New Database** sehingga tampil *pop-up* seperti pada gambar 13.2 berikut :



Gambar 13.2: *Pop-up* Membuat Basis Data

3. Isikan datanya, kemudian tekan tombol **Create**, sehingga nantinya akan muncul informasi basis data yang terbentuk seperti pada gambar 13.3 berikut :



Gambar 13.3: Hasil Pembuatan Basis Data

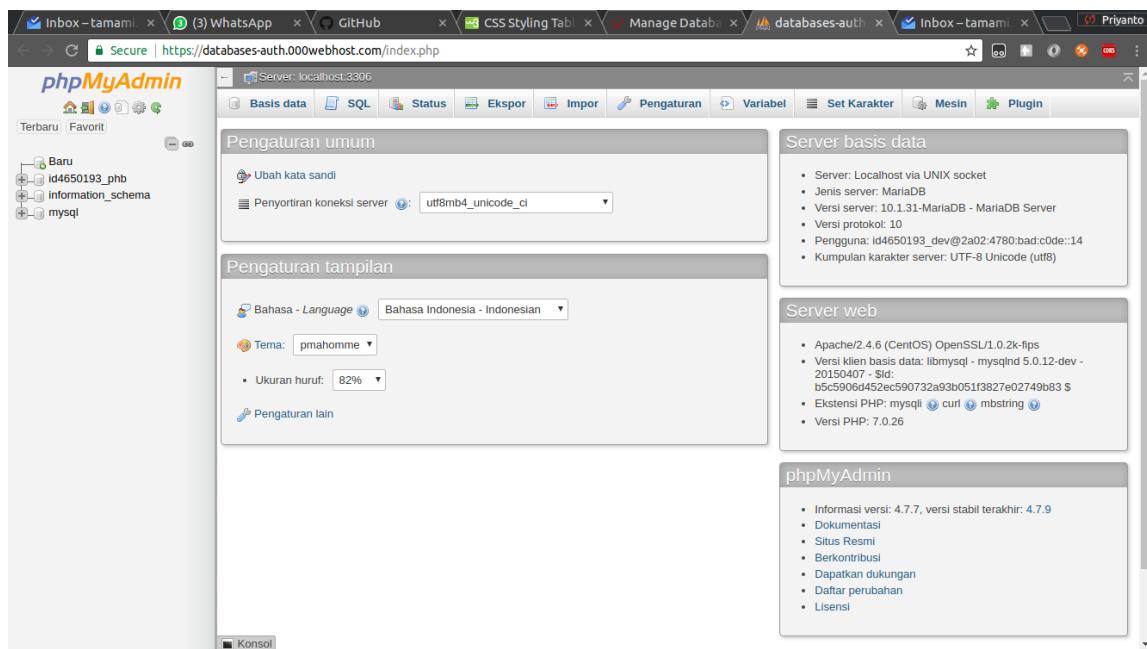
4. Ubah isi *file* konfigurasi basis data kita dengan keterangan yang diberikan oleh 000webhost.com sehingga kodennya menjadi seperti berikut ini :

```

1 <?php
2 $servername = "localhost";
3 $username = "id4650193_dev" ;
4 $password = "rahasia";
5 $db = "id4650193_phb";
6 ?>

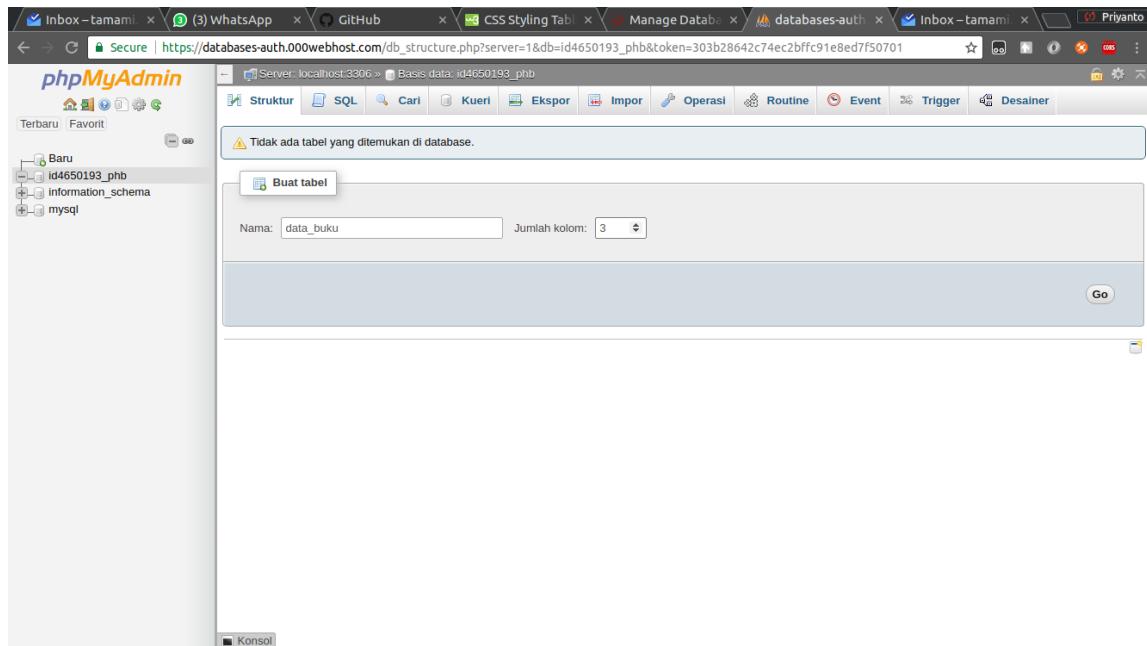
```

