

PENGEMBANGAN

APLIKASI WEB

Oleh :

Noor Ifada



**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS TRUNOJOYO MADURA
TAHUN 2019**

PENGEMBANGAN

APLIKASI WEB

© MNC Publishing, 2019

Penulis

Noor Ifada

Desain Cover & Penata Isi
Tim Media Nusa Creative

Cetakan I, Oktober 2019

Diterbitkan oleh :



Media Nusa Creative
Anggota IKAPI (162/JTI/2015)
Bukit Cemara Tidar H5 No. 34, Malang
Telp. : 0812.3334.0088
E-mail : mncpublishing.layout@gmail.com
Website : www.mncpublishing.com

ISBN 978-602-462-331-9

Hak Cipta dilindungi undang-undang. Dilarang memperbanyak atau memindahkan sebagian atau seluruh isi buku ke dalam bentuk apapun, secara elektronis maupun mekanis, termasuk fotokopi, merekam, atau dengan teknik perekaman lainnya, tanpa izin tertulis dari Penerbit. Undang-Undang Nomor 19 Tahun 2000 tentang Hak Cipta, Bab XII Ketentuan Pidana, Pasal 72, Ayat (1), (2), dan (6)

KATA PENGANTAR

Rasa syukur Alhamdulillah penulis panjatkan kepada Allah SWT, karena hanya atas ijin-Nya maka penulis dapat menyelesaikan Buku Ajar “Pengembangan Aplikasi Web” ini. Buku ini ditulis sebagai bahan ajar untuk mata kuliah Pengembangan Aplikasi Web yang merupakan salah satu mata kuliah wajib di Prodi Teknik Informatika. Bahasa pemrograman yang menjadi fokus pembahasan di sini adalah PHP. Namun, sebagai tambahan pengetahuan, mahasiswa juga diperkenalkan dengan Node.js yang merupakan Bahasa JavaScript yang dapat digunakan untuk pengembangan aplikasi *web* dinamis. Kompetensi yang diharapkan di akhir perkuliahan adalah bahwa mahasiswa memiliki keahlian untuk mengembangkan *web* dinamis dengan menggunakan kombinasi HTML, CSS, PHP, dan sistem basis data.

Tak lupa penulis juga ingin mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang baik secara langsung maupun tidak langsung telah membantu selama proses penulisan buku ini.

Penulis,

Noor Ifada

MNC Publishing

DAFTAR ISI

KATA PENGANTAR	iii
DAFTAR ISI	v
DAFTAR GAMBAR	ix
DAFTAR TABEL	xiv

BAB 1. KONSEP DASAR PENGEMBANGAN APLIKASI WEB	1
1.1. Web Server dan Web Client	1
1.2. Situs Web	2
1.3. Persiapan Pengembangan Web Dinamis	3
1.3.1. Bahasa	3
1.3.2. Aplikasi untuk Pengembangan Web menggunakan PHP	3
1.3.3. Aplikasi untuk Pengembangan Aplikasi Web menggunakan Node.js	10
1.4. Latihan Soal	16
BAB 2. PHP	17
2.1. Dasar PHP	17
2.1.1. Struktur dan Sintaks PHP	18
2.1.2. Penggunaan Komentar	19
2.1.3. Penulisan kombinasi skrip HTML dan PHP	20
2.1.4. Penggunaan Echo	21
2.1.5. Penggunaan Konkatenasi (<i>Concatenation</i>)	21
2.1.6. Penggunaan Tanda Petik (<i>Quote</i>)	22
2.2. Langkah Operasional	23
2.3. Variabel dan Tipe Data	24
2.4. Array dalam PHP	25
2.4.1. Inisialisasi Array	25

2.4.2. Indeks dari Array	26
2.4.3. Indeks Tidak Terurut dalam array	27
2.5. Operator	27
2.5.1. Operator Aritmatika (<i>Arithmetic</i>)	28
2.5.2. Operator Penugasan (<i>Assignment</i>)	28
2.5.3. Operator Perbandingan	29
2.5.4. Operator Logika (<i>Logical</i>)	30
2.5.5. Operator String	31
2.6. Struktur Kondisi	31
2.6.1. Kondisi IF	31
2.6.2. Kondisi SWITCH	32
2.7. Struktur Perulangan	33
2.7.1. Perulangan FOR	33
2.7.2. Perulangan WHILE	33
2.7.3. Perulangan FOREACH	35
2.8. Fungsi (<i>Function</i>)	36
2.8.1. Deklarasi fungsi (<i>User-defined</i>)	36
2.8.2. Pemanggilan fungsi	37
2.9. Fungsi Built-in	37
2.9.1. Fungsi String	37
2.9.2. Fungsi Array	38
2.9.3. Fungsi Date/Time	38
2.9.4. Fungsi Math	38
2.10. Variabel Global Versus Lokal	38
2.11. Pass-by-Value Versus Pass-by-Reference	39
2.12. Penggunaan Include	40
2.13. Pemrograman Berbasis Obyek	42
2.14. Latihan Soal	43
BAB 3. Implementasi PHP	45
3.1. Pemrosesan File	45
3.2. Pemrosesan Form	47
3.3. Validasi Server-Side	50
3.4. Latihan Soal	52

BAB 4. Sistem Basisdata dan Aplikasi Web	55
4.1. Sistem Basisdata MySQL/MariaDB	55
4.1.1. Manajemen basisdata menggunakan PHPMyAdmin	55
4.1.2. Membuat (Create) Basisdata	56
4.1.3. Membuat (Create) Tabel	57
4.1.4. Memasukkan (Insert) Data Tabel	59
4.1.5. Akun dan Hak Akses (Priviledge) User	61
4.1.6. Structured Query Language (SQL)	61
4.2. Aplikasi Web & Implementasi Basisdata	62
4.2.1. PHP Data Objects (PDO)	63
4.2.2. Koneksi Basisdata dengan Halaman Web	64
4.2.3. PDO Query	64
4.2.4. Serangan SQL Injection	65
4.2.5. PDO Prepared Statement	66
4.2.6. PDO Set Attribute	68
4.2.7. Menampilkan Data Foreign Keys	69
4.3. Latihan Soal	69
BAB 5. Keamanan Aplikasi Web	71
5.1. Prinsip Keamanan	71
5.1.1. Keamanan Basisdata	72
5.1.2. Pengamanan Password	72
5.2. Bentuk Serangan dan Cara Menghindarinya	73
5.2.1. SQL Injection	74
5.2.2. Script Injection	74
5.2.3. XML Attack	75
5.3. Implementasi keamanan yang lemah	75
5.4. Session untuk Authorization	76
5.4.1. Kategori halaman web	77
5.4.2. Halaman LOGIN	78
5.4.3. Halaman LOGOUT	80
5.5. Latihan Soal	81

BAB 6. Node.js	83
6.1. Dasar Node.js	83
6.2. Module dalam Node.js	84
6.3. Node.js sebagai Web Server	84
6.4. HTTP Header pada Node.js	85
6.5. Node.js sebagai File Server	85
6.6. NPM (Node.js Package Manager)	86
6.7. Node.js dan Sistem Basisdata	87
6.7.1. Membuat Koneksi ke Basisdata.....	87
6.7.2. Query: Membuat (Create) Basisdata	88
6.7.3. Query: Membuat (Create) Tabel	88
6.7.4. Query: Tambah Data (Insert) ke Tabel	89
6.7.5. Query: Pilih Data (Select) dari Tabel	89
6.7.6. Query: Perbaharui Data (Update) Tabel	90
6.7.7. Query: Hapus Data (Delete) Tabel	90
6.8. Latihan Soal	91
INDEKS	93
DAFTAR PUSTAKA	95
BIOGRAFI PENULIS	96

DAFTAR GAMBAR

Gambar 1.1.	Ilustrasi Web Server dan Web Client	2
Gambar 1.2.	Pilihan instalasi aplikasi XAMPP	5
Gambar 1.3.	Window untuk memulai proses instalasi XAMPP	5
Gambar 1.4.	Window untuk memilih komponen XAMPP yang akan diinstal	6
Gambar 1.5.	Window untuk menentukan direktori yang menjadi lokasi instalasi XAMPP	6
Gambar 1.6.	Window informasi Bitnami untuk XAMPP	7
Gambar 1.7.	Window konfirmasi bahwa proses instalasi XAMPP dapat dimulai	7
Gambar 1.8.	Window konfirmasi bahwa proses instalasi XAMPP telah selesai	8
Gambar 1.9.	Window konfirmasi pilihan penggunaan Bahasa ..	8
Gambar 1.10.	Window XAMPP Control Panel	9
Gambar 1.11.	Window aktivasi modul Apache dan MySQL pada XAMPP Control Panel	9
Gambar 1.12.	Window hasil pengujian instalasi untuk PHP	10
Gambar 1.13.	Window untuk memulai proses instalasi Node.js ..	12
Gambar 1.14.	Window End-User License Agreement aplikasi Node.js	12
Gambar 1.15.	Window untuk menentukan direktori yang menjadi lokasi instalasi Node.js	13
Gambar 1.16.	Window untuk memilih komponen Node.js yang akan diinstal	13
Gambar 1.17.	Window konfirmasi bahwa proses instalasi Node.js dapat dimulai	14
Gambar 1.18.	Window konfirmasi bahwa proses instalasi Node.js telah selesai	14
Gambar 1.19.	Window aplikasi node.exe	15

Gambar 1.20. Contoh beberapa operasi JavaScript sederhana	15
Gambar 1.21. Window hasil pengujian instalasi untuk Node.js menggunakan konsole Node.js lewat Command Line Interface	16
Gambar 2.1. Sintaks PHP	18
Gambar 2.2. Contoh dokumen PHP sederhana	19
Gambar 2.3. Contoh penggunaan komentar dalam PHP	20
Gambar 2.4. Contoh gaya penulisan kombinasi skrip HTML dan PHP	20
Gambar 2.5. Contoh penggunaan echo	21
Gambar 2.6. Contoh penggunaan konkatenasi	22
Gambar 2.7. Contoh tanda petik tunggal tidak memproses karakter string	22
Gambar 2.8. Langkah operasional dalam PHP.....	23
Gambar 2.9. Contoh inisialisasi Array	26
Gambar 2.10. Contoh indeks dari Array	26
Gambar 2.11. Contoh indeks tak terurut dalam Array	27
Gambar 2.12. Contoh kondisi IF	31
Gambar 2.13. Contoh kondisi SWITCH	32
Gambar 2.14. Contoh perulangan FOR	33
Gambar 2.15. Contoh perulangan WHILE	33
Gambar 2.16. Contoh perulangan DO-WHILE	33
Gambar 2.17. Contoh perulangan FOREACH	35
Gambar 2.18. Sintaks deklarasi fungsi dalam PHP	36
Gambar 2.19. Contoh fungsi dalam PHP	36
Gambar 2.20. Contoh perbandingan antara variabel global dan lokal	38
Gambar 2.21. Contoh perbandingan antara “pass-by-value” dan “pass-by-reference”	39
Gambar 2.22. Contoh dua buah file PHP (“page1.php” dan “page2.php”) yang memiliki blok skrip yang sama	40
Gambar 2.23. Contoh pembuatan file “menu.inc” berdasarkan blok skrip yang sama dalam file “page1.php” dan “page2.php”	40
Gambar 2.24. Contoh penggunaan file “menu.inc” pada file “page1.php” dan “page2.php”	41

Gambar 2.25.	Tabel perkalian 12x12	42
Gambar 2.26.	Kalendar bulan ini	42
Gambar 2.27.	“Daftar Buah”: (a) Skrip HTML dan (b) Hasil tampilan	43
Gambar 3.1.	Contoh pemrosesan file yang digunakan untuk pembuatan counter	47
Gambar 3.2.	Alur pemrosesan form dalam PHP	48
Gambar 3.3.	Contoh potongan skrip file “processData_form.html” yang digunakan untuk membuat form	49
Gambar 3.4.	Contoh potongan skrip file “processData.php” yang digunakan untuk menampilkan hasil isian form ...	50
Gambar 3.5.	Alur validasi server-side dalam PHP	51
Gambar 3.6.	Contoh potongan skrip file “basicValidation.php” yang digunakan untuk memvalidasi isian form	51
Gambar 3.7.	Contoh potongan skrip fungsi “validatePattern” dalam file “validate.inc” yang dipanggil dalam file “basicValidation.php”	52
Gambar 4.1.	Logo MySQL	55
Gambar 4.2.	Halaman utama PHPMyAdmin.	56
Gambar 4.3.	Membuat basisdata “customerDB” menggunakan PHPMyAdmin	57
Gambar 4.4.	Membuat tabel “customer” menggunakan PHPMyAdmin	58
Gambar 4.5.	Mendefinisikan struktur kolom pada tabel “customer” menggunakan PHPMyAdmin	59
Gambar 4.6.	Melihat struktur tabel “customer” menggunakan PHPMyAdmin	59
Gambar 4.7.	Memasukkan data baru pada tabel “customer” menggunakan PHPMyAdmin	60
Gambar 4.8.	Melihat isi data pada tabel “customer” menggunakan PHPMyAdmin	60
Gambar 4.9.	Melihat daftar akun yang memiliki akses ke dalam sistem menggunakan PHPMyAdmin	61
Gambar 4.10.	Membuat SQL untuk melakukan perintah SELECT, UPDATE, DELETE, dan INSERT pada tabel “Customer” menggunakan PHPMyAdmin	62

Gambar 4.11. Hasil isi tabel “customer” setelah SQL dalam Gambar 4.10 dieksekusi	62
Gambar 4.12. Diagram yang menunjukkan hubungan antara aplikasi web dan basisdata	63
Gambar 4.13. Pengecekan fitur PDO dalam PHP dengan menggunakan fungsi “phpinfo()”	63
Gambar 4.14. Contoh skrip PHP untuk melakukan koneksi ke basisdata “customerdb” dengan menggunakan PDO	64
Gambar 4.15. Contoh PDO query	65
Gambar 4.16. Contoh penerapan PDO query yang rentan terhadap serangan SQL injection	66
Gambar 4.17. Ekivalensi query SQL yang menyebabkan contoh dalam Gambar 4.16 rentan terhadap serangan SQL injection	66
Gambar 4.18. Contoh penerapan PDO prepared statement untuk menghindari serangan SQL injection	66
Gambar 4.19. Contoh penerapan PDO prepared statement untuk menambah (INSERT) data ke dalam tabel “customer”	67
Gambar 4.20. Contoh penerapan PDO prepared statement untuk memperbarui (UPDATE) data ke dalam tabel “customer”	67
Gambar 4.21. Contoh penerapan PDO prepared statement untuk menghapus (DELETE) data ke dalam tabel “customer”	68
Gambar 4.22. Contoh penggunaan PDO set attribute	68
Gambar 4.23. Contoh tabel “Animals” dan “Breeds” dimana operasi JOIN diperlukan untuk menampilkan data berdasarkan Foreign Key	69
Gambar 4.24. Contoh SQL untuk operasi JOIN antara tabel “Animals” dan “Breeds”	69
Gambar 5.1. Ilustrasi pengamanan password untuk keamanan aplikasi web	73
Gambar 5.2. Tabel “admin” untuk menyimpan data user dan password	73

Gambar 5.3. Menambahkan data baru pada tabel “admin”	73
Gambar 5.4. Contoh manipulasi SQL yang dapat menyebabkan terjadinya SQL injection	74
Gambar 5.5. Contoh serangan script injection	75
Gambar 5.6. Contoh XML attack	75
Gambar 5.7. Contoh kecerobohan penggunaan variabel tanpa inisialisasi	76
Gambar 5.8. Contoh kecerobohan dalam penggunaan variabel global	76
Gambar 5.9. Ilustrasi penggunaan session	77
Gambar 5.10. Contoh pembuatan halaman non-publik	78
Gambar 5.11. Contoh pembuatan halaman LOGIN	79
Gambar 5.12. Contoh form LOGIN	79
Gambar 5.13. Contoh pembuatan halaman LOGOUT	80
Gambar 6.1. Contoh menjadikan Node.js sebagai web server ...	84
Gambar 6.2. Contoh menambahkan HTTP Header pada Node.js	85
Gambar 6.3. Contoh menjadikan Node.js sebagai file server	86
Gambar 6.4. Skrip command prompt untuk instalasi module mysq	87
Gambar 6.5. Contoh membuat koneksi ke basisdata dengan menggunakan Node.js	87
Gambar 6.6. Contoh membuat basisdata baru dengan menggunakan Node.js	88
Gambar 6.7. Contoh membuat tabel baru dengan menggunakan Node.js	88
Gambar 6.8. Contoh menambahkan data baru (INSERT) ke tabel dengan menggunakan Node.js	89
Gambar 6.9. Contoh memilih data (SELECT) pada tabel dengan menggunakan Node.js	89
Gambar 6.10. Contoh meperbaharui data (UPDATE) pada tabel dengan menggunakan Node.js	90
Gambar 6.11. Contoh menghapus data (DELETE) pada tabel dengan menggunakan Node.js	90

DAFTAR TABEL

Tabel 1.1.	Kebutuhan aplikasi untuk Pengembangan Web menggunakan PHP	4
Tabel 1.2.	Ekivalensi lokasi file di direktori dan URL	10
Tabel 1.3.	Kebutuhan aplikasi untuk Pengembangan Web menggunakan Node.js	11
Tabel 2.1.	Perbandingan penggunaan tanda petik dalam PHP22	
Tabel 2.2.	Contoh deklarasi variabel dalam PHP	24
Tabel 2.3.	Daftar Operator Aritmatika dalam PHP	27
Tabel 2.4.	Daftar Operator Penugasan dalam PHP	28
Tabel 2.5.	Daftar Operator Perbandingan dalam PHP	29
Tabel 2.6.	Daftar Operator Logika dalam PHP	30
Tabel 2.7.	Daftar Operator String dalam PHP	30
Tabel 3.1.	Daftar fungsi untuk pemrosesan file	45
Tabel 3.2.	Contoh Fungsi Built-in untuk validasi menggunakan PHP	52

BAB 1.

KONSEP DASAR PENGEMBANGAN APLIKASI *WEB*

Bab ini menjelaskan konsep dasar pengembangan *web* yang meliputi:

- Perbedaan *web server* dan *web client*
- Bentuk situs *web*
- Persiapan pengembangan aplikasi *web*

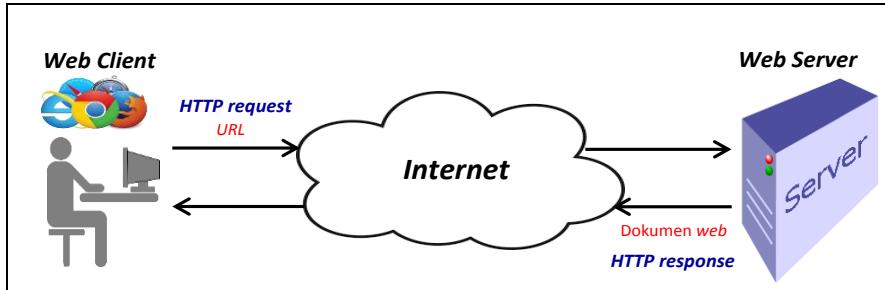
Setelah selesai mempelajari bab ini maka diharapkan kita dapat menjelaskan dan mengimplementasikan konsep dasar tersebut di atas.

1.1. *Web Server* dan *Web Client*

Web server dan *web client* adalah dua sisi selalu saling berkaitan. Gambar 1.1 memperlihatkan bahwa *web client* adalah sisi dimana *user* memasukkan URL¹ melalui *web browser* untuk mengirimkan HTTP² *request* ke *web server* melalui internet. *Web server* akan memproses request yang masuk dan kemudian mengembalikannya dalam bentuk HTTP *response* yang nantinya akan dapat dilihat hasilnya oleh *user* melalui *web browser*. Contoh dari *web client* adalah *web browser* seperti Google Chrome, Mozilla Firefox, Internet Explorer, dan lain-lain. Sedangkan contoh dari *web server* adalah Apache, Microsoft IIS (*Internet Information Server*), dan lain-lain.

¹ URL (*Uniform Resource Locator*) digunakan untuk menentukan lokasi setiap dokumen atau informasi yang ada di dalam situs *web* pada *server*.

² HTTP (*Hypertext Transfer Protocol*) adalah jenis protokol transfer yang menjadi protokol standar untuk akses dokumen *web*



Gambar 1.1. Ilustrasi *Web Server* dan *Web Client*

1.2. Situs Web

Situs *web* pada dasarnya dapat dikategorikan menjadi menjadi dua jenis berdasarkan bentuk informasi yang ditampilkan:

- *Web* statis. Informasi yang ditampilkan di dalam *web* ini bersifat statis atau tetap. Teknologi yang digunakan untuk membangun *web* statis berbasis *client-side* dan fokus pada pengembangan *web* di sisi *front-end*. Contoh teknologi: HTML, CSS, JavaScript, dan lain-lain
- *Web* dinamis. Informasi yang ditampilkan di dalam *web* ini bersifat dinamis. *Web* semacam ini dapat memproses data yang dikirimkan oleh *user* (melalui *form*) dan seringkali juga memerlukan akses basisdata. Teknologi yang digunakan untuk membangun *web* dinamis merupakan gabungan dari teknologi berbasis *client-side* dan *server-side*. Teknologi berbasis *server-side* fokus pada pengembangan *web* di sisi *back-end*. Contoh teknologi: ASP, JSP, ASP.NET, PHP, Node.js, dan lain-lain.



Ketentuan: Buku ini fokus pada pengembangan *web* dinamis. Seluruh *file* yang dicontohkan di dalam buku ini hanya dapat dibuka melalui *web browser* jika *web server*-nya diaktifkan.

1.3. Persiapan Pengembangan *Web* Dinamis

Persiapan yang dibahas di sini meliputi bahasa dan aplikasi apa saja yang akan digunakan di dalam buku ini untuk pembuatan *web* dinamis.

1.3.1. Bahasa

Bahasa yang akan digunakan di dalam buku ini meliputi:

- PHP dan Node.js: merupakan bahasa pemrograman *server-side* untuk pengembangan *web* dinamis. PHP akan mulai dibahas sejak BAB 2, sedangkan Node.js hanya akan dibahas di BAB 6.
- HTML dan CSS: merupakan bahasa *client-side* yang digunakan untuk mendeskripsikan konten atau struktur suatu halaman *web* dan memformat konten atau membuat *layout* halaman *web*
- SQL: yaitu bahasa standar untuk sistem manajemen basisdata relasional



Ketentuan: Buku ini fokus kepada penggunaan PHP untuk pengembangan *web* dinamis. Oleh karena itu, buku ini tidak akan membahas Node.js sedetil pembahasan PHP.

1.3.2. Aplikasi untuk Pengembangan *Web* menggunakan PHP

Aplikasi yang diperlukan untuk pengembangan *web* menggunakan PHP adalah sebagai berikut:

- Penyunting teks, yaitu aplikasi untuk menuliskan skrip dengan menggunakan komputer. Contoh penyunting teks: Notepad++ (<https://notepad-plus-plus.org/>), Sublime Text (<http://www.sublimetext.com/>), Brackets (<http://brackets.io/>)
- *Web browser*, yaitu aplikasi untuk membuka dokumen web. Contoh dokumen web: Google Chrome, Mozilla Firefox, Internet Explorer, Opera
- *Web server*, yaitu aplikasi yang menangani layanan *web*. Contoh *web server*: Apache, Microsoft IIS (Internet Information Server);
- PHP Engine
- Sistem Basisdata server

Tabel 1.1 memperlihatkan daftar kebutuhan aplikasi untuk pengembangan *web* dinamis menggunakan PHP.

