



Bahasa Pemrograman PHP / MySQL

Modul ajar ini dirancang untuk memberikan pemahaman komprehensif tentang pengembangan web menggunakan PHP dan MySQL, mulai dari dasar hingga konsep lanjutan seperti Object-Oriented Programming (OOP), Model-View-Controller (MVC), dan penggunaan framework CodeIgniter 4. Modul ini terdiri dari 10 sesi praktikum yang berfokus pada penerapan langsung konsep-konsep yang diajarkan.

Struktur Setiap Sesi Praktikum

Setiap sesi praktikum akan memiliki struktur sebagai berikut:

1. Tujuan Pembelajaran

Bagian ini menjelaskan apa yang diharapkan dapat dicapai oleh mahasiswa setelah menyelesaikan sesi praktikum.

2. Materi Pembelajaran

Bagian ini berisi penjelasan detail mengenai konsep-konsep yang akan dipelajari, termasuk teori, sintaks, dan contoh penggunaan.

3. Pembahasan

Bagian ini akan menguraikan langkah-langkah praktis dan implementasi dari materi yang telah dijelaskan, seringkali disertai dengan studi kasus atau skenario.

4. Contoh Kode

Bagian ini menyediakan potongan kode yang relevan untuk setiap konsep yang dibahas, yang dapat langsung dicoba oleh mahasiswa.

5. Langkah-langkah Praktikum

Bagian ini berisi instruksi langkah demi langkah untuk menyelesaikan tugas praktikum, memastikan mahasiswa dapat mengikuti dengan mudah dan menerapkan pengetahuan mereka.

Daftar Sesi Praktikum

Berikut adalah daftar 10 sesi praktikum yang akan dibahas dalam modul ini:

1. Praktikum 1: Pengenalan PHP dan Sintaks Dasar

- Tujuan: Memahami dasar-dasar PHP, variabel, tipe data, dan operator.
- Materi: Sejarah PHP, instalasi XAMPP/LAMPP, sintaks dasar PHP, komentar, variabel, tipe data (string, integer, float, boolean, array, object, NULL, resource), operator (aritmatika, penugasan, perbandingan, logika, inkrement/dekremen).

2. Praktikum 2: Struktur Kontrol dan Fungsi PHP

- Tujuan: Menguasai penggunaan struktur kontrol (kondisional dan perulangan) serta membuat dan menggunakan fungsi di PHP.
- Materi: Struktur kondisional (if-else, elseif, switch), struktur perulangan (for, while, do-while, foreach), fungsi (deklarasi, pemanggilan, parameter, nilai kembalian), fungsi bawaan PHP.

3. Praktikum 3: Pengenalan MySQL dan Operasi Dasar Database

- Tujuan: Memahami konsep database relasional dan melakukan operasi dasar pada MySQL menggunakan phpMyAdmin.
- Materi: Konsep database relasional, tabel, kolom, baris, tipe data MySQL, DDL (CREATE DATABASE, CREATE TABLE, ALTER TABLE, DROP TABLE), DML (INSERT, SELECT, UPDATE, DELETE).

4. Praktikum 4: Form Handling dan Validasi

- Tujuan: Mengembangkan aplikasi web yang dapat menerima input dari pengguna melalui form HTML dan melakukan validasi data.
- Materi: Metode GET dan POST, superglobal `$_GET` dan `$_POST`, validasi data sisi server (empty, isset, is_numeric, strlen, filter_var), menampilkan pesan error.

5. Praktikum 5: Koneksi PHP ke MySQL dan CRUD

- Tujuan: Membuat koneksi antara PHP dan MySQL, serta mengimplementasikan operasi CRUD (Create, Read, Update, Delete) pada data database.
- Materi: Ekstensi MySQLi dan PDO, koneksi database, query SQL (INSERT, SELECT, UPDATE, DELETE) melalui PHP, prepared statements untuk keamanan.

6. Praktikum 6: Session dan Cookie Management

- Tujuan: Mengelola status pengguna antar halaman menggunakan session dan cookie.
- Materi: Konsep session dan cookie, fungsi `session_start()` , `$_SESSION` , `session_destroy()` , `setcookie()` , `$_COOKIE` , penggunaan session dan cookie untuk otentifikasi sederhana.