5. Selanjutnya kita akan membuat tabel pada server 000webhost.com, klik tombol **Manage**, kemudian pilih **PhpMyAdmin** sehingga akan ditampilkan halaman seperti pada gambar 13.4 berikut :



Gambar 13.4: Halaman PHPMyAdmin di 000webhost.com

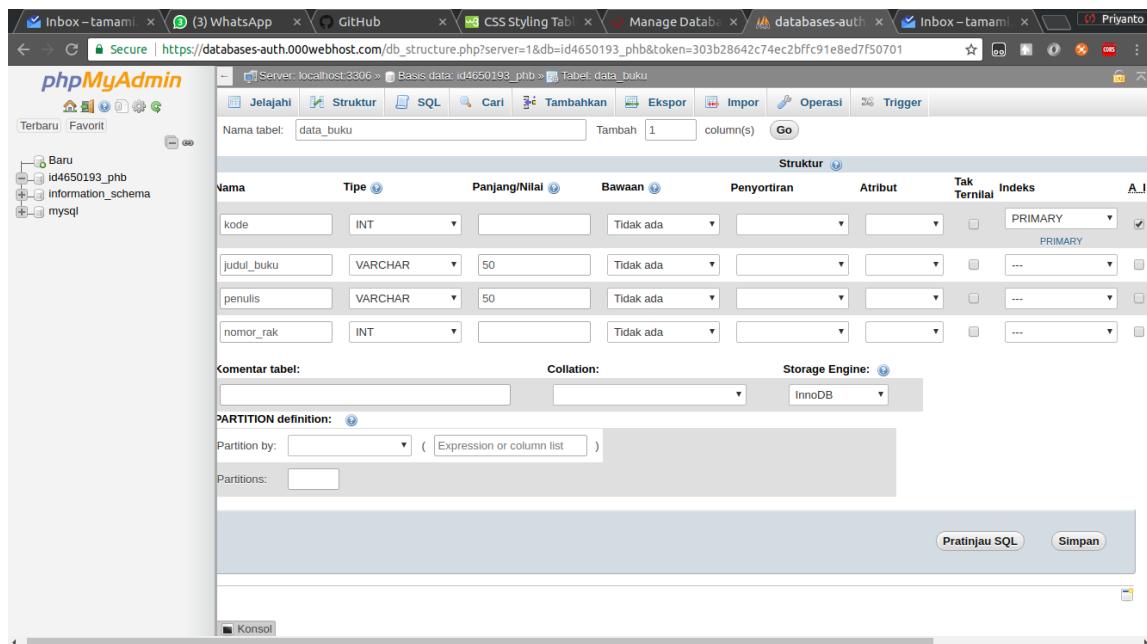
- Kemudian kita pilih nama basis data yang telah kita buat, sehingga muncul halaman untuk membuat tabel, isikan nama tabelnya sesuai aplikasi yang telah kita bangun termasuk jumlah kolom yang dibutuhkan, kemudian klik **go** seperti pada gambar ?? berikut :