Tabel 1.1. Kebutuhan aplikasi untuk Pengembangan *Web* menggunakan PHP

Jenis Aplikasi	Aplikasi yang digunakan
Penyunting teks	Notepad++ ³
Web Browser	Google Chrome ⁴
Web Server	Apache HTTP Server
PHP Engine	PHP 7
Sistem Manajemen Basisdata Relasional Server	MySQL atau MariaDB
Administrator Basisdata	phpMyAdmin

XAMPP adalah paket aplikasi server *localhost* yang meliputi:

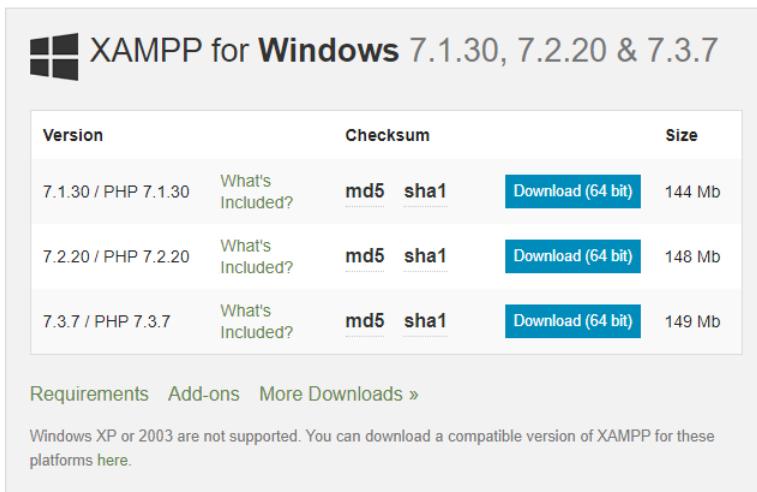
- Apache
- MySQL
- PHP
- phpMyAdmin
- FileZilla FTP Server
- Tomcat
- XAMPP Control Panel

Langkah-langkah instalasi XAMPP di Sistem Operasi Windows:

1. Unduh aplikasi terbaru XAMPP melalui: <https://www.apachefriends.org/download.html>. Gambar 1.2 memperlihatkan pilihan instalasi XAMPP.

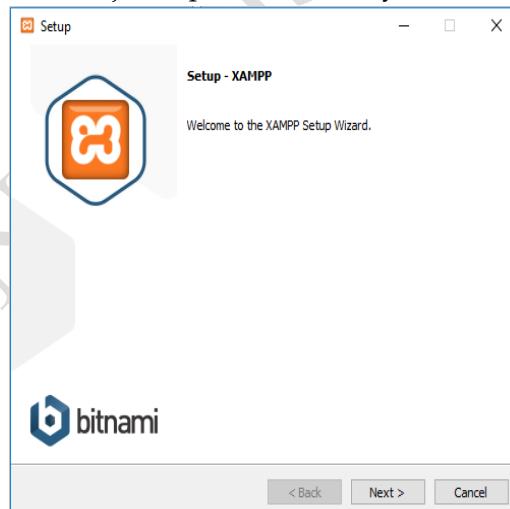
³ Alternatif penyunting teks: Sublime Text (<http://www.sublimetext.com/>), Brackets (<http://brackets.io/>)

⁴ Alternatif *web browser*: Mozilla Firefox, Internet Explorer, Opera



Gambar 1.2. Pilihan instalasi aplikasi XAMPP

- Klik dua kali file instalasi XAMPP yang telah diunduh, maka akan muncul window untuk memulai proses instalasi (Gambar 1.3). Pilih **Next** untuk lanjut ke proses berikutnya.

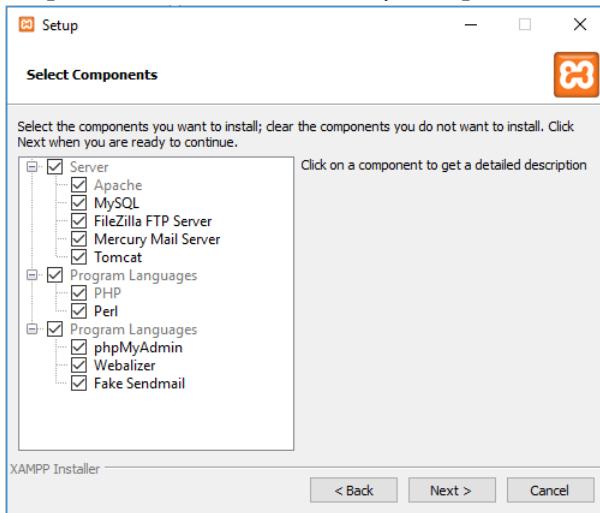


Gambar 1.3. Window untuk memulai proses instalasi XAMPP

- Selanjutnya akan muncul window untuk memilih komponen yang akan di-instal (Gambar 1.4). Komponen yang harus diinstal minimal meliputi *Server: Apache, MySQL; Program Languages:*

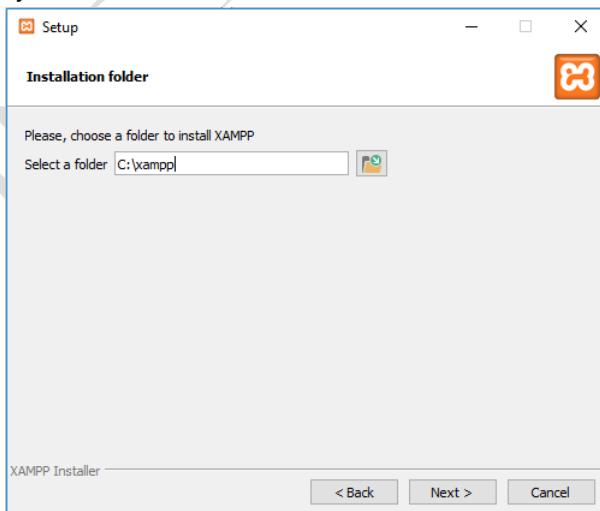
PENGEMBANGAN APLIKASI WEB

PHP, Program Languages: phpMyAdmin. Tapi untuk saat ini pilih semua komponen. Klik **Next** untuk lanjut ke proses selanjutnya.



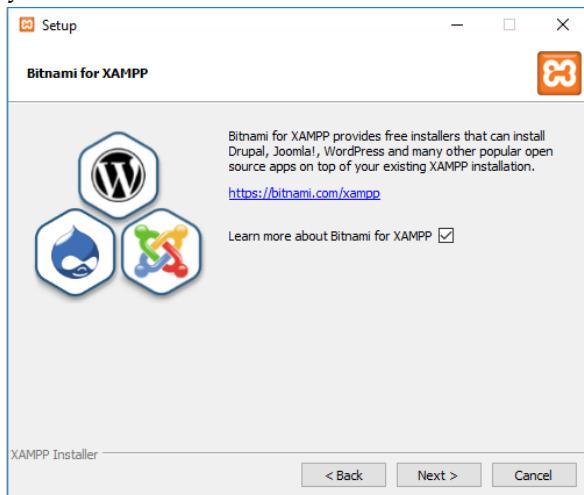
Gambar 1.4. Window untuk memilih komponen XAMPP yang akan diinstal

4. Muncul *window* untuk menentukan direktori yang menjadi lokasi instalasi (Gambar 1.5). Pilih **Next** untuk lanjut ke proses selanjutnya.



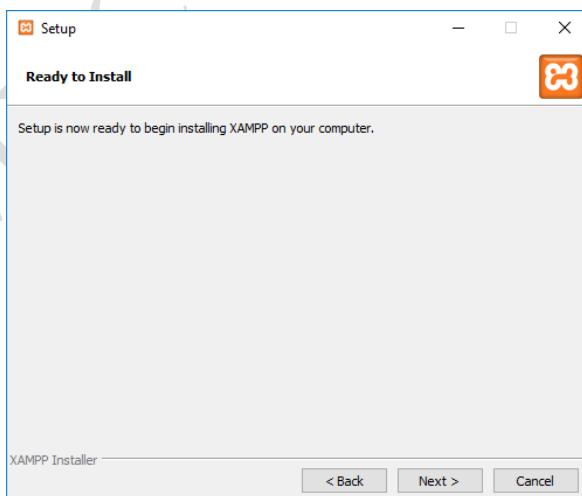
Gambar 1.5. Window untuk menentukan direktori yang menjadi lokasi instalasi XAMPP

5. Muncul *window* yang menginformasikan mengenai Bitnami for XAMPP (Gambar 1.6). Pilih **Next** untuk lanjut ke proses selanjutnya.

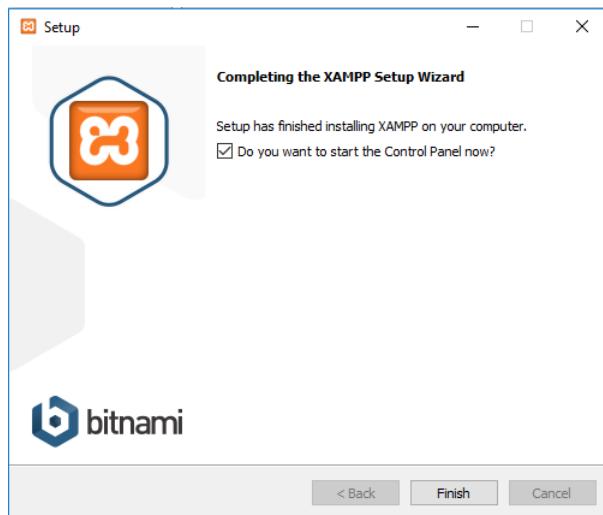


Gambar 1.6. *Window* informasi Bitnami untuk XAMPP

6. Muncul *window* yang mengkonfirmasikan bahwa proses instalasi dapat dimulai (Gambar 1.7). Pilih **Next** untuk lanjut ke proses selanjutnya dan tunggu hingga proses instalasi selesai (Gambar 1.8).

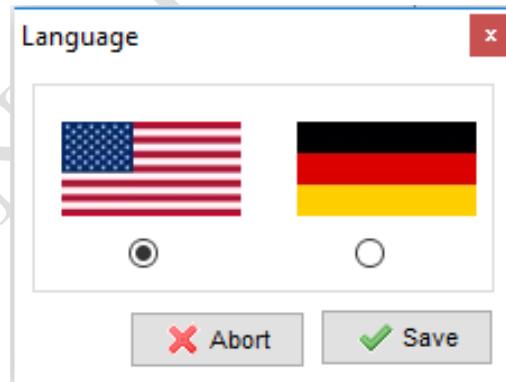


Gambar 1.7. *Window* konfirmasi bahwa proses instalasi XAMPP dapat dimulai



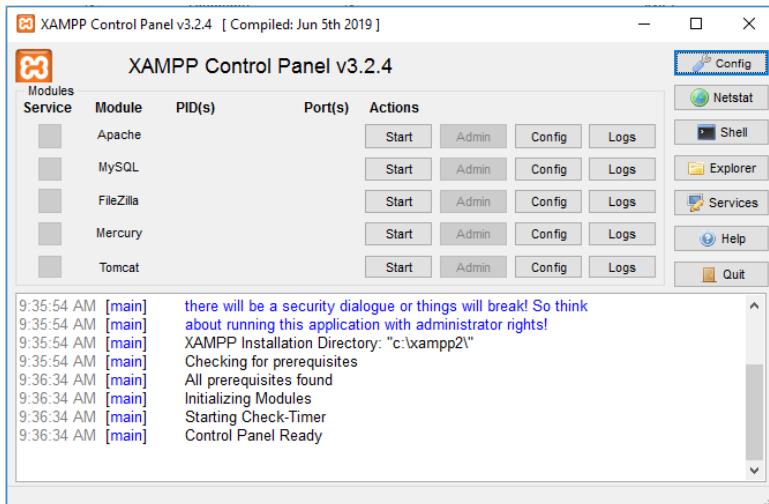
Gambar 1.8. *Window konfirmasi bahwa proses instalasi XAMPP telah selesai*

7. Muncul *window* konfirmasi pilihan penggunaan Bahasa (Gambar 1.9). Pilihan Bahasa Inggris atau Jerman dengan meng-klik *radio button* yang disediakan. Pilih **Save** untuk menyelesaikan proses instalasi.



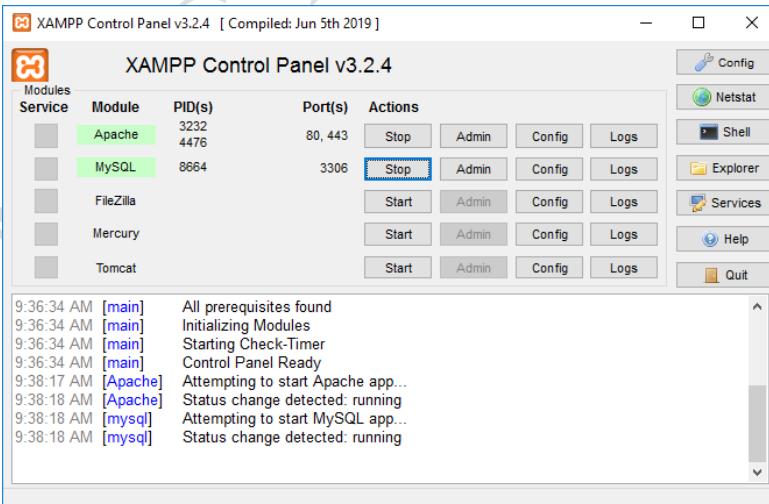
Gambar 1.9. *Window konfirmasi pilihan penggunaan Bahasa*

8. Gambar 1.10 memperlihatkan hasil akhir instalasi berupa XAMPP Control Panel.



Gambar 1.10. Window XAMPP Control Panel

Untuk menggunakan PHP dalam pengembangan *web* dinamis, kita perlu mengaktifkan modul Apache dan MySQL yang diperlukan. Gambar 1.11 memperlihatkan bahwa kedua modul telah aktif.



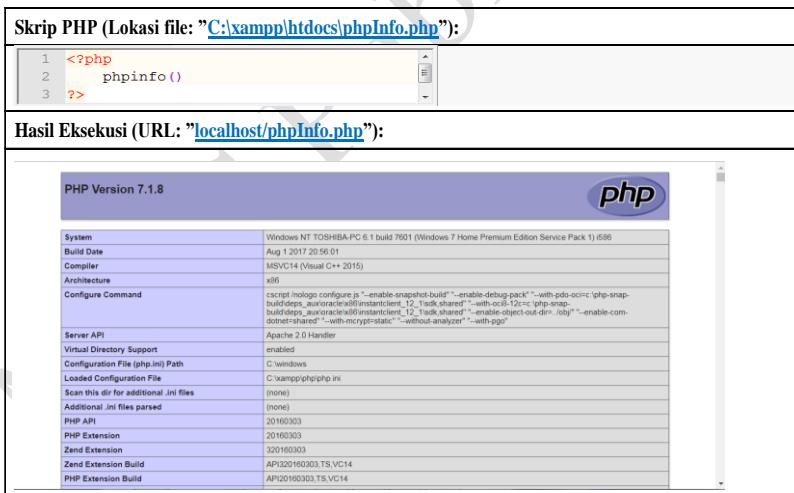
Gambar 1.11. Window aktivasi modul Apache dan MySQL pada XAMPP Control Panel

Untuk pengujian instalasi berarti bahwa kita harus memahami ekivalensi alamat antara lokasi *file* yang berisi skrip PHP di direktori dalam komputer dan URL di *web browser*, seperti yang diperlihatkan dalam Tabel 1.2.

Tabel 1.2. Ekivalensi lokasi file di direktori dan URL

Lokasi	Alamat
Direktori dalam komputer	<u>C:\xampp\htdocs\</u>
URL di <i>web browser</i>	<u>http://localhost/</u>

Buatlah skrip PHP bernama “`phpInfo.php`” dan letakkan di dalam direktori “`C:\xampp\htdocs\`” seperti yang diperlihatkan dalam Gambar 1.12. Selanjutkan buka file tersebut di *web browser* dengan URL “<http://localhost/phpInfo.php>”. Kita dapat memastikan bahwa instalasi telah berhasil dengan baik jika kita dapat melihat hasil eksekusi seperti dalam Gambar 1.12.



Gambar 1.12. Window hasil pengujian instalasi untuk PHP

1.3.3. Aplikasi untuk Pengembangan Aplikasi Web menggunakan Node.js

Aplikasi yang diperlukan untuk pengembangan web menggunakan Node.js adalah sebagai berikut:

- Penyunting teks, yaitu aplikasi untuk menuliskan skrip dengan menggunakan komputer. Contoh penyunting teks: Notepad++ (<https://notepad-plus-plus.org/>), Sublime Text (<http://www.sublimetext.com/>), Brackets (<http://brackets.io/>)
- *Web browser*, yaitu aplikasi untuk membuka dokumen web. Contoh dokumen web: Google Chrome, Mozilla Firefox, Internet Explorer, Opera
- Node.js Engine
- Node.js Package Manager (NPM)
- Sistem Basisdata server
- Command Line Interface

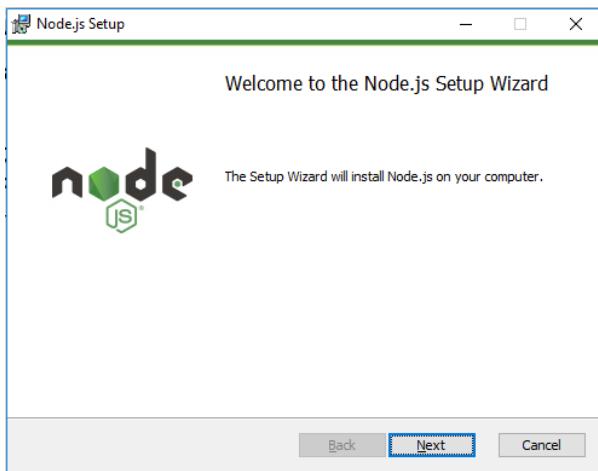
Tabel 1.3. memperlihatkan daftar kebutuhan aplikasi untuk pengembangan *web* dinamis menggunakan Node.js.

Tabel 1.3. Kebutuhan aplikasi untuk Pengembangan Web menggunakan Node.js

Jenis Aplikasi	Aplikasi yang digunakan
Penyunting teks	Notepad++
<i>Web Browser</i>	Google Chrome
Node.js Engine	Node.js
Node.js Package Manager (NPM)	NPM
Sistem Manajemen Basisdata Server	MySQL atau MariaDB
Command Line Interface	Command Prompt

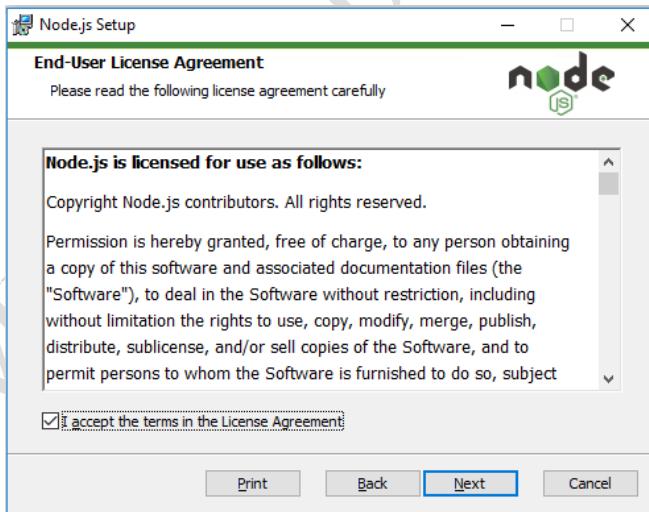
Langkah-langkah instalasi Node.js di Sistem Operasi Windows:

1. Unduh aplikasi Node.js melalui: <https://nodejs.org/en/download/>.
2. Muncul window untuk memulai proses instalasi Node.js (Gambar 1.13). Pilih **Next** untuk lanjut ke proses berikutnya.



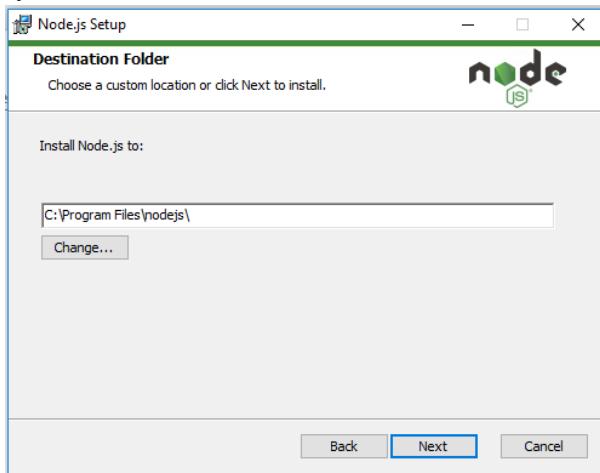
Gambar 1.13. *Window* untuk memulai proses instalasi Node.js

3. Berikutnya akan muncul *window* *End-User License Agreement* Node.js (Gambar 1.14). Aktifkan *checkbox* yang tersedia dan kemudian pilih **Next** untuk lanjut ke proses berikutnya.



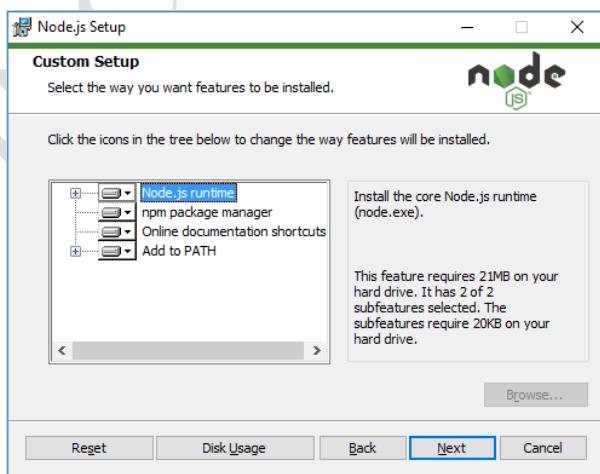
Gambar 1.14. *Window* *End-User License Agreement* aplikasi Node.js

4. Muncul *window* untuk menentukan direktori yang menjadi lokasi instalasi (Gambar 1.15). Pilih **Next** untuk lanjut ke proses selanjutnya.



Gambar 1.15. *Window* untuk menentukan direktori yang menjadi lokasi instalasi Node.js

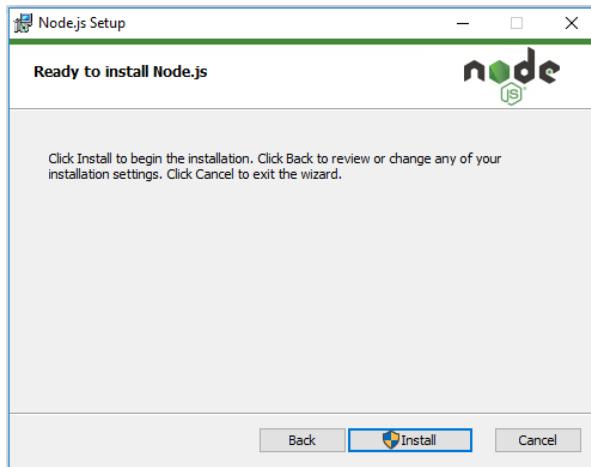
5. Selanjutnya akan muncul *window* untuk memilih komponen yang akan di-instal (Gambar 1.16). Untuk langsung saja klik **Next** untuk lanjut ke proses selanjutnya.



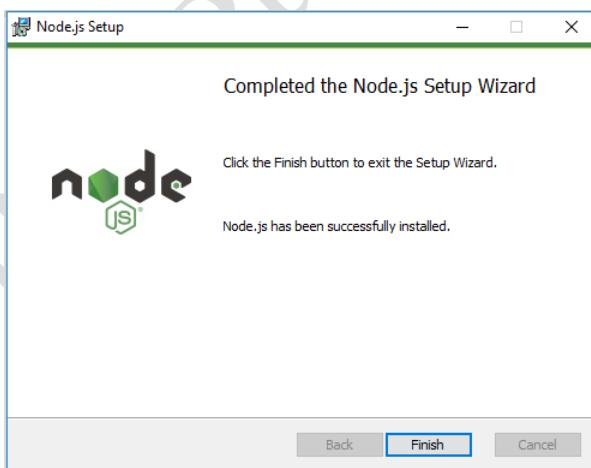
Gambar 1.16. *Window* untuk memilih komponen Node.js yang akan diinstal

PENGEMBANGAN APLIKASI WEB

6. Muncul *window* yang mengkonfirmasikan bahwa proses instalasi dapat dimulai (Gambar 1.17). Pilih **Next** untuk lanjut ke proses selanjutnya dan tunggu hingga proses instalasi selesai (Gambar 1.18).