7. Praktikum 7: Konsep Dasar OOP di PHP

- Tujuan: Memahami dan menerapkan konsep dasar Object-Oriented Programming (OOP) di PHP.
- Materi: Kelas, objek, properti, metode, konstruktor, destruktur, enkapsulasi (public, private, protected), pewarisan (inheritance), polimorfisme, abstraksi, interface.

8. Praktikum 8: Pengenalan Konsep MVC

- Tujuan: Memahami arsitektur Model-View-Controller (MVC) dan menerapkannya dalam proyek PHP sederhana.
- Materi: Konsep MVC, peran Model, View, dan Controller, alur kerja MVC, implementasi MVC sederhana tanpa framework.

9. Praktikum 9: Instalasi dan Konfigurasi CodeIgniter 4

- Tujuan: Menginstal dan mengkonfigurasi framework CodeIgniter 4, serta memahami struktur dasar proyeknya.
- Materi: Persyaratan sistem, instalasi CodeIgniter 4 via Composer, struktur direktori, konfigurasi dasar (baseURL, database), routing dasar.

10. Praktikum 10: CRUD dengan CodeIgniter 4

- Tujuan: Mengimplementasikan operasi CRUD menggunakan framework CodeIgniter 4.
 - Materi: Membuat Controller, Model, dan View di CodeIgniter 4, penggunaan Query Builder, form handling dan validasi di CI4, menampilkan data, menambah, mengubah, dan menghapus data.
-

Praktikum 1: Pengenalan PHP dan Sintaks Dasar

1. Tujuan Pembelajaran

Setelah menyelesaikan praktikum ini, mahasiswa diharapkan mampu:

- Memahami sejarah singkat dan kegunaan PHP dalam pengembangan web.
- Melakukan instalasi dan konfigurasi lingkungan pengembangan PHP (XAMPP/LAMPP).
- Mengenali dan menggunakan sintaks dasar PHP.
- Memahami dan mengaplikasikan konsep variabel dan tipe data di PHP.
- Menggunakan berbagai jenis operator di PHP.

2. Materi Pembelajaran

2.1. Pengantar PHP

PHP (Hypertext Preprocessor) adalah bahasa skrip sisi server yang dirancang khusus untuk pengembangan web. PHP dapat disematkan langsung ke dalam HTML. Sintaksnya mirip dengan C, Java, dan Perl, membuatnya mudah dipelajari. PHP adalah bahasa open-source dan sangat populer untuk membangun situs web dinamis dan aplikasi web. [1]

2.2. Instalasi Lingkungan Pengembangan (XAMPP/LAMPP)

Untuk menjalankan kode PHP, kita memerlukan server web (seperti Apache), database (seperti MySQL), dan interpreter PHP. XAMPP (untuk Windows, macOS, Linux) atau LAMPP (untuk Linux) adalah paket perangkat lunak gratis yang menyediakan semua komponen ini dalam satu instalasi yang mudah. [2]

Langkah-langkah Instalasi XAMPP (Windows):

1. Unduh XAMPP dari situs web resmi Apache Friends (apachefriends.org).
2. Jalankan installer dan ikuti petunjuk. Pastikan untuk memilih komponen Apache, MySQL, dan PHP.
3. Setelah instalasi selesai, buka XAMPP Control Panel.
4. Mulai modul Apache dan MySQL dengan mengklik tombol 'Start' di sampingnya.

2.3. Sintaks Dasar PHP

Kode PHP dieksekusi di sisi server dan hasilnya dikirimkan ke browser sebagai HTML biasa. Kode PHP harus diapit oleh tag pembuka `<?php` dan tag penutup `?>`. Setiap pernyataan di PHP harus diakhiri dengan titik koma `(;)`.

Contoh:

```
<?php  
echo "Hello, World!"; // Menampilkan teks ke browser  
?>
```

2.4. Komentar

Komentar digunakan untuk menjelaskan kode dan tidak akan dieksekusi oleh interpreter PHP. PHP mendukung komentar satu baris dan komentar multi-baris.