Gambar 13.5: Halaman Pembuatan Tabel

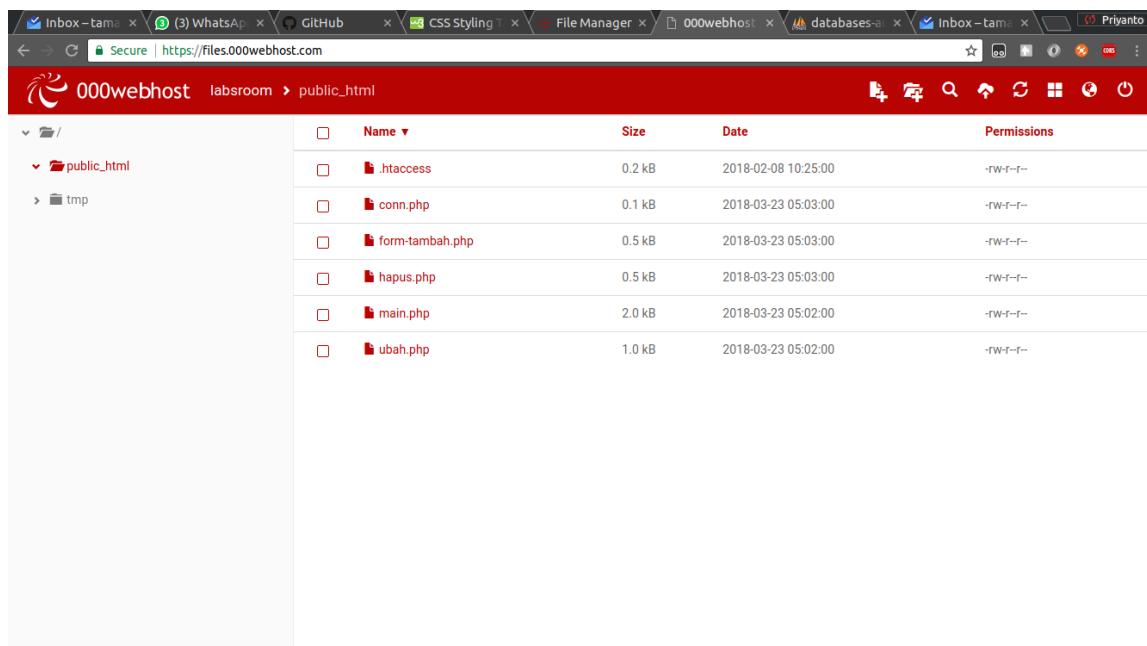
- Setelah itu membuat *field* pada tabel sesuai dengan aplikasi yang telah kita

buat, kemudian menekan tombol **Simpan** seperti pada gambar ?? berikut ini :



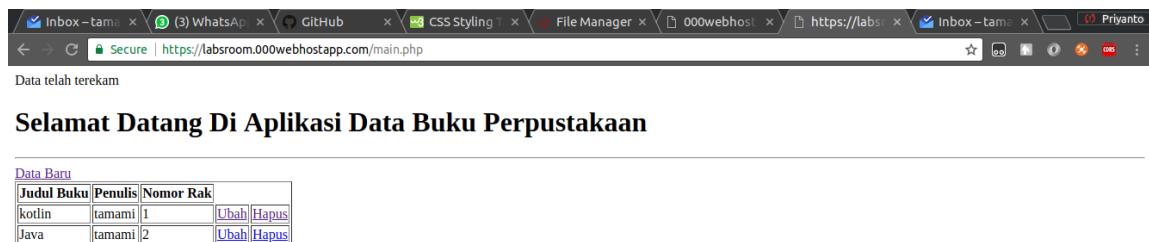
Gambar 13.6: Halaman Pembuatan *Field*

- Setelah tabel selesai dibuat, sekarang tinggal unggah seluruh *file* pada *project* ke 000webhost.com sama seperti pada saat kita mengunggah sebuah *file* PHP, sehingga struktur *file* nya akan terlihat seperti gambar ?? berikut di 000webhost.com :



Gambar 13.7: Struktur *File* Hasil Unggah *Project*

9. Setelah itu aplikasi bisa langsung dapat diakses melalui alamat yang telah disediakan 000webhost.com seperti pada gambar ?? berikut :



Gambar 13.8: Hasil Keluaran Akhir *Project*

13.4 Kesimpulan

Untuk melakukan *publish project* PHP yang telah kita bangun caranya cukup mudah, kita hanya memerlukan penyedia layanan *hosting* yang mendukung PHP dan basis data yang kita buat, setelah itu seluruh *file* pada *project* cukup diunggah ke direktori yang telah ditentukan, maka aplikasi *web* yang telah kita bangun dapat diakses oleh publik.

13.5 Tugas

Lakukan unggah seluruh *file* pada *project* ke *server* penyedia layanan *hosting* yang mendukung PHP dan MySQL, kemudian coba akses alamat yang telah kita buat tersebut.