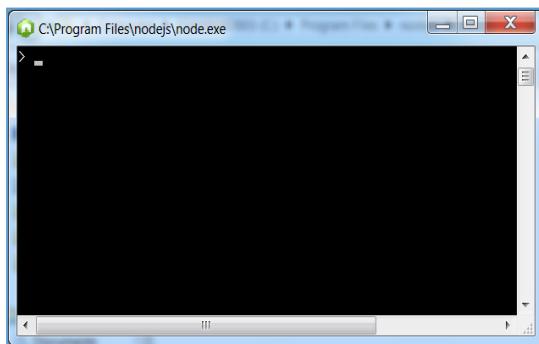
Gambar 1.17. *Window* konfirmasi bahwa proses instalasi Node.js dapat dimulai



Gambar 1.18. *Window* konfirmasi bahwa proses instalasi Node.js telah selesai

Pengujian instalasi Node.js dapat dilakukan dengan dua cara, yaitu:

1. Menggunakan konsole Node.js lewat aplikasi *node*
 - Dengan menggunakan aplikasi *Windows Explorer*, aktifkan aplikasi  **node.exe** (Gambar 1.19) yang ada pada direktori “C:\Program Files\Node.js\”:



Gambar 1.19. Window aplikasi node.exe

- Lakukan beberapa operasi JavaScript sederhana seperti yang diperlihatkan dalam Gambar 1.20:

 A screenshot of the node.js command-line interface. The window title is "C:\Program Files\nodejs\node.e...". The console output shows the following:


```
> 1+3
4
> console.log("Hello, World!")
Hello, World!
undefined
> -
```

Gambar 1.20. Contoh beberapa operasi JavaScript sederhana

2. Menggunakan konsole Node.js lewat *Command Line Interface*
 - Dengan menggunakan aplikasi penyunting teks, buatlah sebuah file Node.js dan berilah nama “main.js” seperti yang diperlihatkan dalam Gambar 1.21:

Skrip Node.js (Lokasi file: "[C:\@ifa\PAW](#)":)

```
1  /* Hello, World! program in node.js */
2  console.log("Hello, World!");
```

Skrip command prompt:

```
C:\@ifa\PAW>node main.js
Hello, World!
```

Gambar 1.21. *Window hasil pengujian instalasi untuk Node.js menggunakan konsole Node.js lewat Command Line Interface*

- Eksekusi file “main.js” dengan menggunakan Node.js interpreter:
 - Buka program *Command Line Interface*: untuk Sistem Operasi Windows, klik tombol *Start* dan carilah program *Command Prompt* (atau ketikkan “cmd” pada kotak pencarian)
 - Arahkan direktori sesuai dengan lokasi penyimpanan file “main.js” dan kemudian eksekusi dengan menggunakan perintah *node*
 - Apabila instalasi Node.js berjalan dengan sukses, maka eksekusi file “main.js” akan menampilkan hasil pada Gambar 1.21.

1.4. Latihan Soal



Latihan 1.1

Lakukan instalasi dan pengujian XAMPP seperti yang dijelaskan di dalam sub-bab 1.3.2!



Latihan 1.2

Lakukan instalasi dan pengujian Node.js seperti yang dijelaskan di dalam sub-bab 1.3.3!

BAB 2. PHP

Bab ini membahas PHP sebagai salah satu bahasa pemrograman yang digunakan untuk pengembangan *web* dinamis. Selain mempelajari dasar PHP, di sini kita juga akan mempelajari perintah-perintah PHP berikut:

- langkah operasional
- pembuatan variabel dan tipe data
- penggunaan operator
- penggunaan struktur kondisi dan struktur perulangan
- pembuatan fungsi
- variabel global versus lokal
- *pass-by-value* versus *pass-by-reference*
- penggunaan *include*
- pemrograman berbasis obyek

Setelah selesai mempelajari bab ini maka diharapkan kita dapat mendemonstrasikan aplikasi perintah-perintah PHP tersebut di atas untuk pengembangan *web* dinamis.

2.1. Dasar PHP

PHP (Hypertext Preprocessor) merupakan bahasa pemrograman *server-side* yang pertama kali dikembangkan oleh Rasmus Lerdorf pada tahun 1994 dan sekarang dikelola oleh PHP Group (<http://php.net/>). PHP dapat secara bebas digunakan pada berbagai sistem operasi seperti Windows, Linux, Mac dan Unix. PHP juga dapat diaplikasikan dengan menggunakan berbagai macam *web server* (contoh: IIS, Apache) dan sistem manajemen basisdata relasional (contoh: MySQL, MS SQL Server, Oracle).



Catatan: Sintaks PHP mirip dengan sintaks C, Java dan Perl. Dokumentasi PHP dapat dilihat pada <http://www.php.net/docs.php>.

2.1.1. Struktur dan Sintaks PHP

Skrip PHP harus diletakkan di dalam *tag* PHP dan setiap perintah harus diakhiri dengan simbol titik koma (";"). Gambar 2.1 memperlihatkan sintaks PHP.

```
<?php  
    // perintah PHP selalu diakhiri dengan tanda titik koma ";"  
?>
```

Gambar 2.1. Sintaks PHP



Catatan: Ekstensi file untuk PHP adalah ".php"



Catatan: Skrip PHP juga dapat diletakkan di antara *tag* pembuka "<?" dan *tag* penutup "?>", atau *tag* pembuka "<%>" dan *tag* penutup "%>". Namun penggunaan kedua alternatif *tag* tersebut tidak direkomendasikan.

Sebuah file PHP umumnya berisi kombinasi skrip HTML dan PHP. Bahkan skrip HMTL saja juga merupakan skrip PHP yang valid. Gambar 2.22 memperlihatkan contoh dokumen PHP sederhana yang mengkombinasikan skrip HTML dan PHP. Penggunaan elemen deklarasi <!DOCTYPE html> menandakan bahwa dokumen PHP merupakan dokumen HTML5. Seluruh skrip HTML dan PHP diletakkan di dalam elemen kontainer <html>. Untuk menambahkan informasi judul dokumen PHP, elemen <title> diletakkan di dalam elemen <head>. Seluruh informasi yang akan ditampilkan di dalam web browser di diletakkan di dalam elemen <body>. Skrip PHP <?php echo "Kami sedang mulai belajar PHP"; ?> digunakan untuk menampilkan tulisan dalam *web browser*.



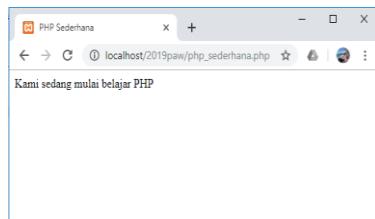
Ketentuan: Untuk kemudahan ilustrasi, contoh-contoh yang ditunjukkan dalam sub-bab dan bab setelah ini (sebagian besar) tidak akan menampilkan penggunaan elemen <!DOCTYPE html>, <html>, <head> dan <body>. Dengan demikian, mereka harus selalu dipahami sebagai (potongan) skrip yang diletakkan di dalam elemen <body>.

Skrip HTML dan PHP:

```

1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <title>PHP Sederhana</title>
5.   </head>
6.   <body>
7.     <?php
8.       echo "Kami sedang mulai belajar PHP";
9.     ?>
10.   </body>
11. </html>
```

Hasil Eksekusi:



Gambar 2.22. Contoh dokumen PHP sederhana

2.1.2. Penggunaan Komentar

Penambahan komentar di dalam skrip PHP dapat dilakukan dengan menggunakan tanda “//” atau “#” di awal baris komentar. Baris-baris komentar juga dapat dibuat dengan menempatkannya di dalam tanda pembuka “/*” dan tanda penutup “*/”. Gambar 2.3 memperlihatkan contoh penggunaan komentar dalam PHP.

 **Catatan:** Dokumentasi penulisan komentar dalam PHP dapat dilihat pada <http://php.net/manual/en/language.basic-syntax.comments.php>.

Skrip PHP:

```
1. <?php
2.     echo 'Line #1'; // membuat komentar per baris
3.     /* Membuat komentar
4.      sebanyak beberapa baris
5.     */
6.     echo 'Line #2';
7.     echo 'Line #3'; # ini juga membuat komentar per baris
8. ?>
```

Gambar 2.3. Contoh penggunaan komentar dalam PHP

2.1.3. Penulisan kombinasi skrip HTML dan PHP

Penulisan kombinasi skrip HTML dan PHP dapat dilakukan dengan dua macam gaya seperti yang diperlihatkan dalam Gambar 2.4. Contoh gaya pertama (baris 2-15) memerlukan lebih banyak penggunaan elemen PHP, sedangkan contoh gaya kedua (baris 18-27) memerlukan penggunaan fungsi `echo` dan tanda *quote* sebagai penanda karakter *string*.

Skrip HTML dan PHP:

```
1. // Gaya penulisan #1: yang menuliskan skrip HMTL dan PHP dalam sintaks yang terpisah
2. <?php
3.     if ($expression)
4.     {
5.     ?>
6.         <b>This is true.</b>
7.     <?php
8.     }
9.     else
10.    {
11.    ?>
12.        <b>This is false.</b>
13.    <?php
14.    }
15.    ?>
16.
17. // Gaya penulisan #2: yang menuliskan skrip HMTL di dalam sintaks PHP
18. <?php
19.     if ($expression)
20.     {
21.         echo "<b>This is true.</b>";
22.     }
23.     else
24.     {
25.         echo "<b>This is false.</b>";
26.     }
27. ?>
```

Gambar 2.4. Contoh gaya penulisan kombinasi skrip HTML dan PHP

2.1.4. Penggunaan Echo

Echo digunakan untuk mencetak atau menampilkan hasil di web *web browser*. Gambar 2.5 memperlihatkan beberapa contoh penggunaan **echo** untuk menampilkan teks “hello world” (baris 2), angka 42 (baris 4), teks “hello world” (baris 6), teks gabungan huruf dan kata “hello42world” (baris 8) dan hasil akar dari perkalian angka 19 dan 64 (baris 11).



Catatan: Dokumentasi penggunaan echo dalam PHP dapat dilihat pada <http://php.net/manual/en/function.echo.php>.

Skrip PHP:

```

1. <?php
2.     echo 'hello world';
3.     echo '<br>';
4.     echo 42;
5.     echo '<br>';
6.     echo ('hello world');
7.     echo '<br>';
8.     echo 'hello', 42, 'world';
9.     echo '<br>';
10.    $value = 19;
11.    echo sqrt($value * 64);
12. ?>
```

Hasil Eksekusi:

```

hello world
42
hello world
hello42world
34.871191548325
```

Gambar 2.5. Contoh penggunaan echo

2.1.5. Penggunaan Konkatenasi (*Concatenation*)

Konkatenasi adalah operasi untuk menggabungkan data teks atau string dengan menggunakan tanda titik (“.”). Gambar 2.6 memperlihatkan contoh konkatenasi antara dua teks “Hello” dan “World!” menjadi satu teks “Hello World!”. Pembahasan detil ada di sub-bab 0.

Skrip PHP:

```
1. <?php  
2.     echo "Hello" . " World!";  
3. ?>
```

Hasil Eksekusi:

Hello World!

Gambar 2.6. Contoh penggunaan konkatenasi

2.1.6. Penggunaan Tanda Petik (*Quote*)

Terdapat dua pilihan penggunaan tanda petik (*quote*) dalam PHP, yaitu menggunakan tanda petik tunggal (*single quote*) dan tanda petik ganda (*double quote*). Tabel 2.1 memperlihatkan perbandingan penggunaan kedua tanda petik dalam PHP.

Tabel 2.1. Perbandingan penggunaan tanda petik dalam PHP

	Tanda petik tunggal	Tanda petik ganda
Tanda petik tunggal	echo 'Today is Paul\'s birthday';	echo "Today is Paul's birthday";
Tanda petik ganda	echo '<p class="heading">Hello</p>';	echo "<p class=\"heading\">Hello</p>";

Satu hal penting yang harus diperhatikan dalam penggunaan tanda petik adalah bahwa tanda petik tunggal tidak memproses karakter string (lihat contoh dalam Gambar 2.7).

Skrip PHP:

```
1. <?php  
2.     $x = 3;  
3.     $y = 7;  
4.     echo 'Multiply: ($x * $y)';  
5.     echo "  
6.     echo "Multiply: ($x * $y)";  
7. ?>
```

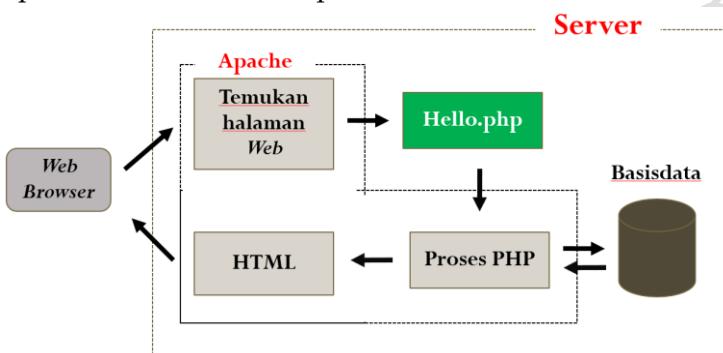
Hasil Eksekusi:

Multiply: (\$x * \$y)
Multiply: (3 * 7)

Gambar 2.7. Contoh tanda petik tunggal tidak memproses karakter *string*

2.2. Langkah Operasional

Gambar 2.8 memperlihatkan langkah operasional dalam PHP. Tahapan operasional dimulai dari *user* dari sisi *client* yang menggunakan *web browser* untuk mengakses atau mencari halaman *web* dengan nama *file* "Hello.php". Dari sisi *client*, proses berlanjut ke sisi *server* yang meliputi *web server* dan basisdata hingga pada akhirnya halaman *web* yang dicari akan dikembalikan atau ditampilkan ke *web browser* kepada *user*.



Gambar 2.8. Langkah operasional dalam PHP

2.3. Variabel dan Tipe Data

Variabel merupakan kontainer yang digunakan untuk menyimpan nilai suatu data. Terdapat beberapa aturan untuk penamaan variabel di dalam PHP, yaitu:

- Variabel selalu diawali dengan simbol “dollar” (\$) dan kemudian diikuti oleh nama variabel tersebut
- Nama variabel harus diawali dengan suatu huruf atau simbol garis bawah (“_”)
- Nama variabel hanya boleh menggunakan karakter alfanumerik dan simbol garis bawah (“_”)
- Nama variabel adalah *case-sensitive* (“\$angka” dan “\$ANGKA” adalah dua variabel yang berbeda)

Terdapat tujuh macam tipe data dalam PHP, yaitu:

- *Boolean*, yang merepresentasikan nilai *true* (benar) atau *false* (salah)

- *String*, yang merepresentasikan suatu teks seperti karakter, kata, atau kalimat
- *Integer*, yang merepresentasikan angka berbentuk bilangan bulat
- *Floating point*, yang merepresentasikan angka berbentuk decimal atau eksponensial
- *Array*, yang berbentuk kontainer untuk menampung satu atau lebih data
- *Null*, yang merepresentasikan suatu nilai yang kosong atau tidak ada
- *Object*, yang menyimpan data dan informasi tentang bagaimana data tersebut diproses

Tabel 2.2 memperlihatkan contoh deklarasi variabel untuk masing-masing tipe data dalam PHP.

Tabel 2.2. Contoh deklarasi variabel dalam PHP

Tipe Data	Contoh
<i>Boolean</i>	\$x = true; \$y = false;
<i>String</i>	\$name = 'eve';
<i>Integer</i>	\$age = 17; \$cost = \$age * 10;
<i>Floating point</i>	\$price = 1000.75; \$discount = \$price * 0.25;
<i>Array</i>	\$number = array(1, 3, 5, 7);
<i>Null</i>	\$initial = null;
<i>Object</i>	new stdClass;



Catatan: Variabel dalam PHP tidak perlu dideklarasikan tipe datanya (*weakly typed*), sama halnya seperti Python. Contoh bahasa pemrograman yang harus mendeklarasikan tipe datanya (*strongly typed languages*) adalah C, C++, Java.



Catatan: Dokumentasi mengenai variabel dalam PHP dapat dilihat pada <http://php.net/manual/en/language.variables.php>.



Catatan: Dokumentasi mengenai tipe data dalam PHP dapat dilihat pada <http://php.net/manual/en/language.types.php>.

2.4. Array dalam PHP

Array dalam PHP perlu dibahas secara khusus karena memiliki karakteristik unik jika dibandingkan dengan bahasa pemrograman lain.

2.4.1. Inisialisasi Array

Array dapat diinisialisasi dengan beberapa cara dalam PHP. Gambar 2.9 memperlihatkan contoh inisialisasi array. Baris 3 memperlihatkan inisialisasi array sedemikian hingga variabel \$foo memiliki tujuh buah data bertipe *integer*. Indeks dalam PHP dimulai dari 0, sehingga hasil eksekusi dari baris 4 dan 5 akan menampilkan angka 36 dan 27. Baris 6 memperlihatkan inisialisasi array untuk memperbarui data variabel \$foo pada indeks ke-3 menjadi bernilai 42.

Kita dapat menggunakan variabel untuk menspesifikasikan indeks dari array. Baris 9 memperlihatkan inisialisasi array untuk memperbarui data variabel \$foo pada indeks ke-\$i (yang bernilai 4 karena telah diinisialisasi pada baris 8) menjadi bernilai 34.

Array juga dapat diinisialisasi sehingga berisi berbagai tipe data. Baris 12 memperlihatkan inisialisasi array sedemikian hingga variabel \$bar memiliki empat buah data yang merupakan kombinasi tipe data *integer*, *floating point*, *string*, dan *Boolean*. Baris 13 memperlihatkan bahwa kita juga dapat menginisialisasi atau memperbarui array agar bernilai kosong.

Skrip PHP:

```
1. <?php
2.     // inisialisasi array:
3.     $foo = array(36, 19, 27, 9, 10, 10, -89);
4.     echo $foo[0];           // 36
5.     echo $foo[2];           // 27
6.     $foo[3] = 42;           // $foo[3] berubah menjadi 42
7.     echo $foo[3];
8.     $i = 4;
9.     $foo[$i] = 34;          // $foo[4] berubah menjadi 34
10.    echo $foo[$i];
11.    // isi array berbeda tipe data:
12.    $bar = array(42, 3.14, 'Fred', True);
13.    $bar = array();          // array kosong
14. ?>
```

Gambar 2.9. Contoh inisialisasi Array

2.4.2. Indeks dari Array

Indeks dari *array* dalam PHP tidak harus bertipe *integer*. Gambar 2.10 memperlihatkan contoh variabel \$person yang bertipe *array* yang memiliki indeks bertipe *string*. Dengan menggunakan konstruktor *array* PHP, komponen *array* dapat dibuat dengan menggunakan kombinasi indeks dan nilai, dimana indeks merupakan penunjuk posisi dimana nilai disimpan. Perhatikan bahwa baris 2 memperlihatkan contoh penggunaan tanda panah ("=>") untuk memberikan nilai kepada indeks.

Skrip PHP:

```
1. <?php
2.     $person = array('nama' => 'Adi', 'usia' => 15, 'pelajar' => True);
3.     echo $person['usia'];
4.     $person['usia'] = 44;
5.     $person['hobi'] = 'Bersepeda';
6.     echo $person['hobi'];           // OK
7.     echo $person['dob'];           // Runtime error
8.     if (array_key_exists('dob', $person))
9.         echo $person['dob'];       // OK
10. ?>
```

Gambar 2.10. Contoh indeks dari Array

2.4.3. Indeks Tidak Terurut dalam array

Indeks dari *array* dalam PHP dapat dibuat menjadi tidak terurut (jika ia bertipe *integer*) seperti yang diperlihatkan dalam Gambar 2.11. Baris 3 memperlihatkan inisialisasi atau perbaharuan *array* sedemikian hingga variabel \$foo memiliki delapan data bertipe

integer dengan indeks dari data yang terbaru bernilai 99 (dan bukannya bernilai 7 seperti pada lazimnya). Dengan demikian, isi variabel \$foo dapat dituliskan dengan menggunakan konstruktor *array* PHP sebagai berikut: “\$foo = array(0 => 36, 1 => 19, 2 => 27, 3 => 9, 4 => 10, 5 => 10, 6 => -89, 99 => 3);”

Skrip PHP:

```

1. <?php
2.   $foo = array(36, 19, 27, 9, 10, 10, -89);
3.   $foo[99] = 3;           // OK
4.   echo $foo[99];         // 3
5.   echo $foo[7];          // Runtime error
6. ?>

```

Gambar 2.11. Contoh indeks tak terurut dalam Array

 **Catatan:** Dokumentasi mengenai tipe data *array* dalam PHP dapat dilihat pada <http://www.php.net/manual/en/language.types.array.php>.

2.5. Operator

Operator dalam PHP meliputi operator: aritmatika, penugasan, perbandingan, logika, *increment/decrement*, *array*, dan *string*.

2.5.1. Operator Aritmatika (*Arithmetic*)

Operator aritmatika dalam PHP digunakan untuk melakukan operasi aritmatika pada angka. Daftar operator aritmatika dapat dilihat dalam Tabel 2.3.

Tabel 2.3. Daftar Operator Aritmatika dalam PHP

Simbol	Deskripsi	Contoh (\$x = 10)	Hasil
+	Penjumlahan	\$x + 5	15
-	Pengurangan	\$x - 5	5
*	Perkalian	\$x * 5	50
/	Pembagian	\$x / 5	2
%	Modulus	\$x % 5	0
**	Exponensiasi	\$x ** 5	100000

++	<i>Pre-Increment</i> (menambahkan nilai variabel dengan 1 dan kemudian mengembalikan nilai variabel)	++\$x	11
++	<i>Post-Increment</i> (mengembalikan nilai variabel dan kemudian menambahkan nilai variabel dengan 1)	\$x++	10
--	<i>Pre-Decrement</i> (mengurangi nilai variabel dengan 1 dan kemudian mengembalikan nilai variabel)	--\$x	9
--	<i>Post-Decrement</i> (mengembalikan nilai variabel dan kemudian mengurangi nilai variabel dengan 1)	\$x--	10

2.5.2. Operator Penugasan (*Assignment*)

Operator penugasan dalam PHP digunakan untuk memberikan nilai pada suatu variabel. Daftar operator penugasan dapat dilihat dalam Tabel 2.4.

Tabel 2.4. Daftar Operator Penugasan dalam PHP

Simbol	Contoh	Ekivalensi
=	\$x = \$y	\$x = \$y
+=	\$x += \$y	\$x = \$x + \$y
-=	\$x -= \$y	\$x = \$x - \$y
*=	\$x *= \$y	\$x = \$x * \$y
/=	\$x /= \$y	\$x = \$x / \$y
%=	\$x %= \$y	\$x = \$x % \$y

2.5.3. Operator Perbandingan

Operator perbandingan dalam PHP digunakan dalam pernyataan logika untuk menentukan kesamaan atau perbedaan antar variabel atau nilai. Daftar operator perbandingan dapat dilihat dalam Tabel 2.5.