- **Komentar satu baris:** Dimulai dengan `//` atau `#`.
- **Komentar multi-baris:** Dimulai dengan `/*` dan diakhiri dengan `*/`.

Contoh:

```
<?php  
// Ini adalah komentar satu baris  
# Ini juga komentar satu baris  
  
/*  
Ini adalah  
komentar multi-baris  
yang bisa mencakup beberapa baris.  
*/  
echo "Ini adalah kode PHP.";  
?>
```

2.5. Variabel

Variabel digunakan untuk menyimpan informasi. Di PHP, variabel dimulai dengan tanda dolar (`$`) diikuti dengan nama variabel. Nama variabel bersifat case-sensitive. Tidak perlu mendeklarasikan tipe data variabel secara eksplisit; PHP akan menentukannya secara otomatis berdasarkan nilai yang diberikan.

Aturan Penamaan Variabel:

- Dimulai dengan `$`.
- Harus dimulai dengan huruf atau underscore (`_`).
- Hanya boleh mengandung karakter alfanumerik dan underscore (`A-z, 0-9, _`).
- Tidak boleh mengandung spasi.

Contoh:

```
<?php
$nama = "Yamin";
$umur = 20;
$_gaji_pokok = 5000000;

echo "Nama: " . $nama . "<br>";
echo "Umur: " . $umur . " tahun";
?>
```

2.6. Tipe Data

PHP mendukung beberapa tipe data:

- **String:** Urutan karakter, diapit oleh tanda kutip tunggal (' ') atau ganda (" ") .
- **Integer:** Bilangan bulat (positif atau negatif).
- **Float (Floating Point Number):** Bilangan desimal.
- **Boolean:** Hanya memiliki dua nilai: true atau false .
- **Array:** Menyimpan beberapa nilai dalam satu variabel.
- **Object:** Instance dari sebuah kelas.
- **NULL:** Variabel tanpa nilai.
- **Resource:** Variabel khusus yang menyimpan referensi ke sumber daya eksternal (misalnya, koneksi database).

Contoh:

```
<?php
$nama = "Alice";           // String
$umur = 25;                // Integer
$tinggi = 1.75;             // Float
$is_active = true;          // Boolean
$hobi = array("Membaca", "Menulis", "Coding"); // Array
$kosong = NULL;              // NULL

var_dump($nama);
var_dump($umur);
var_dump($tinggi);
var_dump($is_active);
```

```
var_dump($hobi);
var_dump($kosong);
?>
```

2.7. Operator

Operator digunakan untuk melakukan operasi pada variabel dan nilai.

- **Operator Aritmatika:** `+`, `-`, `*`, `/`, `%` (modulus), `**` (eksponensial).
- **Operator Penugasan:** `=`, `+=`, `-=`, `*=`, `/=`, `%=`, `.=` (konkatenasi string).
- **Operator Perbandingan:** `==` (sama dengan), `===` (identik), `!=` (tidak sama dengan), `<>`, `!==`, `<`, `>`, `<=`, `>=`.
- **Operator Logika:** `&&` (AND), `||` (OR), `!` (NOT).
- **Operator Increment/Decrement:** `++` (increment), `--` (decrement).

Contoh:



```
<?php
$a = 10;
$b = 3;

// Aritmatika
echo "Penjumlahan: " . ($a + $b) . "<br>"; // 13
echo "Modulus: " . ($a % $b) . "<br>"; // 1

// Penugasan
$a += 5; // $a sekarang 15
echo "Nilai a setelah +=: " . $a . "<br>";

// Perbandingan
var_dump($a == 15); // true
var_dump($a === "15"); // false (tipe data berbeda)

// Logika
$x = true;
$y = false;
var_dump($x && $y); // false

// Increment/Decrement
$c = 5;
echo "Pre-increment: " . (++$c) . "<br>"; // 6
```

```
echo "Post-increment: " . ($c++) . "<br>"; // 6, lalu $c jadi 7
echo "Nilai c sekarang: " . $c . "<br>"; // 7
?>
```