Tabel 2.5. Daftar Operator Perbandingan dalam PHP

Operator	Deskripsi	Contoh (\$x = 1 dan \$y = 2)
<code>==</code>	Sama dengan	\$x == \$y bernilai <i>false</i> \$x == 1 bernilai <i>true</i> \$x == "1" bernilai <i>true</i>
<code>===</code>	Sama nilai dan sama tipe data	\$x === 1 bernilai <i>true</i> \$x === "1" bernilai <i>false</i>
<code>!=</code>	Tidak sama dengan	\$x != \$y bernilai <i>true</i> \$x != 1 bernilai <i>false</i>
<code><></code>	Tidak sama dengan	\$x <> \$y bernilai <i>true</i> \$x <> 1 bernilai <i>false</i>
<code>!==</code>	Tidak sama nilai atau tidak sama tipe data	\$x !== 1 bernilai <i>false</i> \$x !== "1" bernilai <i>true</i>
<code>></code>	Lebih besar dari	\$x > \$y bernilai <i>false</i> \$x > 1 bernilai <i>false</i> \$y > \$x bernilai <i>true</i>
<code><</code>	Lebih kecil dari	\$x < \$y bernilai <i>true</i> \$x < 1 bernilai <i>false</i> \$y < \$x bernilai <i>false</i>
<code>>=</code>	Lebih besar atau sama dengan	\$x >= \$y bernilai <i>false</i> \$x >= 1 bernilai <i>true</i> \$y >= \$x bernilai <i>true</i>
<code><=</code>	Lebih kecil atau sama dengan	\$x <= \$y bernilai <i>true</i> \$x <= 1 bernilai <i>true</i> \$y <= \$x bernilai <i>false</i>

2.5.4. Operator Logika (*Logical*)

Operator logika dalam PHP digunakan untuk menentukan logika antar variabel atau nilai. Daftar operator logika dapat dilihat dalam Tabel 2.6.

Tabel 2.6. Daftar Operator Logika dalam PHP

Operator	Nama	Deskripsi	Contoh (\$x = 1 dan \$y = 2)
AND	Logika AND	Bernilai <i>true</i> jika kedua kondisi (di sisi kiri dan kanan operator) adalah <i>true</i>	(\$x < 2 AND \$y > 1) bernilai <i>true</i>
OR	Logika OR	Bernilai <i>true</i> jika salah satu kondisi (di sisi kiri atau kanan operator) adalah <i>true</i>	(\$x < 2 OR \$y > 2) bernilai <i>true</i>
XOR	Logika XOR	Bernilai <i>true</i> jika hanya salah satu kondisi (di sisi kiri atau kanan operator) adalah <i>true</i>	(\$x < 2 XOR \$y > 1) bernilai <i>false</i>
&&	Logika AND	Bernilai <i>true</i> jika kedua kondisi (di sisi kiri dan kanan operator) adalah <i>true</i>	(\$x < 2 && \$y > 1) bernilai <i>true</i>
	Logika OR	Bernilai <i>true</i> jika salah satu kondisi (di sisi kiri atau kanan operator) adalah <i>true</i>	(\$x < 2 \$y > 2) bernilai <i>true</i>
!	Logika NOT	Kebalikan dari nilai kondisi	!(\$x > \$y) bernilai <i>true</i>

2.5.5. Operator String

Operator *string* dalam PHP adalah operator “.” yang digunakan untuk menggabungkan atau konkatenasi *string*. Daftar operator *string* dan contoh penggunaannya dapat dilihat dalam Tabel 2.7.

Tabel 2.7. Daftar Operator *String* dalam PHP

Operator	Deskripsi	Contoh
.	Konkatenasi atau menggabungkan string	\$firstName = "Almira"; \$lastName = "Wijaya"; \$name = \$firstName . " " . \$lastName; \$name bernilai "Almira Wijaya"
.=	Penugasan konkatenasi atau menggabungkan string	\$txt1 = "Hello"; \$txt1 .= " World"; \$txt1 bernilai "Hello World"

2.6. Struktur Kondisi

Terdapat dua buah strukur kondisi dalam PHP, yaitu struktur **IF** dan **SWITCH**.

2.6.1. Kondisi IF

Struktur kondisi **IF** digunakan untuk memastikan bahwa suatu perintah akan dieksekusi ketika kondisi bernilai *true*. Tambahan struktur **ELSE** diperlukan jika suatu eksekusi akan dilakukan ketika kondisi bernilai *false*. Sedangkan struktur **ELSEIF** digunakan untuk menambahkan kondisi baru ketika kondisi bernilai *false*.

Skrip PHP:

```

1. <?php
2.     $warna = "red";
3.     if ($warna == "green") {
4.         $warna = "hijau";
5.     } elseif ($warna == "blue") {
6.         $warna = "biru";
7.     } else {
8.         $warna = "merah";
9.     }
10.    echo $warna;
11. ?>
```

Gambar 2.12. Contoh kondisi IF

Berdasarkan penyeleksian kondisi dalam Gambar 2.12, maka nilai akhir dari variabel **\$warna** adalah “merah”.

 **Catatan:** Dokumentasi mengenai struktur kondisi IF dalam PHP dapat dilihat pada <https://www.php.net/manual/en/control-structures.if.php>.

2.6.2. Kondisi SWITCH

Struktur kondisi **SWITCH** digunakan untuk memberikan beberapa alternatif kondisi untuk eksekusi.

Skrip PHP:

```
1. <?php
2.     $warna = "red";
3.     switch ($warna){
4.         case "green":
5.             $warna = "hijau";
6.             break;
7.         case "blue":
8.             $warna = "biru";
9.             break;
10.        case "red":
11.            $warna = "merah";
12.            break;
13.        default:
14.            $warna="";
15.    }
16. echo $warna;
17. ?>
```

Gambar 2.13. Contoh kondisi SWITCH

Perhatikan bahwa alternatif kondisi dalam Gambar 2.13 memberikan hasil yang sama dengan dengan contoh yang diberikan pada struktur kondisi **IF** (Gambar 2.12), yaitu nilai akhir dari variabel **\$warna** adalah “merah”. Hal ini menunjukkan bahwa suatu permasalahan dengan struktur **IF** selalu dapat digunakan untuk menyelesaikan permasalahan dengan struktur **SWITCH**. Namun, ini tidak berarti bahwa struktur **SWITCH** akan selalu dapat digunakan untuk menyelesaikan permasalahan dengan struktur **IF**.

 **Catatan:** Dokumentasi mengenai struktur kondisi SWITCH dalam PHP dapat dilihat pada <https://www.php.net/manual/en/control-structures.switch.php>.

2.7. Struktur Perulangan

Struktur perulangan dalam PHP meliputi struktur perulangan **FOR**, **FOREACH**, dan **WHILE**.

2.7.1. Perulangan FOR

Struktur perulangan **FOR** digunakan untuk melakukan perulangan yang jumlahnya sudah diketahui sebelumnya.

Skrip PHP:

```

1. <?php
2.     for ($count = 1; $count <= 10; $count++)
3.     {
4.         echo "Iteration number: " . $count . "<br>";
5.     }
6. ?>

```

Gambar 2.14. Contoh perulangan FOR

Berdasarkan perulangan dalam Gambar 2.14, maka pada *web browser* akan tercetak teks dengan penomoran terurut ke atas mulai dari “Iteration number 1” sampai dengan “Iteration number 10”.



Catatan: Dokumentasi mengenai struktur control FOR dalam PHP dapat dilihat pada <https://www.php.net/manual/en/control-structures.for.php>.

2.7.2. Perulangan WHILE

Struktur perulangan **WHILE** digunakan ketika ingin melakukan perulangan selama kondisi yang ditentukan bernilai *true*. Struktur ini memiliki dua bentuk, yaitu **WHILE** dan **DO-WHILE**. Bentuk yang pertama memastikan bahwa perulangan baru akan dilakukan ketika kondisi yang ditentukan bernilai *true*, sedangkan bentuk yang kedua akan selalu setidaknya melakukan satu kali perulangan, meskipun kondisi yang ditentukan bernilai *false*.

Skrip PHP:

```

1. <?php
2.     $count = 1;
3.     while ($count <= 10)
4.     {
5.         echo "Iteration number: " . $count . "<br>";
6.         $count++;
7.     }
8. ?>

```

Gambar 2.15. Contoh perulangan WHILE**Skrip PHP:**

```

1. <?php
2.     $count = 1;
3.     do
4.     {
5.         echo "Iteration number: " . $count . "<br>";
6.         $count++;
7.     }
8.     while ($count <= 10)
9. ?>

```

Gambar 2.16. Contoh perulangan DO-WHILE

Perhatikan bahwa perulangan dalam Gambar 2.15 dan Gambar 2.16 memberikan hasil yang sama dengan contoh yang diberikan pada perulangan **FOR** (Gambar 2.14), yaitu pada *web browser* akan tercetak teks dengan penomoran terurut ke atas mulai dari “Iteration number 1” sampai dengan “Iteration number 10”. Dengan kata lain, dapat disimpulkan bahwa permasalahan yang dapat diselesaikan dengan perulangan **FOR** akan selalu dapat diselesaikan dengan menggunakan perulangan **WHILE**. Namun tidak demikian halnya dengan kebalikannya, yaitu bahwa perulangan **FOR** mungkin tidak dapat digunakan untuk menyelesaikan permasalahan yang pada perulangan **WHILE**.



Catatan: Dokumentasi mengenai struktur control WHILE dalam PHP dapat dilihat pada <https://www.php.net/manual/en/control-structures.while.php>.



Catatan: Dokumentasi mengenai struktur control DO-WHILE dalam PHP dapat dilihat pada <https://www.php.net/manual/en/control-structures.do.while.php>.

2.7.3. Perulangan FOREACH

Struktur perulangan **FOREACH** digunakan untuk melakukan iterasi pada/dengan menggunakan tipe data *array* atau *object*. Perulangan **FOREACH** pada skrip baris 2-7 dalam Gambar 2.17 memberikan hasil yang sama dengan contoh yang diberikan pada perulangan **FOR** (Gambar 2.14), **WHILE** (Gambar 2.15), dan **DO-WHILE** (Gambar 2.16). Namun perhatikan bahwa perulangan **FOREACH** memerlukan variabel **\$number** yang bertipe data *array* untuk menghasilkan hasil tersebut.

Skrip baris 9-14 dalam Gambar 2.17 memperlihatkan contoh bahwa kita dapat lebih lanjut menggunakan perulangan **FOREACH** untuk memproses tipe data konstruktur *array*, yaitu *array* yang menggunakan kombinasi indeks dan nilai, dimana indeks merupakan penunjuk posisi dimana nilai disimpan.

Skrip PHP:

```

1. <?php
2. $number = array(1,2,3,4,5,6,7,8,9,10);
3. echo "<p><b>Iteration number:</b></p>";
4. foreach ($number as $count)
5. {
6.     echo "Iteration number: " . $count . "<br>";
7. }
8.
9. $places = array(1 => 'Adi', 2 => 'Budi', 3 => 'Citra', 'last' => 'Deni');
10. echo "<p><b>Place position:</b></p>";
11. foreach ($places as $key => $val)
12. {
13.     echo "Position: $key, Athlete: $val<br>";
14. }
15. ?>

```

Hasil Eksekusi:**Iteration number:**

Iteration number: 1
 Iteration number: 2
 Iteration number: 3
 Iteration number: 4
 Iteration number: 5
 Iteration number: 6
 Iteration number: 7
 Iteration number: 8
 Iteration number: 9
 Iteration number: 10

Places position:

Position: 1, Athlete: Adi
 Position: 2, Athlete: Budi
 Position: 3, Athlete: Citra
 Position: last, Athlete: Deni

Gambar 2.17. Contoh perulangan FOREACH

Catatan: Dokumentasi mengenai struktur control FOREACH dalam PHP dapat dilihat pada <http://php.net/manual/en/control-structures.foreach.php>.

2.8. Fungsi (*Function*)

Fungsi adalah sub-program yang dibuat untuk melaksanakan suatu pekerjaan tertentu dan dapat digunakan berulang kali dalam program utama.

2.8.1. Deklarasi fungsi (*User-defined*)

Deklarasi fungsi di dalam PHP memerlukan empat komponen berikut:

- Deklarasi fungsi, yang diawali dengan penggunaan kata “function”

- Nama fungsi, yang harus diawali dengan suatu huruf atau simbol garis bawah (“_”). Nama fungsi adalah *case-insensitive*
- Daftar argumen, yang direpresentasikan melalui parameter diletakkan di dalam tanda kurung (“()”)
- Operasi yang akan dilakukan di dalam badan fungsi. Operasi yang akan dieksekusi oleh fungsi diletakkan di dalam tanda kurung kurawal (“{}”). Fungsi yang menghasilkan suatu nilai keluaran harus menggunakan perintah **return**.

Gambar 2.18 memperlihatkan sintaks deklarasi fungsi dalam PHP:

```
function namaFungsi() {  
    // operasi yang akan dilakukan  
}
```

Gambar 2.18. Sintaks deklarasi fungsi dalam PHP

Gambar 2.19 memperlihatkan sebuah contoh pembuatan fungsi dalam PHP (baris 1–5). Fungsi yang diberi nama **addNumber** dibuat untuk menghitung penjumlahan dua buah argumen yang diisikan ke dalam fungsi melalui parameter **\$value1** dan **\$value2**. Hasil penjumlahan disimpan dalam variabel **\$total** dan menjadi keluaran dari fungsi **addNumber**.

Skrip PHP:

```
1. function addNumber($value1, $value2)  
2. {  
3.     $total = $value1 + $value2;  
4.     return $total;  
5. }  
6.  
7. echo "2 + 1 = ". addNumber(1,2);
```

Gambar 2.19. Contoh fungsi dalam PHP



Catatan: Dokumentasi mengenai pembuatan fungsi dalam PHP dapat dilihat pada <http://php.net/manual/en/functions.user-defined.php>.

2.8.2. Pemanggilan fungsi

Fungsi dapat dipanggil dengan cara menyebutkan nama fungsi dan tanda kurung, beserta daftar parameternya (jika ada). Untuk contoh dalam Gambar 2.19, maka pemanggilan fungsi

addNumber dapat dilakukan dengan cara seperti yang ditunjukkan pada baris 7. Sebagai hasilnya, pada web browser akan tercetak teks “ $2 + 1 = 3$ ”.

2.9. Fungsi *Built-in*

PHP memiliki beberapa fungsi *built-in* yang di antaranya adalah bertipe *String*, *Array*, *Date/Time* dan *Math*.

2.9.1. Fungsi *String*

Beberapa fungsi bertipe *String* dalam PHP adalah: **strlen()**, **substr()**, **trim()**, **strstr()**, dan lain-lain.



Catatan: Dokumentasi mengenai fungsi tipe *String* dalam PHP dapat dilihat pada <http://php.net/manual/en/ref.strings.php>.

2.9.2. Fungsi *Array*

Beberapa fungsi bertipe *Array* dalam PHP adalah: **count()**, **sort()**, **assort()**, **ksort()**, **array_key_exists()**, **array_push()**, **array_pop()**, dan lain-lain.



Catatan: Dokumentasi mengenai fungsi tipe *Array* dalam PHP dapat dilihat pada <http://php.net/manual/en/book.array.php>.

2.9.3. Fungsi *Date/Time*

Beberapa fungsi bertipe *Date/Time* dalam PHP adalah: **checkdate()**, **date_add()**, **date_diff()**, **strtotime()**, **date()**, **getdate()**, **mktime()**, **time()**, dan lain-lain.



Catatan: Dokumentasi mengenai fungsi tipe *Date/Time* dalam PHP dapat dilihat pada <http://php.net/manual/en/book.datetime.php>.

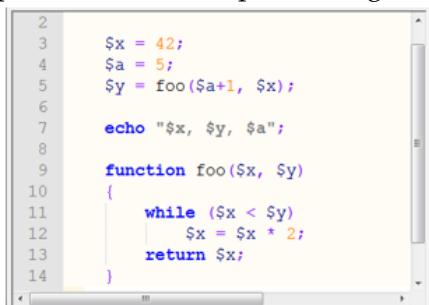
2.9.4. Fungsi *Math*

Beberapa fungsi bertipe *Math* dalam PHP adalah: ***abs()***, ***ceil()***, ***floor()***, ***log()***, ***min()***, ***max()***, ***pow()***, ***round()***, ***rand()***, ***sqrt()***, ***sin()***, ***cos()***, ***tan()***, dan lain-lain.

 **Catatan:** Dokumentasi mengenai fungsi tipe *Math* dalam PHP dapat dilihat pada <http://php.net/manual/en/book.math.php>.

2.10. Variabel Global Versus Lokal

Variabel global adalah variabel yang berlaku di program utama. Sedangkan parameter dan variabel lain yang digunakan di dalam fungsi merupakan variabel lokal dari fungsi tersebut, bahkan jika namanya sama dengan nama variabel global. Gambar 2.20 memperlihatkan contoh perbandingan variabel global dan lokal.



```

2 $x = 42;
3 $a = 5;
4 $y = foo($a+1, $x);
5
6 echo "$x, $y, $a";
7
8 function foo($x, $y)
9 {
10     while ($x < $y)
11     {
12         $x = $x * 2;
13     }
14     return $x;
}

```

Global variables:
 \$x: 42
 \$y: 48
 \$a: 5

Local variables:
 \$x: 48
 \$y: 42

Gambar 2.20. Contoh perbandingan antara variabel global dan lokal

 **Catatan:** Dokumentasi mengenai penggunaan variabel dalam PHP dapat dilihat pada <http://php.net/manual/en/language.variables.scope.php>.

2.11. Pass-by-Value Versus Pass-by-Reference

Secara *default*, parameter fungsi diaplikasikan secara “*pass-by-value*”. Namun, kita juga dapat mengaplikasikan parameter fungsi secara “*pass-by-reference*” sehingga fungsi dapat memodifikasi nilai variabel dengan cara meletakkan tanda “&” di depan parameter fungsi. Gambar 2.21 memperlihatkan contoh perbandingan antara “*pass-by-value*” (yang digunakan dalam fungsi “foo”) dan “*pass-by-reference*” (yang digunakan dalam fungsi “bar”). Perhatikan

perbedaan nilai akhir variabel global di antara kedua aplikasi tersebut.

Skrip PHP:

```

2
3     function foo($a)
4     {
5         $a[1] = 0;
6     }
7
8     function bar(&$a)
9     {
10        $a[1] = 0;
11    }
12
13    $x = array(1, 2, 3);
14
15    foo($x);
16    echo $x[1];
17
18    bar($x);
19    echo $x[1];
20

```

Nilai awal:

Global variables:
\$x: array(1, 2, 3)

<i>Pass-by-value</i> (fungsi “foo”):	<i>Pass-by-reference</i> (fungsi “bar”):
Local variables: \$a: array(1, 2, 3)	Local variables: \$a: array(1, 2, 3)
Local variables: \$a: array(1, 0, 3)	Local variables: \$a: array(1, 0, 3)
Global variables: \$x: array(1, 2, 3)	Global variables: \$x: array(1, 0, 3)

Gambar 2.21. Contoh perbandingan antara “pass-by-value” dan “pass-by-reference”

 **Catatan:** Dokumentasi mengenai “pass-by-reference” dalam PHP dapat dilihat pada <http://php.net/manual/en/language.references.pass.php>.

2.12. Penggunaan *Include*

Include dapat digunakan dalam PHP untuk menghindari pengulangan blok skrip. Gambar 2.22 memperlihatkan contoh dua buah file PHP yang terlihat memiliki blok skrip yang sama. Blok skrip yang sama, yaitu skrip baris 15-26 pada file “page1.php” dan skrip baris 18-29 pada file “page2.php”, dapat disalin tempel pada sebuah file **include** yang diberi nama “menu.inc” seperti yang diperlihatkan

dalam Gambar 2.23. Dan selanjutnya kita tinggal memanggil nama file **include** pada file PHP. Gambar 2.24. memperlihatkan implementasi file “menu.inc” pada file “page1.php” (skrip baris 16) dan “page2.php” (skrip baris 19).

```

1 <!DOCTYPE html>
2 <html xmlns="http://www.w3.org/1999/xhtml">
3 <head>
4   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
5   <meta name="Keywords" content="Arama Little Athletics, Little Athletics" />
6   <link rel="stylesheet" type="text/css" href="styles/page.layout.css" />
7   <link rel="stylesheet" type="text/css" href="styles/menu.css" />
8   <link rel="stylesheet" type="text/css" href="styles/general.css" />
9   <title>Arama Little Athletics - Home Page</title>
10  <meta name="Description" content="Arama Little Athletics was established in 1970 to provide a safe environment for children to learn and develop their athletic abilities." />
11  <meta name="Keywords" content="Arama Little Athletics, Little Athletics" />
12  <link rel="stylesheet" type="text/css" href="scripts/register.css" />
13  <script src="scripts/register.js" type="text/javascript"></script>
14 </head>
15 <body>
16   <div id="main" class="centre">
17     
18     <ul id="nav-container">
19       <li><a href="#">Home</a> </li>
20       <li><a href="#">RegisterMember.php>Join</a> </li>
21       <li><a href="#">ListEvents.php>Results</a> </li>
22       <li><a href="#">Competitions</a> </li>
23       <li><a href="#">Weather</a> </li>
24       <li><a href="#">News</a> </li>
25       <li><a href="#">Coaching</a> </li>
26       <li><a href="#">Parents</a> </li>
27       <li><a href="#">Committee</a> </li>
28       <li><a href="#">Contact Us</a> </li>
29     </ul>
30     <div id="right-colour">
31       <div id="reset">
32         <div id="left">
33           <a href="http://www.glesa.asn.au/">
34             
35           </a>
36         </div>
37         <div id="right">
38           <div id="never-heading">
39             Latest News
40           </div>
41           <div id="news-body">
42             <p>Nominations close for Junior Carnival/Gen...
43           </p>
44         </div>
45       </div>
46     </div>
47   </div>
48 </body>
49 </html>

```

```

1 <!DOCTYPE html>
2 <html xmlns="http://www.w3.org/1999/xhtml">
3 <head>
4   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
5   <meta name="Keywords" content="Arama Little Athletics, Little Athletics" />
6   <link rel="stylesheet" type="text/css" href="styles/page.layout.css" />
7   <link rel="stylesheet" type="text/css" href="styles/menu.css" />
8   <link rel="stylesheet" type="text/css" href="styles/general.css" />
9   <title>Arama Little Athletics - Register Page</title>
10  <meta name="Description" content="Arama Little Athletics was established in 1970 to provide a safe environment for children to learn and develop their athletic abilities." />
11  <meta name="Keywords" content="Arama Little Athletics, Little Athletics" />
12  <link rel="stylesheet" type="text/css" href="scripts/forms.css" />
13  <script src="scripts/register.js" type="text/javascript"></script>
14 </head>
15 <body>
16   <div id="main" class="centre">
17     
18     <ul id="nav-container">
19       <li><a href="#">Index.php>Home</a> </li>
20       <li><a href="#">ListEvents.php>Results</a> </li>
21       <li><a href="#">Competitions</a> </li>
22       <li><a href="#">Weather</a> </li>
23       <li><a href="#">News</a> </li>
24       <li><a href="#">Coaching</a> </li>
25       <li><a href="#">Parents</a> </li>
26       <li><a href="#">Committee</a> </li>
27       <li><a href="#">Contact Us</a> </li>
28     </ul>
29     <div id="right-colour">
30       <div id="reset">
31         <div id="left">
32           <a href="http://www.glesa.asn.au/">
33             
34           </a>
35         </div>
36         <div id="right">
37           <h3>Registration Form</h3>?<form action="RegisterMember.i...
38             <fields>
39               <label>Athlete Information</label>
40             </fields>
41           </form>
42         </div>
43       </div>
44     </div>
45   </div>
46 </body>
47 </html>

```

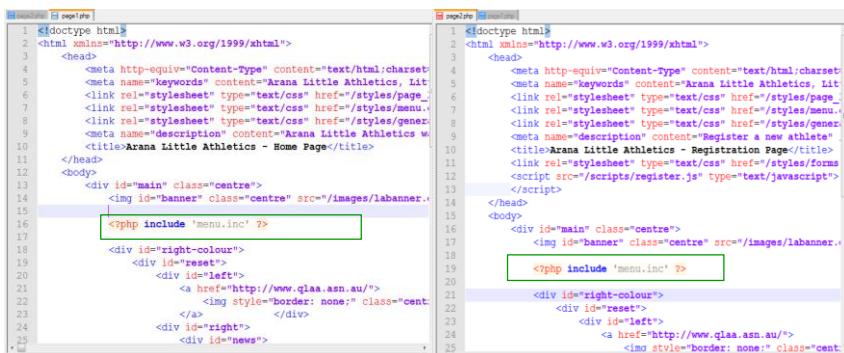
Gambar 2.22. Contoh dua buah file PHP (“page1.php” dan “page2.php”) yang memiliki blok skrip yang sama

```

1 <ul id="nav-container">
2   <li><a href="#">Home</a> </li>
3   <li><a href="#">RegisterMember.php>Join</a> </li>
4   <li><a href="#">ListEvents.php>Results</a> </li>
5   <li><a href="#">Competitions</a> </li>
6   <li><a href="#">Weather</a> </li>
7   <li><a href="#">News</a> </li>
8   <li><a href="#">Coaching</a> </li>
9   <li><a href="#">Parents</a> </li>
10  <li><a href="#">Committee</a> </li>
11  <li><a href="#">Contact Us</a> </li>
12 </ul>

```

Gambar 2.23. Contoh pembuatan file “menu.inc” berdasarkan blok skrip yang sama dalam file “page1.php” dan “page2.php”



```

1 <!DOCTYPE html>
2 <html xmlns="http://www.w3.org/1999/xhtml">
3 <head>
4   <meta http-equiv="Content-Type" content="text/html;charset=UTF-8" />
5   <meta name="keywords" content="Arana Little Athletics, Ltd." />
6   <link rel="stylesheet" type="text/css" href="/styles/page_.css" />
7   <link rel="stylesheet" type="text/css" href="/styles/menu_.css" />
8   <link rel="stylesheet" type="text/css" href="/styles/general.css" />
9   <meta name="description" content="Arana Little Athletics website" />
10  <title>Arana Little Athletics - Home Page</title>
11 </head>
12 <body>
13   <div id="main" class="centre">
14     
15     <?php include 'menu.inc' ?>
16 
17     <div id="right-colour">
18       <div id="reset">
19         <div id="left">
20           <a href="http://www.qlaa.asn.au/">
21             <img style="border: none;" class="cent" alt="QLA logo" />
22           </a>
23         </div>
24         <div id="right">
25           <div id="news">

```



```

1 <!DOCTYPE html>
2 <html xmlns="http://www.w3.org/1999/xhtml">
3 <head>
4   <meta http-equiv="Content-Type" content="text/html;charset=UTF-8" />
5   <meta name="keywords" content="Arana Little Athletics, Ltd." />
6   <link rel="stylesheet" type="text/css" href="/styles/page_.css" />
7   <link rel="stylesheet" type="text/css" href="/styles/menu_.css" />
8   <link rel="stylesheet" type="text/css" href="/styles/general.css" />
9   <meta name="description" content="Register a new athlete" />
10  <title>Arana Little Athletics - Registration Page</title>
11 </head>
12 <body>
13   <div id="main" class="centre">
14     
15     <?php include 'menu.inc' ?>
16 
17     <div id="right-colour">
18       <div id="reset">
19         <div id="left">
20           <a href="http://www.qlaa.asn.au/">
21             <img style="border: none;" class="cent" alt="QLA logo" />
22           </a>
23         </div>
24         <div id="right">
25           <div id="news">

```

Gambar 2.24. Contoh penggunaan file “menu.inc” pada file “page1.php” dan “page2.php”

 **Catatan:** Dokumentasi mengenai *include* dalam PHP dapat dilihat pada <https://www.php.net/manual/en/function.include.php>.

 **Catatan:** Dokumentasi mengenai alternatif penggunaan *include* dalam PHP adalah *include_once* (<https://www.php.net/manual/en/function.include-once.php>), *require* (<https://www.php.net/manual/en/function.require.php>), dan *require_once* (<https://www.php.net/manual/en/function.require-once.php>).

2.13. Pemrograman Berbasis Obyek

Pemrograman berbasis obyek (PBO) atau *Object Oriented Programming* (OOP) adalah pemrograman yang memecahkan permasalahan yang ingin diselesaikan dengan menggunakan konsep *obyek* yang memiliki atribut data (*atribut* yang menjelaskan tentang obyek) dan prosedur (*function*) yang dikenal dengan istilah *method*. Pada dasarnya, PHP menerapkan konsep pemrograman prosedural. Namun, PHP juga menyediakan fitur tambahan PBO. Buku ini tidak membahas bagaimana pembuatan PBO dalam PHP dan hanya fokus konsep pemrograman prosedural.

 **Catatan:** Dokumentasi mengenai pemrograman berbasis obyek dalam PHP dapat dilihat pada <http://php.net/manual/en/language.oop5.php>.

2.14. Latihan Soal



Latihan 2.1

Membuat program PHP yang dapat menampilkan “Tabel Perkalian 12x12” seperti yang diperlihatkan dalam Gambar 2.25.

Tabel Perkalian 12x12

	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	3	4	5	6	7	8	9	10	11	12
2	2	4	6	8	10	12	14	16	18	20	22	24
3	3	6	9	12	15	18	21	24	27	30	33	36
4	4	8	12	16	20	24	28	32	36	40	44	48
5	5	10	15	20	25	30	35	40	45	50	55	60
6	6	12	18	24	30	36	42	48	54	60	66	72
7	7	14	21	28	35	42	49	56	63	70	77	84
8	8	16	24	32	40	48	56	64	72	80	88	96
9	9	18	27	36	45	54	63	72	81	90	99	108
10	10	20	30	40	50	60	70	80	90	100	110	120
11	11	22	33	44	55	66	77	88	99	110	121	132
12	12	24	36	48	60	72	84	96	108	120	132	144

Gambar 2.25. Tabel perkalian 12x12



Latihan 2.2

Membuat program PHP untuk menampilkan “Kalender Bulan Ini” dan berikan penanda khusus untuk tanggal hari ini. Contoh dalam Gambar 2.26 memperlihatkan contoh kalender bulan September 2019. Tanggal 1 September 2019 diberi pendanda khusus karena aplikasi web dieksekusi pada tanggal tersebut.

September 2019

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

@University of Trunojoyo Madura

Gambar 2.26. Kalender bulan ini



Latihan 2.3

Gunakan perulangan **FOR** pada PHP untuk mencetak teks dengan penomoran terurut ke bawah pada *web browser*, yaitu dari “Iteration number 10” hingga “Iteration number 1”!



Latihan 2.4

Gunakan perulangan **WHILE** pada PHP untuk menyelesaikan permasalahan pada Latihan 2.3!



Latihan 2.5

Gunakan perulangan **DO-WHILE** pada PHP untuk menyelesaikan permasalahan pada Latihan 2.3!



Latihan 2.6

Membuat program PHP dengan menggunakan **FOREACH** untuk menampilkan “Daftar Buah” seperti yang diperlihatkan dalam Gambar 2.27(b) berdasarkan skrip HTML yang diperlihatkan dalam Gambar 2.27(a).

```
<h1>Daftar Buah:</h1>
<ul>
    <li>Apel</li>;
    <li>Jeruk</li>;
    <li>Pisang</li>;
</ul>
```

(a)

Daftar Buah:

- Apel
- Jeruk
- Pisang

(b)

Gambar 2.27. “Daftar Buah”: (a) Skrip HTML dan (b) Hasil tampilan

MNC Publishing

BAB 3.

IMPLEMENTASI PHP

Bab ini membahas implementasi PHP yang digunakan untuk:

- Pemrosesan *file*
- Pemrosesan *form*
- Validasi *server-side*

Setelah selesai mempelajari bab ini maka diharapkan kita dapat mendemonstrasikan implementasi PHP tersebut di atas untuk pengembangan *web* dinamis.

3.1. Pemrosesan File

File merupakan tempat untuk menyimpan data sekunder secara permanen di dalam suatu media penyimpanan, seperti *hardisk* dan *flashdisk*. Secara umum, format *file* dapat berupa teks atau *binary*. Pengaksesan file dapat dilakukan dalam tiga tahapan, yaitu:

1. Membuka *file*
2. Memproses *file*/manipulasi *file*
3. Menutup *file*

Tabel 3.. memperlihatkan daftar fungsi yang dapat digunakan untuk pemrosesan file.

Tabel 3.1. Daftar fungsi untuk pemrosesan file

Fungsi	Deskripsi
fopen()	membuka/mengakses <i>file</i> (fungsi ini dapat mengakses <i>file</i> dari sistem <i>file</i> , atau melalui HTTP atau FTP di internet)
fclose()	menutup <i>file</i> digunakan
fgets()	melihat isi dari <i>file</i> yang digunakan
fputs()	memasukkan data ke dalam <i>file</i>
feof()	menentukan akhir dari sebuah <i>file</i> (Jika sudah pada akhir <i>file</i> , fungsi ini akan bernilai TRUE)

copy()	mengkopi suatu <i>file</i> (<i>file</i> yang akan dikopi dapat diambil melalui masukan data <i>form</i>)
unlink()	menghapus <i>file</i> (bernilai TRUE jika <i>file</i> berhasil dihapus dan FALSE jika terjadi kegagalan dalam proses penghapusan)
rename()	mengganti nama <i>file</i> (bernilai TRUE jika nama file berhasil diubah dan FALSE jika terjadi kegagalan dalam proses pengubahan)
file_exists()	memeriksa kondisi suatu <i>file</i> , apakah file tersebut ada atau tidak (bernilai TRUE jika <i>file</i> ada dan FALSE jika <i>file</i> tidak ada)
filesize()	mengetahui besar ukuran suatu <i>file</i> digunakan (hasil bertipe integer, yang menyatakan ukuran <i>file</i> dalam satuan byte)

Gambar 3.1 memperlihatkan contoh pemrosesan *file* yang digunakan untuk pembuatan *counter*. Fungsi *counter* di sini adalah untuk menghitung berapa kali suatu halaman *web* telah dibuka oleh pengunjung. Pembuatan *counter* dapat dilakukan dengan membuat dua buah file berikut:

1. *File* teks yang misalnya diberi nama “counter.txt”. Fungsi dari *file* ini adalah untuk:
 - menginisialisasi nilai *counter* dengan nilai 0
 - dijadikan acuan nilai *counter* sehingga setiap kali halaman *web* dibuka/ditampilkan maka isi dari file “counter.txt” akan dibaca nilainya dan kemudian diperbarui (ditambahkan nilainya dengan 1)
2. *File* php yang misalnya diberi nama “counter.php”. Fungsi dari *file* ini adalah untuk:
 - membaca nilai di dalam *file* teks (baris 2-5)
 - memperbarui nilai *counter* (baris 6)
 - menyimpan nilai yang baru di *file* “counter.txt” (baris 7-9)
 - menampilkan nilainya di *web browser* (baris 10)

Skrip PHP (counter.php):

```

1. <?php
2.     $filecounter="counter.txt"; // menentukan file yang akan diproses
3.     $open=fopen($filecounter,'r+'); // membuka file
4.     $counter=fgets($open); // melihat isi file
5.     fclose($open); // menutup file
6.     $counter++; // memanipulasi isi file
7.     $write=fopen($filecounter,'w'); // membuka file
8.     fputs($write,$counter); // memasukkan data ke dalam file
9.     fclose($write); // menutup file
10.    echo "<b> Anda adalah pengunjung ke : $counter</b>"; //mencetak hasil counter
11. ?>

```

Skrip TXT (counter.txt):

```
1. 0
```

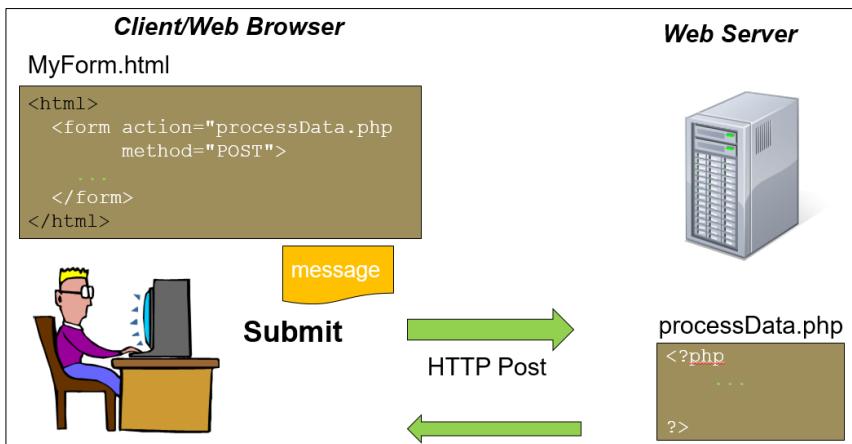
Hasil Eksekusi (setelah 3 kali percobaan):

Anda adalah pengunjung ke : 3

Gambar 3.1. Contoh pemrosesan *file* yang digunakan untuk pembuatan *counter*

3.2. Pemrosesan *Form*

Form dan PHP adalah dua hal yang tidak dapat dipisahkan karena lazimnya PHP digunakan untuk memroses *form*. Gambar 3.2. memperlihatkan alur pemrosesan *form* dalam PHP. Alur diawali oleh *user* dari sisi *client* yang menggunakan *web browser* untuk mengisi *form* yang berada dalam *file* "myForm.html". Ketika user mengirimkan *form*, proses berlanjut ke sisi *server* dengan menggunakan HTTP Post (berdasarkan atribut method yang ditentukan dalam *file* "myForm.html") agar diproses lebih lanjut pada *file* "processData.php" (berdasarkan atribut action yang ditentukan dalam *file* "myForm.html"). Pada akhirnya, hasil dari *file* "processData.php" akan dikembalikan atau ditampilkan ke *web browser* kepada *user*.



Gambar 3.2. Alur pemrosesan *form* dalam PHP

Catatan: Pemrosesan *server-side* dilakukan berdasarkan atribut *name* yang digunakan dalam file HTML. Atribut *id* dalam file HTML hanya digunakan pada pemrosesan *client-side*.

Berikut ini kita akan membuat contoh pemrosesan *form* yang digunakan untuk menampilkan hasil isian data pribadi seseorang. Pemrosesan *form* isian data pribadi dapat dilakukan dengan membuat dua buah file berikut:

1. File HTML yang misalnya diberi nama "processData_form.HTML" (Gambar 3.3). Fungsi dari file ini adalah untuk:
 - menentukan bahwa proses akan dilanjutkan ke "processData.php" dan bahwa HTTP method yang digunakan adalah "POST" (baris 17)
 - membuat *form* yang digunakan untuk mengisi data yang terdiri dari masukan:
 - berbentuk *textbox* untuk isian "Surname" (baris 18-21)
 - berbentuk *textbox* untuk isian "Email address" (baris 22-25)
 - berbentuk *password* untuk isian "Password" (baris 26-29)
 - berbentuk *textarea* untuk isian "Street Address" (baris 30-33)
 - berbentuk *dropdown menu* untuk pilihan "State" (baris 34-47)

- data *hidden* untuk data “country” yang diisi dengan teks “Australia” (baris 48-50)
 - berbentuk *radio button* untuk pilihan “Gender” (baris 51-55)
 - berbentuk *checkbox* untuk isian “Vegetarian?” (baris 56-59)
 - berbentuk tombol *Submit* untuk mengirimkan *form* dan tombol *Reset* untuk menghapus isian *form* (baris 60-63)
2. File PHP yang misalnya diberi nama “processData.php” (Gambar 3.4). Fungsi dari file ini adalah untuk:
- menampilkan isian *form* yang dikirimkan dari file “processData_form.HTML” dengan memanfaatkan perulangan **FOREACH** untuk memproses tipe data konstruktor *array* (baris 16-17)

Skrip HTML (processData_form.HTML):

```

16 <legend>Person Details</legend>
17 <form name="processData" action="processData.php" method="POST">
18   <div class="field">
19     <label for="surname">Surname</label>
20     <input type="text" name="surname" size="31"/>
21   </div>
22   <div class="field">
23     <label for="email">Email address</label>
24     <input type="text" name="email" size="31"/>
25   </div>
26   <div class="field">
27     <label for="password">Password</label>
28     <input type="password" name="password" size="31"/>
29   </div>
30   <div class="field">
31     <label for="address">Street Address</label>
32     <textarea name="address" rows="3" cols="22"></textarea>
33   </div>
34   <div class="field">
35     <label for="state">State</label>
36     <select name="state">
37       <option value="1">Australian Capital Territory</option>
38       <option value="2">Queensland</option>
39       <option value="3">Victoria</option>
40       <option value="4">South Australia</option>
41       <option value="5">Tasmania</option>
42       <option value="6">North Australia</option>
43       <option value="7">Western Australia</option>
44       <option value="8">Northern Territory</option>
45     </select>
46   </div>
47   <div class="field">
48     <input type="hidden" name="country" value="Australia">
49   </div>
50   <div class="field">
51     <label for="gender">Gender</label>
52     <input type="radio" name="sex" value="male"/>Male
53     <input type="radio" name="sex" value="female"/>Female
54   </div>
55   <div class="field">
56     <label for="vegetarian">Vegetarian?</label>
57     <input type="checkbox" name="vegetarian" />
58   </div>
59   <div class="field">
60     <label for="submit">Submit</label>
61     <input type="submit" value="Submit" name="submit" /><input type="reset" value="Reset" name="reset" />
62   </div>
63 </form>
64 </fieldset>

```

Hasil Eksekusi:

Gambar 3.3. Contoh potongan skrip file “processData_form.html” yang digunakan untuk membuat *form*

<p>Skrip PHP (processData.php):</p> <pre> 14 <h3>Posted data: </h3> 15 <?php 16 foreach (\$_POST as \$key => \$value) 17 echo "\$key => (\$value)
"; 18 19 // \$surname = \$_POST['surname']; 20 ?> </pre>	<p>Hasil Eksekusi:</p> <pre> Posted data: (surname) => (Smith) (email) => (j.smith@gmail.com) (passwd) => (123456) (address) => (126 Margaret Street Brisbane 4000) (state) => (2) (country) => (Australia) (sex) => (male) (submit) => (Submit) </pre>
---	---

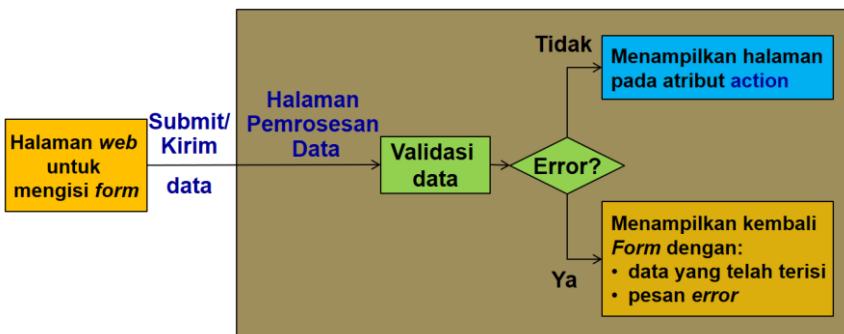
Gambar 3.4. Contoh potongan skrip file “processData.php” yang digunakan untuk menampilkan hasil isian form

3.3. Validasi *Server-Side*

Validasi *server-side* adalah proses validasi yang dilakukan di sisi *server*. Suatu aplikasi *web* harus menerapkan validasi *server-side* dan tidak hanya mengandalkan validasi *client-side* karena:

- *User* mungkin tidak mengaktifkan JavaScript pada *web browser* yang digunakan
- *Hacker* dapat menuliskan program untuk mengirimkan pesan HTTP *request* ke halaman PHP

Gambar 3.5. memperlihatkan alur validasi *server-side* dalam PHP. Alur diawali oleh *user* yang menggunakan *web browser* untuk mengisi *form* pada halaman *web* dan kemudian mengirimkannya. Selanjutnya, halaman pemrosesan data akan melakukan validasi data. Jika data yang dikirimkan tidak memiliki kesalahan (*error*), maka *web browser* akan menampilkan halaman *web* berdasarkan nilai dari atribut *action* pada *form*. Namun jika data yang dikirimkan memiliki kesalahan (*error*), maka *web browser* akan menampilkan kembali *form* beserta isian datanya dan penjelasan kesalahan apa (saja) yang telah dilakukan oleh *user* dalam mengisi *form*.

Gambar 3.5. Alur validasi *server-side* dalam PHP

Gambar 3.6. memperlihatkan contoh potongan skrip file “basicValidation.php” yang digunakan untuk memvalidasi isian form. Baris 5 menunjukkan bahwa file ini memerlukan penggunaan file “validate.inc” sebanyak satu kali, dimana di dalamnya terdapat skrip fungsi “validatePattern” (Gambar 3.7). Baris 8 memperlihatkan pemanggilan fungsi “validatePattern” yang digunakan untuk memvalidasi atau memberikan ketentuan bahwa isian “name” harus berupa huruf alfabet. Sedangkan baris 8 memperlihatkan pemanggilan fungsi “validatePattern” yang digunakan untuk memvalidasi atau memberikan ketentuan bahwa isian “phone” harus berupa kombinasi digit angka tertentu.

```

basicValidation.php
4  <?php
5   require_once 'validate.inc';
6
7   $errors = array();
8   validatePattern($errors, $_POST, 'name', '/^[a-zA-Z]+$/');
9   validatePattern($errors, $_POST, 'phone', '/^((\d{4}\s|\s\d{4})|(\d{2}\s|\s\d{2})\d{3})$/');
10  validateDate($errors, $_POST, 'dob');
11  // ...
12
13  if (count($errors) == 0)
14  {
15      // Data validates so continue with processing
16      // ...
17  }
18  else
19  {
20      // Data doesn't validate, so display error message
21      // ...
22  }
?>
  
```

Gambar 3.6. Contoh potongan skrip file “basicValidation.php” yang digunakan untuk memvalidasi isian form

```

11     function validatePattern(&$errors, $field_list, $field_name, $pattern)
12     {
13         if (!isset($field_list[$field_name]) || $field_list[$field_name] == '')
14             $errors[$field_name] = 'Required';
15         else if (!preg_match($pattern, $field_list[$field_name]))
16             $errors[$field_name] = 'Invalid';
17     }

```

Gambar 3.7. Contoh potongan skrip fungsi “validatePattern” dalam file “validate.inc” yang dipanggil dalam file “basicValidation.php”

Tabel 3.2 memperlihatkan daftar contoh fungsi built-in yang dapat digunakan untuk melakukan validasi.

Tabel 3.2. Contoh Fungsi Built-in untuk validasi menggunakan PHP

Jenis Fungsi	Contoh Fungsi Built-in
<i>Regular expressions</i>	pregmatch
<i>String</i>	trim, strtolower, strtoupper
<i>Filter</i>	filter_var, filter_input, FILTER_VALIDATE_EMAIL, FILTER_VALIDATE_URL, FILTER_VALIDATE_FLOAT, FILTER_VALIDATE_IP
<i>Type testing</i>	is_float, is_int, is_numeric, is_string
<i>Date</i>	checkdate

3.4. Latihan Soal



Latihan 3.1

Implementasikan contoh pemrosesan *file* yang digunakan untuk pembuatan *counter* seperti yang diperlihatkan dalam Gambar 3.1!



Latihan 3.2

Implementasikan contoh pemrosesan *form* yang digunakan untuk menampilkan hasil isian data pribadi seseorang seperti yang diperlihatkan dalam Gambar 3.3 dan Gambar 3.4!



Latihan 3.3

Implementasikan contoh validasi isian *form* seperti yang diperlihatkan dalam Gambar 3.6 dan Gambar 3.7!



Latihan 3.4

Implementasikan contoh validasi isian *form* untuk menyelesaikan permasalahan pada Latihan 3.1 dengan menerapkan konsep *Self-submission*! Self-submission adalah mengirimkan isian *form* ke *file* yang berisi skrip *form* itu sendiri (nilai atribut *action* adalah nama file itu sendiri)

MNC Publishing

BAB 4.

SISTEM BASISDATA DAN APLIKASI WEB

Bab ini membahas mengenai keterkaitan dan implementasi sistem basisdata dan aplikasi *web*. Setelah selesai mempelajari bab ini maka diharapkan kita dapat mendemonstrasikan pengembangan *web* dinamis yang terkoneksi dengan basisdata.