3. Pembahasan

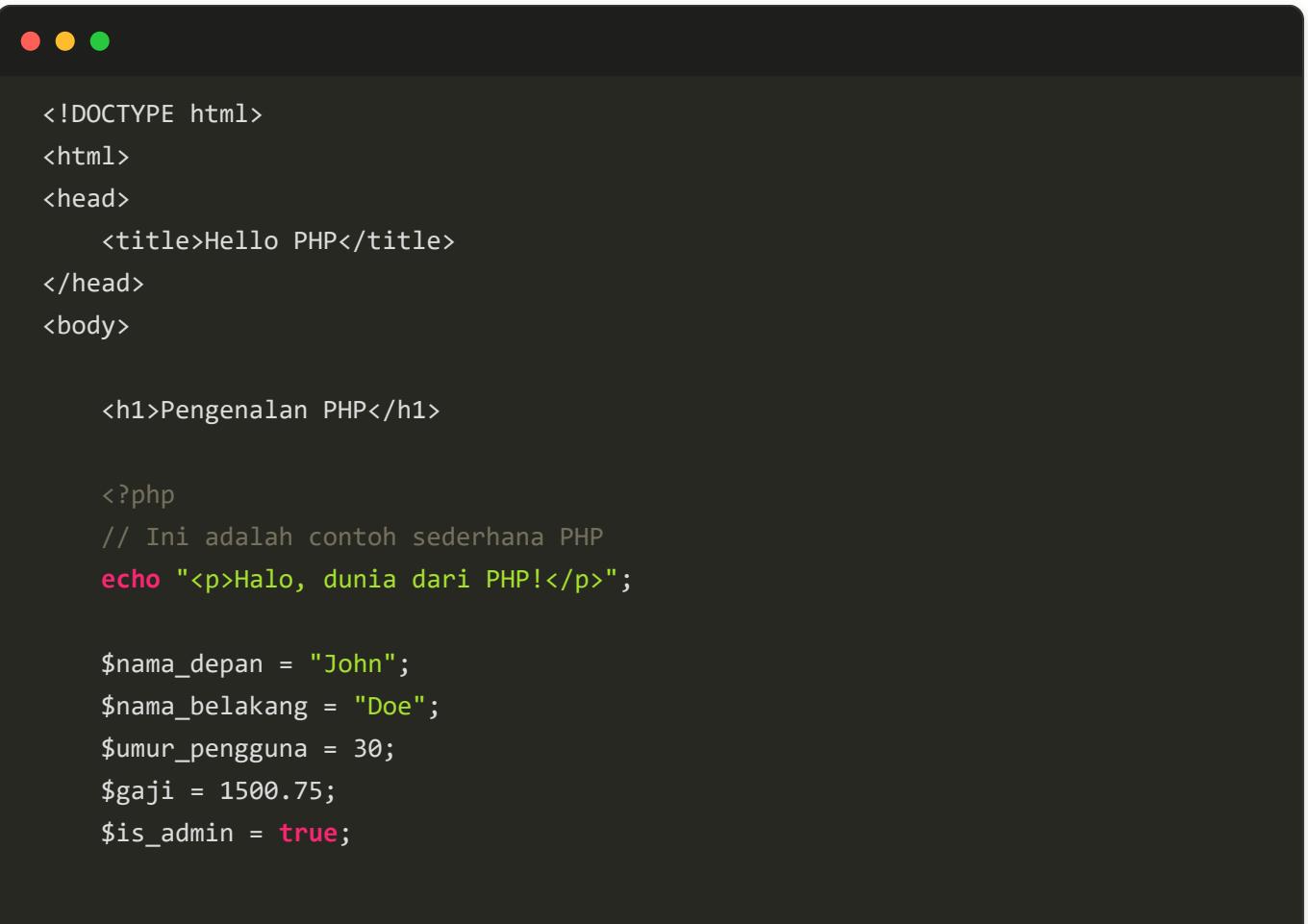
Pada praktikum ini, kita akan fokus pada penyiapan lingkungan pengembangan dan penulisan kode PHP dasar. Penting untuk memahami bagaimana PHP bekerja di sisi server dan bagaimana sintaks dasarnya digunakan untuk menampilkan informasi, menyimpan data dalam variabel, dan melakukan operasi sederhana.