4.1. Sistem Basisdata MySQL/MariaDB

MySQL adalah sistem basisdata *open source* yang paling populer digunakan. Sistem ini pertama kali dikembangkan oleh Michael Widenius & David Axmark pada tahun 1994 dan kemudian diakusisi oleh Sun (sekarang dikenal sebagai Oracle) pada tahun 2008 (<http://www.mysql.com/>). MariaDB sendiri merupakan sistem basisdata yang dikembangkan pengembang MySQL. Oleh karena itu MySQL dan MariaDB terkadang dianggap sebagai sistem basisdata yang sama. Kedua sistem ini dapat secara bebas digunakan pada berbagai sistem operasi seperti Windows, Linux, Mac dan Unix. Gambar 4.1. memperlihatkan logo MySQL.

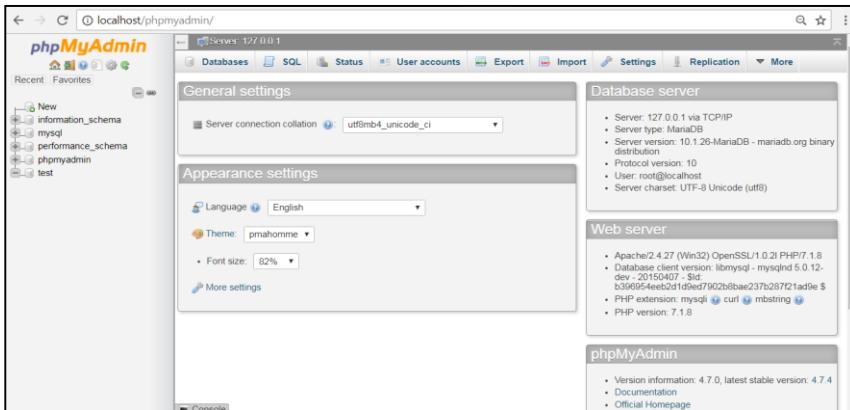


Gambar 4.1. Logo MySQL

4.1.1. Manajemen basisdata menggunakan PHPMyAdmin

PHPMyAdmin adalah aplikasi manajemen sistem basisdata yang terintegrasi dalam instalasi XAMPP dan dibuat khusus untuk

mengelola basisdata MySQL/MariaDB. Gambar 4.2. memperlihatkan tampilan halaman utama PHPMyAdmin.

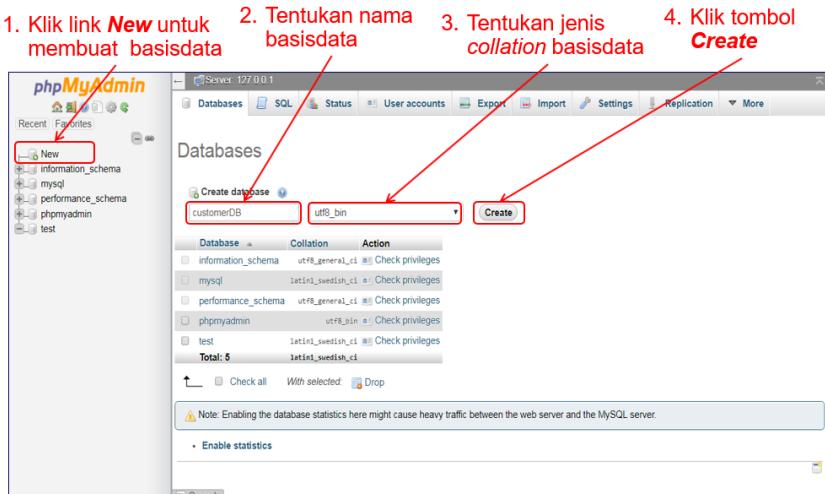


Gambar 4.2. Halaman utama PHPMyAdmin

4.1.2. Membuat (*Create*) Basisdata

Membuat basisdata merupakan langkah pertama yang harus dilakukan agar kita dapat memanfaatkan aplikasi manajemen basisdata lebih lanjut. Gambar 4.3. memperlihatkan langkah-langkah yang harus dilakukan ketika ingin membuat suatu basisdata baru dengan menggunakan PHPMyAdmin, yaitu:

1. Meng-klik **link New** untuk mulai membuat basisdata
2. Mengisikan nama basisdata yang akan dibuat ke dalam *textbox* yang telah disediakan. Dalam contoh ini, kita akan membuat basisdata baru bernama “customerDB”
3. Menentukan jenis *collation* dari basisdata yang akan dibuat. Dalam hal ini, “utf8_bin” adalah jenis *collation* yang umum untuk digunakan
4. Meng-klik tombol **Create** untuk mengeksekusi pembuatan basisdata.



Gambar 4.3. Membuat basisdata “customerDB” menggunakan PHPMyAdmin

4.1.3. Membuat (*Create*) Tabel

Setelah basisdata “customerdb” dibuat, maka selanjutnya kita dapat mulai membuat tabel yang dapat digunakan untuk menyimpan data. Gambar 4.4. memperlihatkan langkah-langkah yang harus dilakukan ketika ingin membuat suatu tabel baru dengan menggunakan PHPMyAdmin, yaitu:

1. Memilih basisdata yang akan digunakan. Dalam contoh ini tentu saja kami memilih basisdata “customerdb” yang telah dibuat sebelumnya
2. Menentukan nama tabel yang akan dibuat. Dalam contoh ini, kita akan membuat tabel baru bernama “customer”
3. Menentukan jumlah kolom dalam tabel
4. Meng-klik tombol **Go** untuk lanjut ke tahap pendefinisian struktur kolom secara lebih detil.



Gambar 4.4. Membuat tabel “customer” menggunakan PHPMyAdmin

Sebagai kelanjutannya, Gambar 4.5 memperlihatkan langkah-langkah yang harus dilakukan ketika ingin mendefinisikan struktur kolom pada tabel dengan menggunakan PHPMyAdmin, yaitu:

1. Melengkapi spesifikasi keempat kolom yang akan dibuat dalam tabel “customer”
2. Meng-klik tombol **Preview SQL** untuk mengecek skrip SQL pembuatan tabel “customer” secara lengkap. Tahap ini bermanfaat untuk memastikan bahwa hasil struktur tabel nantinya akan benar-benar sesuai dengan yang kita inginkan
3. Meng-klik tombol **Save** untuk mengeksekusi pembuatan tabel.

5. Lengkapi spesifikasi setiap kolom

7. Klik tombol Save

6. Cek skrip SQL pembuatan tabel

```

CREATE TABLE `customerdb`.`Customer` (
  `customerID` INT NOT NULL AUTO_INCREMENT,
  `firstname` VARCHAR(45) NOT NULL,
  `address` VARCHAR(256) NULL,
  `balance` DECIMAL NOT NULL,
  PRIMARY KEY (`customerID`) ENGINE =InnoDB;
  
```

Gambar 4.5. Mendefinisikan struktur kolom pada tabel “customer” menggunakan PHPMyAdmin

Gambar 4.6. memperlihatkan struktur tabel “customer” sebagai hasil dari tahapan-tahapan yang telah dilakukan sebelumnya.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	customerID	int(11)	utf8_bin	No	None			AUTO_INCREMENT	Primary Index More
2	firstname	varchar(45)	utf8_bin	No	None				Primary Index More
3	address	varchar(256)	utf8_bin	Yes	NULL				Primary Index More
4	balance	decimal(10,0)		No	None				Primary Index More

Struktur tabel **customer** dalam basisdata **customerdb**

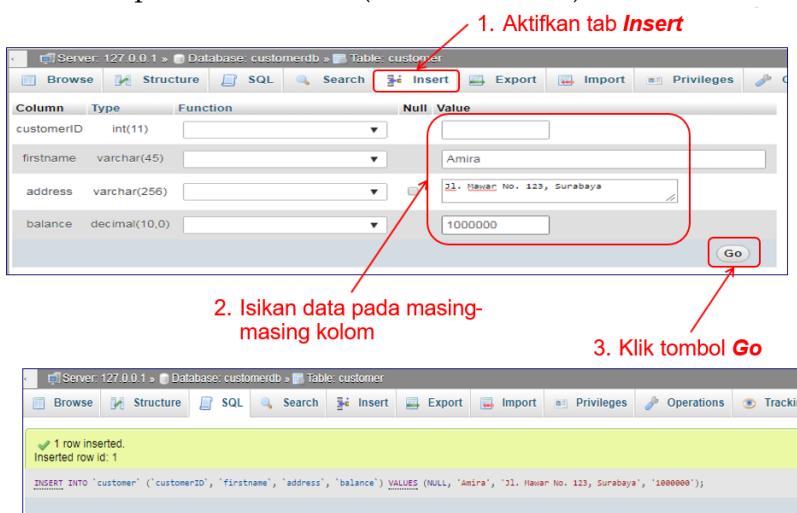
Gambar 4.6. Melihat struktur tabel “customer” menggunakan PHPMyAdmin

4.1.4. Memasukkan (*Insert*) Data Tabel

Setelah tabel “customer” dibuat, maka selanjutnya kita dapat mulai mengisi data pada tabel tersebut. Gambar 4.7. memperlihatkan langkah-langkah yang harus dilakukan ketika ingin memasukkan data baru ke tabel “customer” dengan menggunakan PHPMyAdmin, yaitu:

PENGEMBANGAN APLIKASI WEB

1. Aktifkan tab Insert untuk membuka tab pengisian data. Namun sebelumnya pastikan bahwa tabel yang aktif pada window adalah tabel “customer”
2. Isikan data yang ingin diisi pada masing-masing kolom pada *textbox* yang sudah tersedia
3. Meng-klik tombol Go untuk mengeksekusi penambahan data pada tabel “customer”. Selanjutnya akan muncul notifikasi ketika penambahan data (“1 row inserted”) berhasil dilakukan.



Gambar 4.7. Memasukkan data baru pada tabel “customer” menggunakan PHPMyAdmin

Gambar 4.8. memperlihatkan bahwa kita dapat melihat isi data pada tabel “customer” dengan cara mengaktifkan tab **Browse**.

customerID	firstname	address	balance
1	Amira	Jl. Mawar No. 123, Surabaya	1000000

Gambar 4.8. Melihat isi data pada tabel “customer” menggunakan PHPMyAdmin

4.1.5. Akun dan Hak Akses (*Priviledge*) User

Kita juga dapat menggunakan PHPMyAdmin untuk melihat daftar akun yang memiliki akses ke dalam sistem basisdata. Gambar 4.9. memperlihatkan bahwa kita melihat daftar akun yang memiliki akses ke dalam sistem dengan mengaktifkan tab **User accounts**. Selanjutnya kita dapat lebih detil mengecek akses dengan meng-klik link **Edit privileges**. Kita juga dapat menambah akun baru dengan cara meng-klik link **Add user account**.

Aktifkan tab **User accounts**

User name	Host name	Password	Global privileges	User group	Grant	Action
Any	%	No	USAGE		No	Edit privileges Export
Any	localhost	No	USAGE		No	Edit privileges Export
pma	localhost	No	USAGE		No	Edit privileges Export
root	127.0.0.1	No	ALL PRIVILEGES		Yes	Edit privileges Export
root	::1	No	ALL PRIVILEGES		Yes	Edit privileges Export
root	localhost	No	ALL PRIVILEGES		Yes	Edit privileges Export

New [Add user account](#)

[Remove selected user accounts](#)

(Revoke all active privileges from the users and delete them afterwards.)
 Drop the databases that have the same names as the users.

Gambar 4.9. Melihat daftar akun yang memiliki akses ke dalam sistem menggunakan PHPMyAdmin

4.1.6. Structured Query Language (SQL)

PHPMyAdmin juga memberikan pilihan kepada kita untuk mengetikkan perintah SQL secara manual. Gambar 4.10 memperlihatkan contoh membuat SQL untuk melakukan perintah SELECT, UPDATE, DELETE, dan INSERT pada tabel “customer” dengan cara mengaktifkan tab **SQL**.



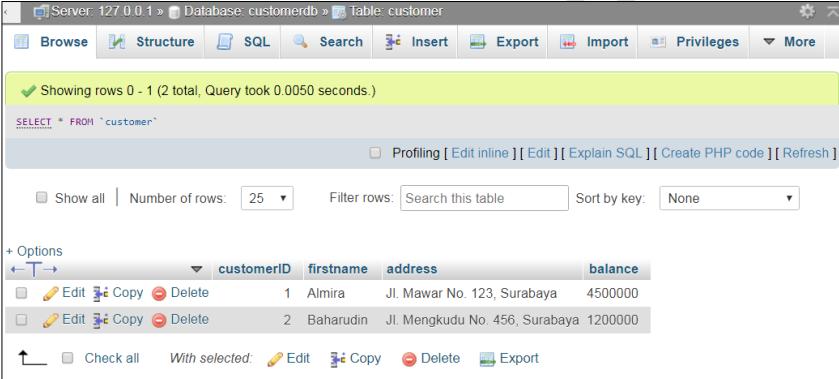
```

1 SELECT firstname, address
2 FROM customer
3 WHERE balance > 0;
4 UPDATE customer
5 SET balance = 4500000, firstname = 'Almira'
6 WHERE customerID = 1;
7 DELETE
8 FROM customer
9 WHERE balance < 0;
10 INSERT INTO customer (firstname, address, balance)
11 VALUES ('Baharudin','Jl. Mengkudu No. 456, Surabaya',1200000);
12
13
14

```

Gambar 4.10. Membuat SQL untuk melakukan perintah SELECT, UPDATE, DELETE, dan INSERT pada tabel “Customer” menggunakan PHPMyAdmin

Gambar 4.11. memperlihatkan hasil isi tabel “customer” setelah SQL dalam Gambar 4.10. dieksekusi.

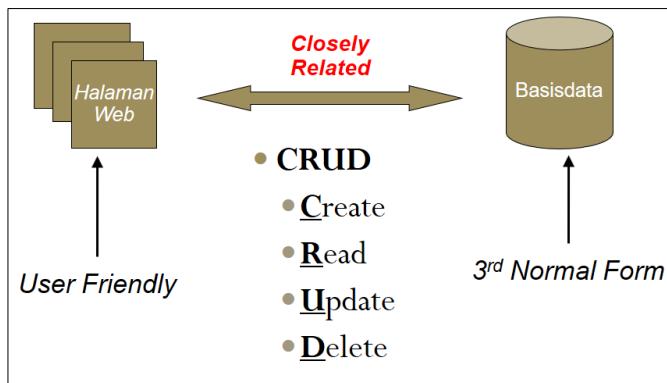


Showing rows 0 - 1 (2 total, Query took 0.0050 seconds.)																
<input type="checkbox"/> Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]																
<input type="checkbox"/> Show all Number of rows: 25 Filter rows: Search this table Sort by key: None																
<input checked="" type="checkbox"/> + Options																
<input type="checkbox"/> Edit Copy Delete																
<table border="1"> <thead> <tr> <th>customerID</th><th>firstname</th><th>address</th><th>balance</th></tr> </thead> <tbody> <tr> <td>1</td><td>Almira</td><td>Jl. Mawar No. 123, Surabaya</td><td>4500000</td></tr> <tr> <td>2</td><td>Baharudin</td><td>Jl. Mengkudu No. 456, Surabaya</td><td>1200000</td></tr> </tbody> </table>					customerID	firstname	address	balance	1	Almira	Jl. Mawar No. 123, Surabaya	4500000	2	Baharudin	Jl. Mengkudu No. 456, Surabaya	1200000
customerID	firstname	address	balance													
1	Almira	Jl. Mawar No. 123, Surabaya	4500000													
2	Baharudin	Jl. Mengkudu No. 456, Surabaya	1200000													
<input type="checkbox"/> Check all With selected: Edit Copy Delete Export																

Gambar 4.11. Hasil isi tabel “customer” setelah SQL dalam Gambar 4.10 dieksekusi

4.2. Aplikasi Web & Implementasi Basisdata

Implementasi basisdata sering kali digunakan dalam pengembangan *web* dinamis. Hubungan erat antara aplikasi *web* dan basisdata dapat digambarkan seperti diagram yang ditunjukkan dalam Gambar 4.12. Dengan menggunakan aplikasi *web* (yang terdiri dari halaman-halaman *web*), kita dapat melakukan operasi **CRUD** (*Create, Read, Update, Delete*) pada basisdata. Dan demikian juga sebaliknya, hasil operasi **CRUD** pada basisdata dapat kita tampilkan pada aplikasi *web*.



Gambar 4.12. Diagram yang menunjukkan hubungan antara aplikasi web dan basisdata

4.2.1. PHP Data Objects (PDO)

PHP *Data Objects* (PDO) merupakan *Object-Oriented extension library*. Fitur ini mulai tersedia pada PHP versi 5.1 dan setelahnya. PDO digunakan sebagai antar muka untuk mengakses sistem basisdata.

Kita dapat mengecek apakah PHP yang kita gunakan telah memiliki fitur PHP atau tidak dengan menggunakan fungsi “`phpinfo()`” yang sebelumnya digunakan dalam pengujian instalasi PHP (cek sub-bab 1.3.2). PHP memiliki fitur PDO jika hasil eksekusi fungsi “`phpinfo()`” dapat memperlihatkan nama fitur PDO seperti yang diperlihatkan dalam Gambar 4.13.

PDO	
PDO support	enabled
PDO drivers	mysql, sqlite
<code>pdo_mysql</code>	
PDO Driver for MySQL	enabled
Client API version	mysqlnd 5.0.12-dev - 20150407 - \$Id: b396954eeb2d1d9ed7902b8bae237b287f21ad9e \$

Gambar 4.13. Pengecekan fitur PDO dalam PHP dengan menggunakan fungsi “`phpinfo()`”

 **Catatan:** Dokumentasi mengenai PDO dalam PHP dapat dilihat pada <http://php.net/manual/en/book pdo.php>.

4.2.2. Koneksi Basisdata dengan Halaman Web

Gambar 4.14. memperlihatkan contoh skrip PHP untuk melakukan koneksi basisdata dengan menggunakan PDO. Koneksi basisdata dapat dilakukan dengan membuat obyek PDO baru dan mengidentifikasi nilai dari parameter-parameter berikut:

- Sistem basisdata yang digunakan, yaitu: "mysql"
- Nama *host* yang merupakan lokasi server, yaitu "localhost"
- Nama basisdata yang digunakan, yaitu: "customerdb"
- *Username* dari akun basisdata yang digunakan, yaitu "root"
- *Password* dari akun basisdata yang digunakan, yaitu tanpa *password*.

```

1. <?php
2. $dbc = new PDO('mysql:host=localhost;dbname=customerdb','root','');
3.
4. // Use the connection ...
5.
6. ?>

```

Sistem Basisdata Nama host Nama basisdata Username basisdata Password

Gambar 4.14. Contoh skrip PHP untuk melakukan koneksi ke basisdata "customerdb" dengan menggunakan PDO

4.2.3. PDO Query

Setelah koneksi basisdata dibuat dengan PDO, maka selanjutnya kita dapat mulai melakukan operasi PDO *query* pada tabel yang diinginkan. Gambar 4.15. memperlihatkan contoh PDO *query* untuk menampilkan data pada kolom "firstname" dan "address" pada tabel "customer", dan lalu menyimpannya dalam variabel \$statement (baris 4). Baris 6-10 digunakan untuk menampilkan hasil dari PDO *query* pada *web browser*.

Skrip PHP:

```
1 <?php  
2     $dbc = new PDO('mysql:host=localhost;dbname=customerdb','root','');
3  
4     $statement = $dbc->query("SELECT firstname, address FROM customer WHERE balance > 0");
5  
6     foreach ($statement as $row)
7     {
8         echo "<h1>{$row['firstname']}</h1>";
9         echo "<p>{$row['address']}</p>";
10    }
11 ?>
```

Hasil Eksekusi:

Almira Jl. Mawar No. 123, Surabaya
Baharudin Jl. Mengkudu No. 456, Surabaya

Gambar 4.15. Contoh PDO query

 **Catatan:** Dokumentasi mengenai PDO *Query* dalam PHP dapat dilihat pada <http://php.net/manual/en/pdo.query.php>.

4.2.4. Serangan SQL *Injection*

Serangan SQL *injection* adalah jenis serangan keamanan yang dapat membuat si penyerang memiliki akses ke dalam basis data. Gambar 4.16. memperlihatkan contoh skrip PHP yang memroses masukan *form* dengan menggunakan HTTP GET dengan menggunakan PDO *query*. Ketika kita memasukkan nilai 2 pada kotak isian/*textbox* “Customer ID” pada *form*, maka kita mendapatkan hasil eksekusi berupa data “firstname” dan “address” dari “Baharudin” (sesuai dengan *query* dan skrip PHP pada baris 4-10). Contoh pada Gambar 4.16 ini rentan terhadap serangan SQL *injection* karena *user* yang tidak tahu menahu mengenai data “Customer ID” dapat dengan mudah memasukkan nilai “true” pada kotak isian/*textbox* *form* yang dapat mengakibatkan seluruh data “firstname” dan “address” pada tabel “customer” ditampilkan pada *web browser*. Ekivalensi *query* SQL yang menyebabkan contoh dalam Gambar 4.16 rentan terhadap serangan SQL *injection* diperlihatkan dalam Gambar 4.17.

Skrip PHP:

```

1 <?php
2     $dbc = new PDO('mysql:host=localhost;dbname=customerdb','root','');
3
4     $statement = $dbc->query("SELECT firstname, address FROM customer WHERE customerID = $_GET['customerID']");
5
6     foreach ($statement as $row)
7     {
8         echo "<h1>{$row['firstname']}</h1>";
9         echo "<p>{$row['address']}</p>";
10    }
11 >

```

Masukan Form:

Customer ID:	<input type="text" value="2"/>	<input type="button" value="Find"/>
--------------	--------------------------------	-------------------------------------

Hasil Eksekusi:

Baharudin

Jl. Mengkudu No. 456, Surabaya

Gambar 4.16. Contoh penerapan PDO *query* yang rentan terhadap serangan SQL *injection*

```
SELECT firstname, address FROM customer WHERE customerID = 2 OR true
```

Gambar 4.17. Ekivalensi *query* SQL yang menyebabkan contoh dalam Gambar 4.16 rentan terhadap serangan SQL *injection*

4.2.5. PDO *Prepared Statement*

Solusi untuk menghindari terjadinya serangan SQL injection adalah dengan menerapkan konsep PDO *prepared statement*. Gambar 4.18 memperlihatkan penerapan PDO *prepared statement*, sebagai pengganti PDO *query* dalam Gambar 4.16, yang digunakan untuk menghindari serangan SQL *injection*.

Skrip PHP:

```

1 <?php
2     $dbc = new PDO('mysql:host=localhost;dbname=customerdb','root','');
3
4     $statement = $dbc->prepare("SELECT * FROM customer WHERE firstname = :firstname");
5     $statement->bindValue(':firstname', $_GET['firstname']);
6     $statement->execute();
7
8     foreach ($statement as $customer)
9     {
10        echo "<p>{$customer['address']}</p>";
11    }
12 >

```

Gambar 4.18. Contoh penerapan PDO *prepared statement* untuk menghindari serangan SQL *injection*

Gambar 4.19, Gambar 4.20, dan Gambar 4.21 secara berurutan memperlihatkan bagaimana PDO *prepared statement* digunakan untuk melakukan operasi penambahan (**INSERT**), perbaharuan (**UPDATE**), dan penghapusan (**DELETE**) data pada tabel “customer”.

Skrip PHP:

```
<?php
$dbc = new PDO('mysql:host=localhost;dbname=customerdb','root','');
$statement = $dbc->prepare("INSERT INTO customer (firstname, address, balance)
VALUES (:firstname, :address, :balance)");
$statement->bindValue(':firstname', $_GET['firstname']);
$statement->bindValue(':address', $_GET['address']);
$statement->bindValue(':balance', 0);
$statement->execute();
?>
```

Masukan Form:

Customer Data	
Firstname:	<input type="text" value="Citra"/>
Address	Jl. Cendana No. 17, Surabaya
	<input type="button" value="Add Data"/> <input type="button" value="Reset"/>

Hasil pada tabel “customer”:

+ Options				
	customerID	firstname	address	balance
<input type="checkbox"/>	1	Almira	Jl. Mawar No. 123, Surabaya	4500000
<input type="checkbox"/>	2	Baharudin	Jl. Mengkudu No. 456, Surabaya	1200000
<input type="checkbox"/>	3	Citra	Jl. Cendana No. 17, Surabaya	0

Gambar 4.19. Contoh penerapan PDO *prepared statement* untuk menambah (INSERT**) data ke dalam tabel “customer”**

Skrip PHP:

```
<?php
$dbc = new PDO('mysql:host=localhost;dbname=customerdb','root','');
$statement = $dbc->prepare("UPDATE customer SET firstname = :firstname, address = :address
WHERE customerID = :customerID");
$statement->bindValue(':firstname', $_GET['firstname']);
$statement->bindValue(':address', $_GET['address']);
$statement->bindValue(':customerID', $_GET['customerID']);
$statement->execute();
?>
```

Gambar 4.20. Contoh penerapan PDO *prepared statement* untuk memperbaharui (UPDATE**) data ke dalam tabel “customer”**

Skrip PHP:

```
<?php  
$dbc = new PDO('mysql:host=localhost;dbname=customerdb','root','');  
  
$statement = $dbc->prepare("DELETE FROM customer WHERE customerID = :customerID");  
$statement->bindValue(':customerID', $_GET['customerID']);  
$statement->execute();  
?>
```

Gambar 4.21. Contoh penerapan PDO *prepared statement* untuk menghapus (DELETE) data ke dalam tabel “customer”

 **Catatan:** Dokumentasi mengenai PDO *prepared statement* dalam PHP dapat dilihat pada <http://php.net/manual/en/pdo.prepared-statements.php>.

4.2.6. PDO Set Attribute

Kita dapat menggunakan fitur PDO *set attribute* untuk menampilkan pesan kesalahan (*error*) pada *web browser* jika ada kesalahan pada skrip SQL yang digunakan dalam skrip PHP. Gambar 4.22. memperlihatkan contoh penambahan fitur PDO *set attribute* sehingga kesalahan sintaks SQL yang dilakukan pada PDO *prepared statement* akan ditampilkan di *web browser*.

Skrip PHP:

```
<?php  
try  
{  
    $dbc = new PDO('mysql:host=localhost;dbname=customerdb','root','');  
    $dbc->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);  
  
    $statement = $dbc->prepare("INSERT INTO customer (fffirstname, aaaddress, bbbalance)  
VALUES (:firstname, :address, :balance)");  
    $statement->bindValue(':firstname', $_GET['firstname']);  
    $statement->bindValue(':address', $_GET['address']);  
    $statement->bindValue(':balance', 0);  
    $statement->execute();  
}  
catch (PDOException $err)  
{  
    echo $err->getMessage();  
}  
?>
```

Hasil Eksekusi:

```
SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in  
your SQL syntax; check the manual that corresponds to your MariaDB server  
version for the right syntax to use near 'INSERT INTO customer (fffirstname,  
aaaddress, bbbalance) VALUES ('Fahra' at line 1
```

Gambar 4.22. Contoh penggunaan PDO *set attribute*

 **Catatan:** Dokumentasi mengenai PDO *set attribute* dalam PHP dapat dilihat pada <http://php.net/manual/en/pdo.setattribute.php>.

 **Catatan:** PDO *Error Mode* secara default adalah *ERRMODE_SILENT* (*silently ignore all database errors*).

4.2.7. Menampilkan Data *Foreign Keys*

Data *Primary Key* atau *Foreign Key* biasanya tidak perlu ditampilkan dalam aplikasi *web*, terkecuali jika memang dianggap sangat penting untuk diketahui oleh *end-user*. Kadang kala operasi **JOIN** perlu dilakukan untuk menampilkan data berdasarkan *Foreign Key*. Gambar 4.23 memperlihatkan contoh tabel “Animals” dan “Breeds” dimana operasi **JOIN** diperlukan untuk menampilkan data berdasarkan *Foreign Key*, yaitu data “BreedName” berdasarkan data “BreedID”. Contoh SQL operasi **JOIN** diperlihatkan dalam Gambar 4.24.

ID	PuppyName	BreedID	Description	Price	Picture	Sold	PuppyOfTheDay
1	1. Johnny	4	Good for a farm	\$100.00	Johnny.jpg		
2	2. Bully	3	A fighter - Excellent watchdog	\$59.99	Bully.jpg		
3	3. Bo-Bo	2	Suit sweet old lady	\$150.00	Bo-Bo.jpg		
4	4. Albert	6	Family dog	\$20.00	Albert.jpg		
5	5. Fritz	1	Watchdog	\$120.00	Fritz.jpg		
6	7. Sam	7	Good for nothing	\$100.00	Sam.jpg		
7	8. Teddy	8	Cuddly	\$150.00	Teddy.jpg		
8	9. Bandit	9	Rat dog	\$99.00	Bandit.jpg	Y	
9	10. Bruiser	8	Good with kids	\$79.00	Bruiser.jpg		Y
10	11. Fred	4	Best driver's companion	\$123.50	NoPicture.gif	Y	
11	12. Max	1	Ishn de Shan	\$33.95			
12	13. Fred	3	A builder's dog	\$99.99			
13	14. WatchDog	2					
14	15. Bob	4					

BreedID	BreedName	Temperament
1	Dobeman	Aggressive
2	Poodle	Nervous
3	Pit Bull	Nasty
4	Cattle Dog	Friendly
5	Alsatian	Faithful
6	Beagle	Smooches
7	Schnauzer	Ruffly
8	Jack Russel	Psychopathic

Gambar 4.23. Contoh tabel “Animals” dan “Breeds” dimana operasi **JOIN** diperlukan untuk menampilkan data berdasarkan *Foreign Key*

Kripik SQL:

```
1. SELECT PuppyName, Breed.BreedName, Picture, Sold
2. FROM Animals, Breeds
3. WHERE Animals.BreedID = Breeds.BreedID;
```

Gambar 4.24. Contoh SQL untuk operasi **JOIN** antara tabel “Animals” dan “Breeds”

4.3. Latihan Soal



Latihan 4.1

Implementasikan contoh penerapan PDO *query* yang rentan terhadap serangan SQL *injection* seperti yang diperlihatkan dalam Gambar 4.16!



Latihan 4.2

Implementasikan contoh penerapan PDO *prepared statement*, sebagai pengganti PDO *query*, seperti yang diperlihatkan dalam Gambar 4.18!



Latihan 4.3

Implementasikan contoh penerapan PDO *prepared statement* untuk menambah (INSERT) data ke dalam tabel “customer” seperti yang diperlihatkan dalam Gambar 4.19!



Latihan 4.4

Implementasikan contoh penerapan PDO *prepared statement* untuk memperbarui (UPDATE) data ke dalam tabel “customer” seperti yang diperlihatkan dalam Gambar 4.20!



Latihan 4.5

Implementasikan contoh penerapan PDO *prepared statement* untuk menghapus (DELETE) data ke dalam tabel “customer” seperti yang diperlihatkan dalam Gambar 4.21!

BAB 5.

KEAMANAN APLIKASI WEB

Bab ini membahas mengenai keamanan aplikasi *web* yang meliputi:

- Konsep prinsip keamanan pada umumnya
- Implementasi keamanan basisdata
- Mekanisme pengamanan *password*
- Bentuk serangan
- Implementasi keamanan yang lemah
- Penggunaan *session*

Setelah selesai mempelajari bab ini maka diharapkan kita dapat mendemonstrasikan implementasi keamanan dalam pengembangan *web* dinamis.

5.1. Prinsip Keamanan

Prinsip keamanan pada umumnya menerapkan konsep **C** (*Confidentiality*) - **I** (*Integrity*) - **A** (*Availability*) - **A** (*Authenticity*) - **N** (*Non-repudiation*), yaitu:

- ***Confidentiality***: mencegah kebocoran data dan memastikan bahwa data tidak dapat diakses oleh pihak yang tidak berwenang
- ***Integrity***: melindungi konsistensi data
- ***Availability***: memastikan bahwa data selalu tersedia ketika dibutuhkan
- ***Authenticity***: memvalidasi identitas *user* sebagai pihak yang ingin mengakses data
- ***Non-repudiation***: mencegah terjadinya penolakan transaksi

Dalam buku ini diberikan contoh penerapan prinsip *Confidentiality* (dengan cara menghindari *SQL injection*) dan

Authenticity (dengan pengamanan *password* dan penggunaan *session*).

5.1.1. Keamanan Basisdata

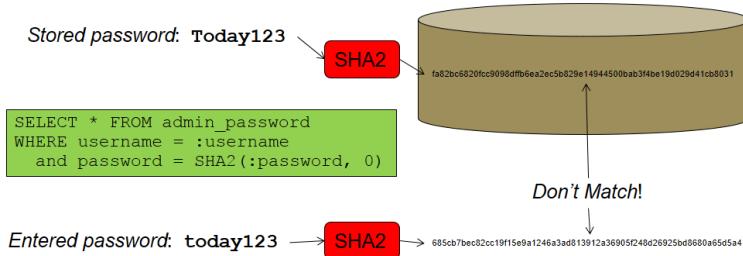
Keamanan basisdata yang dimaksud di sini adalah perlindungan akses terhadap basisdata sehingga hanya *user* yang akunnya terdaftar di sistem basisdata sajalah yang dapat melakukan koneksi dari PHP ke basisdata. Perhatikan kembali daftar akun yang memiliki akses ke dalam sistem basisdata yang diperlihatkan dalam Gambar 4.9. Dan kemudian perhatikan kembali juga Gambar 4.14 yang memperlihatkan bahwa koneksi basisdata dapat dilakukan (dengan membuat obyek PDO baru) jika nilai dari parameter “*Username*” dan “*Password*” sesuai dengan salah satu detil akun ada di dalam Gambar 4.9.

5.1.2. Pengamanan *Password*

Pengamanan *password* untuk keamanan aplikasi *web* dapat dilakukan dengan menerapkan mekanisme berikut:

- Tidak menyimpan data *password* dalam bentuk “*clear text*” di dalam tabel basisdata
- Menggunakan fungsi *hashing* untuk mengenkripsi data.
Contoh: **md5()**, **sha2()**, dan lain-lain.

Gambar 5.1 memperlihatkan ilustrasi pengamanan *password* untuk keamanan aplikasi *web*. Misalkan contoh data “*password*” yang tersimpan di dalam tabel “admin_password” adalah “Today123”. Data dienkripsi dengan menggunakan fungsi **sha2()** sehingga tidak dapat dibaca secara kasat mata. Untuk keamanan aplikasi *web*, enkripsi data “*password*” yang tersimpan di dalam tabel “admin_password” harus dipastikan sama dengan enkripsi data *password* yang dimasukkan lewat *form* (cek skrip SQL).



Gambar 5.1. Ilustrasi pengamanan password untuk keamanan aplikasi web

Berikut adalah langkah-langkah untuk menyimpan data *password* di dalam tabel basisdata dalam bentuk terenkripsi:

1. Buat tabel baru untuk menyimpan data *user* dan *password*. Contoh: tabel “admin”

#	Name	Type
1	username	varchar(64)
2	password	varchar(64)

New
customerdb
New
admin
customer

Gambar 5.2. Tabel “admin” untuk menyimpan data user dan password

2. Tambahkan data baru ke dalam tabel “admin” dengan cara menerapkan fungsi *hashing sha2()* pada isian kolom “password”. Sebagai hasilnya, data *password* tersimpan sebagai data terenkripsi.

Skrip SQL:	
Run SQL query/queries on table customerdb.admin: <input type="button" value="Execute"/>	
1 INSERT INTO admin (username, password) VALUES('Amira', SHA2('Secret!', 0));	
Hasil Eksekusi:	
username	password
Amira	5de772715ff750859d6efa965201bb4ccd059ed04740dd867d...

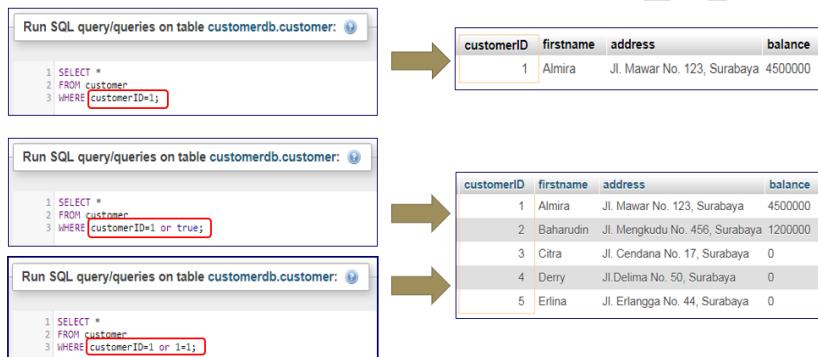
Gambar 5.3. Menambahkan data baru pada tabel “admin”

5.2. Bentuk Serangan dan Cara Menghindarinya

Serangan terhadap keamanan aplikasi *web* dapat berbentuk *SQL injection*, *script injection* ataupun *XML attack*.

5.2.1. SQL Injection

Seperti yang telah dijelaskan di dalam sub-bab 4.2.4, SQL *injection* adalah jenis serangan keamanan *Confidentiality* yang dapat membuat si penyerang memiliki akses ke dalam basis data dan menyebabkan terjadinya kebocoran data. Untuk menghindarinya, maka kita jangan menggunakan pernyataan SQL yang menggabungkan masukan *user* (yang diberikan lewat isian data *form*) secara langsung. Caranya adalah dengan memanfaatkan fitur PDO *Prepared Statements*, yaitu dengan menggunakan kombinasi fungsi **prepare()**, **bindValue()**, dan **execute()**. Gambar 5.4 memperlihatkan manipulasi SQL yang dapat menyebabkan terjadinya SQL *injection*.



Gambar 5.4. Contoh manipulasi SQL yang dapat menyebabkan terjadinya SQL *injection*

5.2.2. Script Injection

Script injection adalah jenis serangan keamanan yang dapat membuat si penyerang mengirimkan *script* yang berbahaya melalui isian *form*. Untuk menghindarinya, maka kita jangan secara langsung menampilkan masukan *user* (yang diberikan lewat isian data *form*) yang berupa skrip. Caranya adalah dengan menggunakan fungsi **htmlspecialchars()**. Gambar 5.5 memberlihatkan contoh serangan *script injection*.



Gambar 5.5. Contoh serangan *script injection*

5.2.3. XML Attack

XML *attack* merupakan *Denial-of-Service* (DoS) attack yang bertujuan untuk mematikan (*shut down*) layanan *web*. Serangan ini sendiri berbentuk XML schema yang *well-formed* dan *valid*. Gambar 5.6 memperlihatkan contoh XML *attack*.

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
  <!ENTITY lol "lol">
  <!ENTITY lol2 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
  <!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
  <!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
  <!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
  <!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
  <!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
  <!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
  <!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
]>
<lolz>&lol9;</lolz>
```

Gambar 5.6. Contoh XML *attack*

5.3. Implementasi keamanan yang lemah

Berikut adalah beberapa contoh implementasi keamanan yang lemah yang harus dihindari:

- Tidak mengimplementasikan sistem keamanan terbaru (pada *web browser*, *web server*, dan lain-lain)

- Tidak mengimplementasikan mekanisme *password* yang baik
- Hanya mengandalkan validasi *client-side*
- Kecerobohan pengembang aplikasi *web*. Misalnya: tidak mengaktifkan fitur pesan kesalahan (*error reporting*), menggunakan variabel tanpa inisialisasi nilai (Gambar 5.7), bergantung pada penggunaan variabel global (Gambar 5.8) yang dilakukan dengan cara mengubah file “*php.ini*” agar parameter “*register_global = On*”.

```
<?php
    $intNoValue ;
    if ($intNoValue == 0 )
        echo "<p>Equals Zero</p>";
?>
```

Gambar 5.7. Contoh kecerobohan penggunaan variabel tanpa inisialisasi

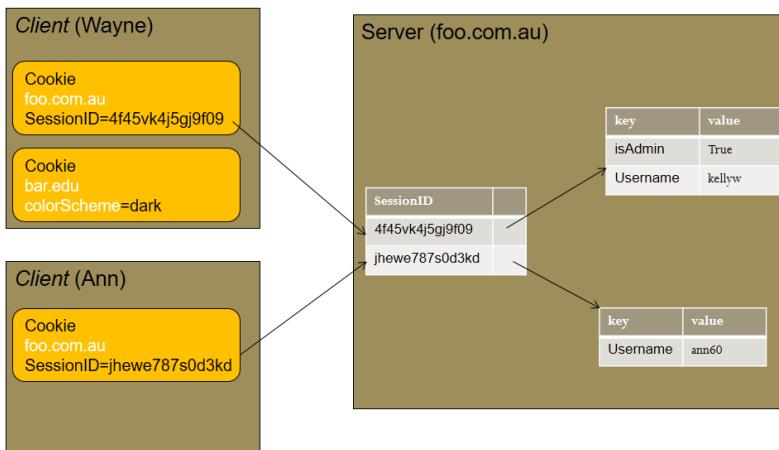
```
<legend>Login</legend>
<div class="field"> <!-- Name field and its error message -->
    <label for="username">Username:</label>
    <input type="text" name="strusername" id="username" />
</div>
<div class="field"> <!-- Password field and its error message -->
    <label for="password">Password:</label>
    <input type="password" name="strpassword" id="password" />
</div>
<div class="field">
    <input class="specialsubmit" type="submit" name="login" value="Login"/>
    <input class="specialsubmit" type="hidden" name="intokay" value="1"/>
</div>
```

```
<?php
if (checkUser ($strusername, $strpassword))
    $intOkay = 1;
if ($intOkay)
    echo "<p>Valid user</p>";
function checkUser ($username, $password) {
    return false;
}
?>
```

Gambar 5.8. Contoh kecerobohan dalam penggunaan variabel global

5.4. Session untuk Authorization

Penggunaan *session* adalah untuk menyimpan informasi mengenai *user* (dalam bentuk variabel) yang diletakkan di suatu *server*. Informasi ini dapat digunakan secara global pada server tersebut dan akan hilang ketika *user* menutup *web browser*. Gambar 5.9 memperlihatkan ilustrasi penggunaan *session*.



Gambar 5.9. Ilustrasi penggunaan *session*

Catatan: Perbedaan antara *session* dan *cookies* adalah bahwa *session* disimpan di *server-side* sedangkan *cookies* disimpan di *client-side* (*web browser*). Oleh sebab itulah *session* lebih aman untuk digunakan jika dibandingkan dengan *cookies*.

Penggunaan *session* untuk prinsip *Authorization* adalah untuk memvalidasi identitas *user* sebagai pihak yang ingin mengakses suatu halaman *web*. Di sini kita akan mencoba membuat *session* dengan menerapkan tiga langkah berikut: membuat kategori halaman *web*, halaman *LOGIN*, dan halaman *LOGOUT*.

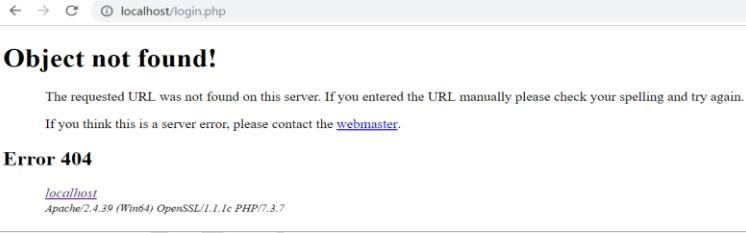
5.4.1. Kategori halaman *web*

Kategorikan halaman-halaman yang ada pada aplikasi *web* menjadi dua kelompok berikut:

- Halaman publik, yaitu halaman yang dapat dilihat/diakses secara langsung oleh *user* tanpa melalui proses *login*
- Halaman non-publik, yaitu halaman yang mengharuskan *user* untuk melakukan proses *login*

Gambar 5.10 memperlihatkan contoh pembuatan halaman non-publik. File "private.php" adalah file untuk halaman publik. Untuk menerapkan prinsip authorization pada halaman ini maka kita perlu membuat *session* dengan menambahkan skrip yang memanfaatkan fungsi **require_once** untuk memanggil file

“adminPermission.inc” (baris 1-3). Pada file “adminPermission.inc”, buatlah session (baris 2) dan pengkondisionan sedemikian hingga jika user akan selalu diarahkan ke halaman *LOGIN* (file “login.php”) untuk *authorization* (baris 3-8). Perhatikan URL pada hasil eksekusi file “private.php”.

Skrip PHP (“private.php”): 1. <?php 2. require 'adminPermission.inc'; 3. ?> 4. 5. <!DOCTYPE html> 6. <html> 7. ... 8. </html>
Skrip PHP (“adminPermission.inc”): 1. <?php 2. session_start(); 3. if (!isset(\$_SESSION['isAdmin'])) // isAdmin adalah contoh label/nama session 4. { 5. // user akan diarahkan ke halaman login untuk authorization 6. header("Location: http://localhost/login.php"); 7. exit(); 8. } 9. ?>
Hasil eksekusi: 

Gambar 5.10. Contoh pembuatan halaman non-publik

5.4.2. Halaman LOGIN

Gambar 5.11 memperlihatkan contoh untuk pembuatan halaman *LOGIN*. Perhatikan bahwa baris 2 mengindikasikan bahwa pada *form* pada halaman *LOGIN* (file “login.php”) harus dikirimkan ke dirinya sendiri (*self-submission*). Gambar 5.12 memperlihatkan contoh *form* *LOGIN* yang skripnya ditambahkan di baris 20 pada Gambar 5.11.

Skrip PHP (“login.php”):

```

1. <?php
2.     if (isset($_POST['login']))
3.     {
4.         include 'checkPassword.inc'; // panggil file include yang berisi fungsi untuk memvalidasi dan memproses isian username dan password yang dikirimkan melalui form
5.         if (checkPassword($_POST['username'], $_POST['password'])) // pengkondisionan berdasarkan fungsi checkPassword
6.             session_start(); // Aktifkan session baru untuk memcatat bahwa seorang user telah berhasil login
7.             $_SESSION['isAdmin'] = true;
8.             header('Location: http://localhost/private.php'); //arahkan user ke halaman non-publik
9.             exit(); //exit dari halaman LOGIN
10.    }
11. ?
12.
13. <!DOCTYPE html>
14. <html>
15.     <head>
16.     . . .
17.     </head>
18.     <body>
19.         <form action="login.php" method="POST">
20.             . . . // kotak isian masukan untuk username dan password
21.         </body>
22.     </html>

```

Skrip Include (“checkPassword.inc”):

```

1. <?php
2.     function checkPassword()
3.     {
4.         $pdo = new PDO('mysql:host=localhost;dbname=customerdb', 'root','');
5.         $pdo ->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
6.         try
7.         {
8.             $query = $pdo->prepare('SELECT * FROM admin WHERE username = :username and password = SHA2(:password, 0)');
9.             $query ->bindValue(':username', $_POST['username']);
10.            $query ->bindValue(':password', $_POST['password']);
11.            $query ->execute();
12.            return $query ->rowCount()>0;
13.        }
14.        catch (PDOException $e)
15.        {
16.            echo $e ->getMessage();
17.        }
18.    }
19.
20. ?

```

Gambar 5.11. Contoh pembuatan halaman LOGIN**Login**

The image shows a screenshot of a web browser displaying a login form. The title bar of the browser window says "Login". Below the title, there is a light blue rectangular area containing two text input fields. The first field is labeled "Username" and the second is labeled "Password", both in small black text. Below these fields is a single blue rectangular button labeled "Login" in white text.

Gambar 5.12. Contoh form LOGIN

Berikut adalah langkah-langkah untuk memvalidasi & memrosesan isian *username* dan *password* yang dikirimkan melalui *form LOGIN*:

- a. Buatlah fungsi (*file* “checkPassword.inc”) untuk mengecek apakah *username* dan *password* yang dimasukkan adalah benar. *File* ini kemudian dipanggil di dalam *file* “login.php” dengan menggunakan fungsi include (baris 4). Gunakan fungsi PDO *Prepared Statement* untuk mengeksekusi skrip SQL yang digunakan untuk memvalidasi data masukan *form* dengan yang data yang ada di dalam tabel basisdata seperti yang dibuat dalam Gambar 5.2 (baris 4-11). Ketika skrip SQL menghasilkan keluaran setidaknya satu baris, maka hasil validasi bernilai *true* (baris 12)
- b. Tambahkan struktur kondisi berdasarkan keluaran fungsi *checkPassword* (*file* “login.php” baris 5). Jika pengecekan *username* dan *password* yang dimasukkan adalah benar, maka aktifkan session baru untuk memcatat bahwa seorang *user* “admin” telah berhasil *login* (baris 6-7). Selanjutnya arahkan *user* “admin” ke halaman non-publik (baris 8) dan keluar dari halaman *LOGIN* (baris 9).

5.4.3. Halaman LOGOUT

Gambar 5.13 memperlihatkan contoh untuk pembuatan halaman *LOGOUT*. Dalam halaman *LOGOUT* (*file* “logout.php”), tambahkan skrip untuk membersihkan data *user* dari *session* (baris 1-4): Kemudian tampilkan informasi bahwa *user* telah berhasil *logout* (tambahkan skripnya pada baris 12).

Skrup PHP (“logout.php”):