Setelah menginstal XAMPP/LAMPP, Anda akan menempatkan file PHP Anda di direktori `htdocs` (untuk XAMPP) atau `www` (untuk LAMPP) di dalam folder instalasi XAMPP/LAMPP. Kemudian, Anda dapat mengakses file tersebut melalui browser dengan mengetik `http://localhost/nama_file_anda.php`.

Perhatikan perbedaan antara operator `==` dan `===`. Operator `==` hanya membandingkan nilai, sedangkan `===` membandingkan nilai dan tipe data. Ini adalah konsep penting dalam PHP untuk menghindari bug yang tidak terduga.

4. Contoh Kode

File: `hello.php`



```
<!DOCTYPE html>
<html>
<head>
    <title>Hello PHP</title>
</head>
<body>

    <h1>Pengenalan PHP</h1>

    <?php
        // Ini adalah contoh sederhana PHP
        echo "<p>Halo, dunia dari PHP!</p>";

        $nama_depan = "John";
        $nama_belakang = "Doe";
        $umur_pengguna = 30;
        $gaji = 1500.75;
        $is_admin = true;
```

```
echo "<p>Nama Lengkap: " . $nama_depan . " " . $nama_belakang . "</p>";
echo "<p>Umur: " . $umur_pengguna . " tahun</p>";
echo "<p>Gaji: $" . $gaji . "</p>";
echo "<p>Status Admin: " . ($is_admin ? "Ya" : "Tidak") . "</p>";

$angka1 = 20;
$angka2 = 7;

echo "<p>Hasil Penjumlahan: " . ($angka1 + $angka2) . "</p>";
echo "<p>Hasil Pengurangan: " . ($angka1 - $angka2) . "</p>";
";
echo "<p>Hasil Perkalian: " . ($angka1 * $angka2) . "</p>";
echo "<p>Hasil Pembagian: " . ($angka1 / $angka2) . "</p>";
echo "<p>Sisa Bagi (Modulus): " . ($angka1 % $angka2) . "</p>";

$counter = 10;
echo "<p>Nilai awal counter: " . $counter . "</p>";
$counter++;
echo "<p>Setelah increment: " . $counter . "</p>";
$counter--;
echo "<p>Setelah decrement: " . $counter . "</p>";
?>

</body>
</html>
```

5. Langkah-langkah Praktikum

1. Instalasi XAMPP/LAMPP:

- Jika belum, unduh dan instal XAMPP (untuk Windows/macOS) atau LAMPP (untuk Linux) dari situs resmi Apache Friends.
- Pastikan modul Apache dan MySQL berjalan di XAMPP Control Panel.

2. Buat Direktori Proyek:

- Di dalam folder instalasi XAMPP Anda, navigasikan ke direktori `htdocs`.
- Buat folder baru bernama `praktikum_php` di dalam `htdocs`.

3. Buat File PHP Pertama Anda:

- Buka editor teks favorit Anda (misalnya VS Code, Sublime Text, Notepad++).
- Salin dan tempel kode dari bagian

4. Contoh Kode

File: hello.php

```
<!DOCTYPE html>
<html>
<head>
    <title>Hello PHP</title>
</head>
<body>

    <h1>Pengenalan PHP</h1>

    <?php
        // Ini adalah contoh sederhana PHP
        echo "<p>Halo, dunia dari PHP!</p>";

        $nama_depan = "John";
        $nama_belakang = "Doe";
        $umur_pengguna = 30;
        $gaji = 1500.75;
        $is_admin = true;

        echo "<p>Nama Lengkap: " . $nama_depan . " " . $nama_belakang . "</p>";
        echo "<p>Umur: " . $umur_pengguna . " tahun</p>";
        echo "<p>Gaji: $" . $gaji . "</p>";
        echo "<p>Status Admin: " . ($is_admin ? "Ya" : "Tidak") . "</p>";

        $angka1 = 20;
        $angka2 = 7;

        echo "<p>Hasil Penjumlahan: " . ($angka1 + $angka2) . "</p>";
        echo "<p>Hasil Pengurangan: " . ($angka1 - $angka2) . "</p>";
        echo "<p>Hasil Perkalian: " . ($angka1 * $angka2) . "</p>";
        echo "<p>Hasil Pembagian: " . ($angka1 / $angka2) . "</p>";
        echo "<p>Sisa Bagi (Modulus): " . ($angka1 % $angka2) . "</p>";

        $counter = 10;
        echo "<p>Nilai awal counter: " . $counter . "</p>";
        $counter++;
        echo "<p>Setelah increment: " . $counter . "</p>";
        $counter--;
        echo "<p>Setelah decrement: " . $counter . "</p>";

    ?>
```