```
1. <?php
2.     session_start();
3.     unset($_SESSION['isAdmin']);
4. ?
5.
6. <!DOCTYPE html>
7. <html>
8.   <head>
9.     .
10.    </head>
11.    <body>
12.      // tampilkan informasi bahwa user telah berhasil logout
13.    </body>
14. </html>
```

Gambar 5.13. Contoh pembuatan halaman *LOGOUT*

5.5. Latihan Soal



Latihan 5.1

Implementasikan contoh penyimpanan data *password* di dalam tabel basisdata dalam bentuk terenkripsi seperti yang diperlihatkan dalam Gambar 5.2 dan Gambar 5.3!



Latihan 5.2

Implementasikan contoh pembuatan *session* untuk authorization seperti yang diperlihatkan dalam sub-bab 5.4!

MNC Publishing

BAB 6. NODE.JS

Bab ini membahas Node.js sebagai bahasa pemrograman yang juga dapat digunakan untuk pengembangan *web* dinamis. Selain mempelajari dasar PHP, di sini kita juga akan mempelajari perintah-perintah Node.js berikut:

- *Module* dalam Node.js
- Node.js sebagai *web server*
- HTTP *header* pada Node.js
- Node.js sebagai *file server*
- NPM (Node.js *Package Manager*)
- Node.js dan Sistem Basisdata

Setelah selesai mempelajari bab ini maka diharapkan kita dapat mendemonstrasikan aplikasi perintah-perintah Node.js tersebut di atas untuk pengembangan *web* dinamis.

6.1. Dasar Node.js

Node.js merupakan *Open-source framework* dengan MIT license (<https://nodejs.org>) yang menggunakan JavaScript untuk membangun aplikasi *server-side*. Node.js menggunakan *single-threaded model* dan bekerja secara *asynchronous* yang menjadikannya dapat bekerja lebih cepat daripada *framework* lain. Sama halnya dengan PHP, Node.js dapat secara bebas digunakan pada berbagai sistem operasi seperti Windows, Linux, Mac dan Unix.

Node.js memiliki *virtual environment* atau *Node shell* yang disebut sebagai REPL (*Read-Eval-Print-Loop*). Konsole digunakan untuk membuat dan menguji skrip Node.js/JavaScript. Cek sub-bab 1.3.3 untuk melihat contoh penggunaan konsole.

6.2. *Module dalam Node.js*

Module dalam Node.js adalah ekivalen dengan libraries dalam JavaScript. Module merupakan sekumpulan fungsi (*function*) yang dapat digunakan dalam aplikasi web. Seperti halnya fungsi pada umumnya, module dapat berbentuk *built-in* ataupun *user-defined*. Cara menggunakan *module* adalah dengan menggunakan perintah “`require(' nama_module')`”.

 **Catatan:** Daftar *module built-in* dalam Node.js dapat dilihat pada https://www.w3schools.com/nodejs/ref_modules.asp.

6.3. *Node.js sebagai Web Server*

Gambar 6.1 memperlihatkan contoh cara menjadikan Node.js sebagai *web server* (nama file “`http_server.js`”). Baris 2 memperlihatkan bahwa kita menggunakan *module http* agar Node.js dapat melakukan transfer data dengan menggunakan *Hyper Text Transfer Protocol* (HTTP). Kemudian kita menggunakan fungsi `createServer()` untuk membuat HTTP server (baris 5), menentukan *response* apa yang ditampilkan pada *web browser* (baris 6), dan port untuk server (baris 9). Untuk mengaktifkan *web server*, eksekusi program melalui *command prompt* dan kemudian buka port 3000 melalui *web browser*.

Skrip Node.js (“ <code>http_server.js</code> ”):
<pre> 1 //include HTTP module 2 var http = require('http'); 3 4 //create a server object: 5 var server = http.createServer(function(req, res) { 6 res.write('Hello, World!'); //write a response to the client 7 res.end(); //end the response 8 }); 9 server.listen(3000); //the server object listens on port 3000 </pre>
Skrip command prompt:
<pre>C:\@ifa\PAW>node http_server</pre>
Hasil Eksekusi:


Gambar 6.1. Contoh menjadikan Node.js sebagai *web server*

6.4. HTTP Header pada Node.js

Gambar 6.2 memperlihatkan contoh cara menjadikan Node.js sebagai *web server* (nama file "http_header.js"). Baris 6 memperlihatkan bahwa kita menambahkan HTTP Header untuk dapat menampilkan *response* dari *web server* sesuai dengan tipe konten yang diinginkan. Untuk mengaktifkan *web server*, eksekusi program melalui *command prompt* dan kemudian buka port 3000 melalui *web browser*. Perhatikan perbedaan jenis *font* antara hasil eksekusi pada Gambar 6.1 dan Gambar 6.2.

Skrip Node.js ("http_header.js"):

```

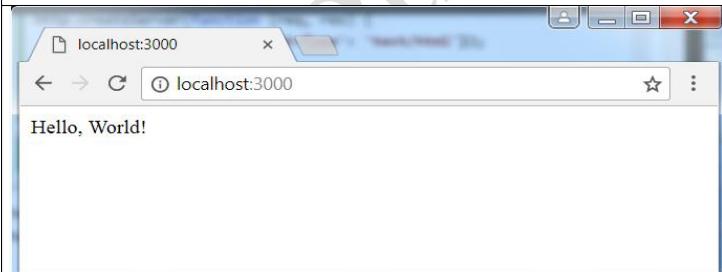
1 //include HTTP module
2 var http = require('http');
3
4 //create a server object:
5 var server = http.createServer(function(req, res) {
6   res.writeHead(200, {'Content-Type': 'text/html'}); //add an
7   //HTTP header to display response as HTML
8   res.write('Hello, World!'); //write a response to the client
9   res.end(); //end the response
10 });
11 server.listen(3000); //the server object listens on port 3000

```

Skrip command prompt:

```
C:\@ifa\PAW>node http_header
```

Hasil Eksekusi:



Gambar 6.2. Contoh menambahkan HTTP Header pada Node.js

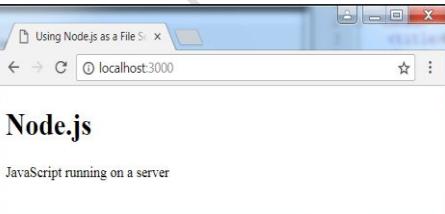
6.5. Node.js sebagai File Server

Kita dapat menggunakan *module fs* agar Node.js dapat berkerja dengan sistem *file (file system)* yang ada di komputer dan melakukan perinta-perintah berikut:

- *Read files*
- *Create files*
- *Update files*
- *Delete files*

- *Rename files*

Gambar 6.3 memperlihatkan contoh cara menjadikan Node.js sebagai *file server* (nama file "fs_read.js"). Baris 9 menunjukan bahwa kita akan membaca file "index.html".

Skrip HTML ("index.html"):
1 <!DOCTYPE html> 2 <head> 3 <title>Using Node.js as a File Server</title> 4 </head> 5 <body> 6 <h1>Node.js</h1> 7 <p>JavaScript running on a server</p> 8 </body> 9 </html>
Skrip Node.js ("fs_read.js"):
1 //include HTTP module 2 var http = require('http'); 3 4 //include FS module 5 var fs = require('fs'); 6 7 //create a server object: 8 var server = http.createServer(function (req, res) { 9 fs.readFile('./index.html', function(err, data) 10 //read file on computer 11 { 12 res.writeHead(200, {'Content-Type': 'text/html'}); 13 //add an HTTP header to display response as HTML 14 res.write(data); //write a response to the client 15 res.end(); //end the response 16 }); 17 server.listen(3000); //the server object listens on port 18 3000
Skrip command prompt:
C:\Qifa\PAW>node fs_read
Hasil Eksekusi:


Gambar 6.3 Contoh menjadikan Node.js sebagai *file server*

6.6. NPM (*Node.js Package Manager*)

NPM (*Node.js Package Manager*) digunakan untuk melakukan instalasi *module node*. Program ini secara otomatis terinstal ketika

Node.js terinstal. Cara mengunduh package adalah dengan mengetikkan perintah berikut “**npm install nama_package**”.



Catatan: Daftar *module* yang terinstal dalam Node.js tersimpan di dalam folder “node_modules”.

6.7. Node.js dan Sistem Basisdata

Node.js dapat digunakan untuk mengakses sistem basisdata. Langkah pertama yang harus dilakukan adalah dengan mengunduh dan menginstalasi *module mysql* lewat NPM agar basisdata MySQL dapat diakses dengan menggunakan Node.js (Gambar 6.4).

```
C:\@ifa\PAW>npm install mysql
```

Gambar 6.4. Skrip command prompt untuk instalasi *module mysql*

6.7.1. Membuat Koneksi ke Basisdata

Untuk melakukan koneksi, gunakan nama *host*, *username* dan *password* yang digunakan untuk mengakses basisdata. Di sini pastikan bahwa *server* basisdata sudah diaktifkan (lewat XAMPP Control Panel). Perhatikan Gambar 6.5 tentang *file* “db_connection.js”.

Skrip Node.js (“db_connection.js”):

```

1 var mysql = require('mysql');
2
3 var connection = mysql.createConnection({
4   host: "localhost",           ← nama host
5   user: "root",                ← username basisdata
6   password: ""                 ← password basisdata
7 });
8
9 connection.connect(function(err){
10   if (err) throw err;
11   console.log("Database is connected!");
12 });

```

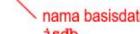
Skrip command prompt:

```
C:\@ifa\PAW>node db_connection
Database is connected!
```

Gambar 6.5. Contoh membuat koneksi ke basisdata dengan menggunakan Node.js

6.7.2. Query: Membuat (Create) Basisdata

Gunakan perintah **CREATE DATABASE** untuk membuat basisdata baru dengan menggunakan Node.js. Perhatikan Gambar 6.6 tentang file “db_create.js”.

Skrip Node.js (“db_create.js”):	<pre> 1 var mysql = require('mysql'); 2 3 var connection = mysql.createConnection({ 4 host: "localhost", 5 user: "root", 6 password: "" 7 }); 8 9 connection.connect(function(err) { 10 if (err) throw err; 11 console.log("Database is connected!"); 12 connection.query("CREATE DATABASE jsdb", function (err, result) { 13 if (err) throw err; 14 console.log("jsdb database is created"); 15 }); 16 }); </pre>
	 nama basisdata: jsdb
Skrip command prompt:	C:\@ifa\PAW>node db_create Database is connected! jsdb database is created

Gambar 6.6. Contoh membuat basisdata baru dengan menggunakan Node.js

6.7.3. Query: Membuat (Create) Tabel

Gunakan perintah **CREATE TABLE** untuk membuat tabel baru. Perhatikan Gambar 6.7 tentang file “db_create_table.js”.

Skrip Node.js (“db_create_table.js”):	<pre> 1 var mysql = require('mysql'); 2 3 var connection = mysql.createConnection({ 4 host: "localhost", 5 user: "root", 6 password: "", 7 database: "jsdb" ← basisdata yang digunakan 8 }); 9 10 connection.connect(function(err) { 11 if (err) throw err; 12 console.log("Database is connected!"); 13 var sql = "CREATE TABLE customer (customerID INT(6) NOT NULL 14 AUTO_INCREMENT, firstName VARCHAR(45) NOT NULL, address VARCHAR(256) 15 NOT NULL, phone VARCHAR(20), email VARCHAR(128) PRIMARY KEY(customerID))"; 16 connection.query(sql, function (err, result) { 17 if (err) throw err; 18 console.log("customer table is created"); 19 }); 20 }); </pre>
Skrip command prompt:	C:\@ifa\PAW>node db_create_table Database is connected! customer table is created
Hasil Eksekusi:	

Gambar 6.7. Contoh membuat tabel baru dengan menggunakan Node.js

6.7.4. Query: Tambah Data (Insert) ke Tabel

Gunakan perintah **INSERT INTO** untuk menambahkan data pada tabel. Perhatikan Gambar 6.8 tentang file “db_insert.js”.

Skrip Node.js (“db_insert.js”):	<pre>1 var mysql = require('mysql'); 2 3 var connection = mysql.createConnection({ 4 host: "localhost", 5 user: "root", 6 password: "", 7 database: "jsdb" 8 }); 9 10 connection.connect(function(err) { 11 if (err) throw err; 12 console.log("Database is connected!"); 13 var sql = "INSERT INTO customer (firstname,address,balance) VALUES 14 ('Amira', 'Jl. Mawar No. 123, Surabaya',1000000)"; 15 connection.query(sql, function (err, result) { 16 if (err) throw err; 17 console.log(result.affectedRows + " record inserted into customer 18 table"); 19 }); 20});</pre>								
Skrip command prompt:	C:\@ifa\PAW>node db_insert Database is connected! 1 record inserted into customer table								
Hasil Eksekusi:	<table border="1"> <thead> <tr> <th>customerID</th> <th>firstname</th> <th>address</th> <th>balance</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Amira</td> <td>Jl. Mawar No. 123, Surabaya</td> <td>1000000</td> </tr> </tbody> </table>	customerID	firstname	address	balance	1	Amira	Jl. Mawar No. 123, Surabaya	1000000
customerID	firstname	address	balance						
1	Amira	Jl. Mawar No. 123, Surabaya	1000000						

Gambar 6.8. Contoh menambahkan data baru (**INSERT**) ke tabel dengan menggunakan Node.js

6.7.5. Query: Pilih Data (Select) dari Tabel

Gunakan perintah **SELECT** untuk memilih data pada tabel. Perhatikan Gambar 6.9 tentang file “db_select.js”.

Skrip Node.js (“db_select.js”):	<pre>1 var mysql = require('mysql'); 2 3 var connection = mysql.createConnection({ 4 host: "localhost", 5 user: "root", 6 password: "", 7 database: "jsdb" 8 }); 9 10 connection.connect(function(err) { 11 if (err) throw err; 12 var sql = "SELECT * FROM customer"; 13 connection.query(sql, function (err, result) { 14 if (err) throw err; 15 console.log(result); 16 }); 17});</pre>
Skrip command prompt:	C:\@ifa\PAW>node db_select RowDataPacket { customerID: 1, firstname: 'Amira', address: 'Jl. Mawar No. 123, Surabaya', balance: 1000000 }

Gambar 6.9. Contoh memilih data (**SELECT**) pada tabel dengan menggunakan Node.js

6.7.6. Query: Perbaharui Data (Update) Tabel

Gunakan perintah **UPDATE** untuk memperbaharui data tabel. Perhatikan Gambar 6.10 tentang file “db_update.js”.

Skrip Node.js (“db_update.js”):								
<pre> 1 var mysql = require('mysql'); 2 3 var connection = mysql.createConnection({ 4 host: "localhost", 5 user: "root", 6 password: "", 7 database: "jsdb" 8 }); 9 10 connection.connect(function(err) { 11 if (err) throw err; 12 console.log("Database is connected!"); 13 var sql = "UPDATE customer SET balance = 4500000 WHERE customerID = 1"; 14 connection.query(sql, function (err, result) { 15 if (err) throw err; 16 console.log(result.affectedRows + " record updated from customer 17 table"); 18 }); 19 }); </pre>								
Skrip command prompt:								
<pre>C:\Bifa\PAW>node db_update Database is connected! 1 record updated from customer table</pre>								
Hasil Eksekusi:								
<table border="1"> <thead> <tr> <th>customerID</th> <th>firstname</th> <th>address</th> <th>balance</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Amira</td> <td>Jl. Mawar No. 123, Surabaya</td> <td>4500000.00</td> </tr> </tbody> </table>	customerID	firstname	address	balance	1	Amira	Jl. Mawar No. 123, Surabaya	4500000.00
customerID	firstname	address	balance					
1	Amira	Jl. Mawar No. 123, Surabaya	4500000.00					

Gambar 6.10. Contoh memperbaharui data (**UPDATE**) pada tabel dengan menggunakan Node.js

6.7.7. Query: Hapus Data (Delete) Tabel

Gunakan perintah **DELETE** untuk menghapus data tabel. Perhatikan Gambar 6.11 tentang file “db_delete.js”.

Skrip Node.js (“db_delete.js”):
<pre> 1 var mysql = require('mysql'); 2 3 var connection = mysql.createConnection({ 4 host: "localhost", 5 user: "root", 6 password: "", 7 database: "jadb" 8 }); 9 10 connection.connect(function(err) { 11 if (err) throw err; 12 console.log("Database is connected!"); 13 var sql = "DELETE FROM customer WHERE balance < 0"; 14 connection.query(sql, function (err, result) { 15 if (err) throw err; 16 console.log(result.affectedRows + " record deleted from customer 17 table"); 18 }); 19 }); </pre>
Skrip command prompt:
<pre>C:\Bifa\PAW>node db_delete Database is connected! 0 record deleted from customer table</pre>

Gambar 6.11. Contoh menghapus data (**DELETE**) pada tabel dengan menggunakan Node.js

6.8. Latihan Soal



Latihan 6.1

Implementasikan contoh menjadikan Node.js sebagai *web server* seperti yang diperlihatkan dalam Gambar 6.1!



Latihan 6.2

Implementasikan contoh menambahkan HTTP *Header* pada Node.js seperti yang diperlihatkan dalam Gambar 6.2!



Latihan 6.3

Implementasikan contoh menjadikan Node.js sebagai *file server* seperti yang diperlihatkan dalam Gambar 6.3!



Latihan 6.4

Implementasikan contoh-contoh penggunaan Node.js untuk mengakses sistem basisdata seperti yang diperlihatkan dalam subbab 6.7!

MNC Publishing

INDEKS

A

Array, v, 24, 25, 26, 27, 37, 38
atribut, 42, 47, 50, 53

B

basisdata, vii, xi, xii, xiii, 2, 3,
17, 23, 55, 56, 57, 61, 62, 63,
64, 71, 72, 73, 80, 85, 86, 89,
94
built-in, 37, 52, 82

C

client-side, 2, 3, 50, 76
Create, vii, viii, 56, 57, 62, 83, 86

D

Delete, viii, 62, 84, 88
dinamis, iii, 2, 3, 4, 9, 11, 17, 45,
55, 62, 71, 81

E

Echo, 21

F

FOR, 33, 34, 35, 44
FOREACH, 33, 35, 44, 49
form, xi, xiii, 2, 45, 46, 47, 48,
49, 50, 51, 52, 53, 65, 72, 74,
78, 79
fungsi, vi, x, xi, xii, xiv, 17, 20,
36, 37, 38, 39, 45, 51, 52, 63,
72, 73, 74, 77, 82

G

global, x, 17, 38, 39, 76

H

html, xi, 4, 18, 47, 49, 84
HTTP, viii, xiii, 1, 4, 45, 47, 48,
50, 65, 81, 82, 83, 89

I

IF, 31, 32
include, 17, 40
Indeks, vi, 25, 26, 27
Insert, vii, viii, 59, 60, 87
instalasi, ix, x, xiii, 4, 5, 6, 7, 8,
9, 10, 11, 12, 13, 14, 15, 16, 55,
63, 84, 85

J

JavaScript, iii, x, 2, 15, 50, 81, 82

K

keamanan, vii, xii, 65, 71, 72,
73, 74, 75
Konkatenasi, v, 21, 31
Konsole, 81

L

lokal, 17, 38, 39

M

MariaDB, 4, 11, 55, 56
Module, viii, 81, 82
MySQL, vii, ix, 4, 5, 9, 11, 17,
55, 56, 85

N

Node.js, iii, viii, ix, x, xiii, 2, 3,
10, 11, 12, 13, 14, 15, 16, 81,

82, 83, 84, 85, 86, 87, 88, 89,
93

O

obyek, 17, 42, 64, 72
Operator, vi, xiv, 27, 28, 29, 30,
31

P

parameter, 36, 38, 39, 64, 72, 76
pass-by-reference, x, 17, 39, 40
pass-by-value, x, 17, 39, 40
Password, 48, 64, 72
PDO, vii, xii, 63, 64, 65, 66, 67,
68, 69, 70, 72, 74
PHP, iii, vii, x, xii, 2, 3, 4, 6, 9,
10, 17, 18, 19, 20, 22, 23, 24,
25, 26, 27, 28, 29, 30, 31, 33,
36, 37, 38, 40, 41, 42, 43, 44,
45, 47, 48, 50, 51, 52, 63, 64,
65, 68, 72, 81

S

script injection, 73, 74, 75
server-side, xi, 2, 3, 17, 45, 50,
51, 81
session, 71, 72, 76, 77, 80

SQL, vii, xi, xii, xiii, 3, 17, 58,
61, 62, 65, 66, 68, 69, 71, 72,
73, 74
SQL injection, 65, 66, 69, 71, 73,
74
statis, 2
SWITCH, 31, 32, 33

T

tanda petik, x, xiv, 22, 23
tipe data, 17, 24, 26, 29, 35, 49

U

Update, viii, 62, 83, 88
user-defined, 82

V

Validasi, vi, 45, 50
variabel, x, xiii, xiv, 17, 24, 25,
26, 27, 28, 29, 30, 32, 35, 36,
38, 39, 64, 76

W

WHILE, 33, 34, 35, 44

X

XAMPP, ix, 4, 5, 6, 7, 8, 9, 16,
55, 85
XML attack, 73, 75

DAFTAR PUSTAKA

- Johanan, J, T Khan, dan R Zea. 2016. *Web Developer's Reference Guide*. Packt Publishing.
- Krause, Jörg. 2016. *Introducing Web Development*. Berlin: Apress.
- w3schools.com. -. *Node.js Tutorial*. Diakses August 12, 2019.
<https://www.w3schools.com/nodejs/>.
- Wellens, P. 2015. *Practical Web Development*. Packt Publishing.

BIOGRAFI PENULIS



Dr. Noor Ifada memperoleh gelar Sarjana bidang Teknik Elektro dari Institut Teknologi Sepuluh Nopember (ITS) Indonesia pada tahun 2000, gelar Master bidang *Information Systems Development* dari HAN University The Netherlands pada tahun 2007, dan gelar Doktor bidang *Electrical Engineering dan Computer Science* dari Queensland University of Technology (QUT) Australia pada tahun 2016.

Kualifikasi:

- Pengalaman mengajar: dosen di Universitas Trunojoyo Madura (UTM) Indonesia sejak tahun 2003 dan *Sessional Academic* di QUT dari tahun 2013 hingga 2016
- Topik penelitian: *Web Technology* dan Sistem Rekomendasi
- Mata kuliah ajar: mata kuliah yang berkaitan dengan algoritma pemrograman, basisdata, pemrograman *web*, dan sistem rekomendasi
- Sertifikasi: Kementerian Pendidikan Nasional Republik Indonesia (2011) dan *Associate Fellow* dari *The Higher Education Academy* dengan *UK Professional Standards Framework* (2016)
- Pengalaman menulis: Buku ajar Dasar Pemrograman *Web*, Diktat mata kuliah Algoritma Pemrograman, Modul ajar mata kuliah Basisdata serta Pemrograman Basisdata berbasis *Web*.

Kontak: ke noor.ifada@trunojoyo.ac.id.