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

5. Langkah-langkah Praktikum

1. Instalasi XAMPP/LAMPP:

- Jika belum, unduh dan instal XAMPP (untuk Windows/macOS) atau LAMPP (untuk Linux) dari situs resmi Apache Friends.
- Pastikan modul Apache dan MySQL berjalan di XAMPP Control Panel.

2. Buat Direktori Proyek:

- Di dalam folder instalasi XAMPP Anda, navigasikan ke direktori `htdocs`.
- Buat folder baru bernama `praktikum_php` di dalam `htdocs`.

3. Buat File PHP Pertama Anda:

- Buka editor teks favorit Anda (misalnya VS Code, Sublime Text, Notepad++).
- Salin dan tempel kode dari bagian "Contoh Kode" di atas ke dalam file baru.
- Simpan file tersebut dengan nama `hello.php` di dalam folder `htdocs/praktikum_php`.

4. Akses Melalui Browser:

- Buka browser web Anda.
- Ketik `http://localhost/praktikum_php/hello.php` di bilah alamat dan tekan Enter.
- Amati output yang ditampilkan di browser. Pastikan semua informasi ditampilkan dengan benar.

5. Eksplorasi Variabel dan Operator:

- Modifikasi nilai variabel `$nama_depan`, `$nama_belakang`, `$umur_pengguna`, dan `$gaji` di `hello.php`.
- Ubah operasi aritmatika yang dilakukan pada `$angka1` dan `$angka2`.
- Tambahkan contoh penggunaan operator perbandingan dan logika.
- Simpan perubahan dan refresh halaman di browser untuk melihat hasilnya.

6. Coba Komentar:

- Tambahkan komentar satu baris dan multi-baris di berbagai bagian kode `hello.php` untuk menjelaskan fungsinya.
- Perhatikan bahwa komentar tidak akan muncul di output browser.

7. Verifikasi Tipe Data:

- Gunakan fungsi `var_dump()` untuk memeriksa tipe data dan nilai dari beberapa variabel yang Anda buat.
 - Contoh: `var_dump($nama_depan);`
 - Ini akan membantu Anda memahami bagaimana PHP secara otomatis menentukan tipe data.
-

Referensi

- [1] PHP.net. (n.d.). *What is PHP?*. Retrieved from <https://www.php.net/manual/en/intro-whatis.php>
- [2] Apache Friends. (n.d.). XAMPP. Retrieved from <https://www.apachefriends.org/>

Praktikum 2: Struktur Kontrol dan Fungsi PHP

1. Tujuan Pembelajaran

Setelah menyelesaikan praktikum ini, mahasiswa diharapkan mampu:

- Menggunakan struktur kondisional (if-else, elseif, switch) untuk mengontrol alur program.
- Menggunakan struktur perulangan (for, while, do-while, foreach) untuk mengulang blok kode.
- Membuat dan memanggil fungsi kustom di PHP.
- Memahami penggunaan parameter dan nilai kembalian pada fungsi.

2. Materi Pembelajaran

2.1. Struktur Kondisional

Struktur kondisional memungkinkan kita untuk mengeksekusi blok kode tertentu berdasarkan kondisi yang terpenuhi.

- **if-else** : Mengeksekusi satu blok kode jika kondisi **true**, dan blok lain jika **false**.
- **elseif** : Digunakan untuk menambahkan kondisi tambahan setelah **if**.
- **switch** : Digunakan untuk memilih salah satu dari banyak blok kode yang akan dieksekusi, berdasarkan nilai variabel.

Contoh **if-else** dan **elseif**:

```
<?php
$nilai = 75;

if ($nilai >= 90) {
    echo "Nilai Anda A";
} elseif ($nilai >= 80) {
    echo "Nilai Anda B";
} elseif ($nilai >= 70) {
    echo "Nilai Anda C";
} else {
    echo "Nilai Anda D";
}
?>
```

Contoh `switch` :

```
<?php
$hari = "Senin";

switch ($hari) {
    case "Senin":
        echo "Hari ini adalah hari kerja.";
        break;
    case "Sabtu":
    case "Minggu":
        echo "Hari ini adalah akhir pekan.";
        break;
    default:
        echo "Nama hari tidak valid.";
}
?>
```

2.2. Struktur Perulangan

Struktur perulangan digunakan untuk mengeksekusi blok kode berulang kali.

- `for` : Digunakan ketika jumlah iterasi diketahui.
- `while` : Mengeksekusi blok kode selama kondisi `true`.
- `do-while` : Mirip dengan `while`, tetapi blok kode dieksekusi setidaknya sekali sebelum kondisi diperiksa.
- `foreach` : Digunakan untuk mengulang elemen dalam array.

Contoh `for` :

```
<?php
for ($i = 1; $i <= 5; $i++) {
    echo "Perulangan ke-" . $i . "<br>";
}
?>
```

Contoh `while` :

```
<?php  
$i = 1;  
while ($i <= 5) {  
    echo "Perulangan ke-" . $i . "<br>";  
    $i++;  
}  
?>
```

Contoh `do-while` :

```
<?php  
$i = 1;  
do {  
    echo "Perulangan ke-" . $i . "<br>";  
    $i++;  
} while ($i <= 5);  
?>
```

Contoh `foreach` :

```
<?php  
$warna = array("Merah", "Hijau", "Biru");  
  
foreach ($warna as $value) {  
    echo $value . "<br>";  
}  
  
$mahasiswa = array("nama" => "Yamin", "umur" => 20, "jurusan" => "Informatika");  
foreach ($mahasiswa as $key => $value) {  
    echo $key . ":" . $value . "<br>";  
}  
?>
```

2.3. Fungsi PHP

Fungsi adalah blok kode yang dapat digunakan kembali. Fungsi membantu dalam mengorganisir kode, membuatnya lebih mudah dibaca, dan mengurangi duplikasi kode.

- **Deklarasi Fungsi:** Dimulai dengan kata kunci `function`, diikuti nama fungsi, tanda kurung `()`, dan blok kode `{}`.
- **Parameter Fungsi:** Variabel yang dilewatkan ke fungsi.
- **Nilai Kembalian:** Fungsi dapat mengembalikan nilai menggunakan kata kunci `return`.

Contoh Fungsi Sederhana:

```
<?php
function sapa($nama) {
    echo "Halo, " . $nama . "!";
}

sapa("Dunia"); // Memanggil fungsi
?>
```

Contoh Fungsi dengan Nilai Kembalian:

```
<?php
function tambah($angka1, $angka2) {
    $hasil = $angka1 + $angka2;
    return $hasil;
}

$jumlah = tambah(5, 3);
echo "Hasil penjumlahan: " . $jumlah; // Output: 8
?>
```

3. Pembahasan

Praktikum ini akan memperdalam pemahaman Anda tentang bagaimana mengontrol alur eksekusi program PHP. Struktur kondisional sangat penting untuk membuat keputusan dalam kode Anda, sementara perulangan memungkinkan Anda untuk memproses data dalam jumlah besar atau melakukan tugas berulang secara efisien. Fungsi adalah fondasi dari kode yang terorganisir dan modular, memungkinkan Anda untuk menulis kode yang lebih bersih dan mudah dikelola.

Saat menggunakan `switch`, pastikan untuk selalu menyertakan `break;` di akhir setiap `case` untuk mencegah

eksekusi kode berlanjut ke `case` berikutnya. Untuk perulangan, pilih jenis perulangan yang paling sesuai dengan kebutuhan Anda; `for` untuk iterasi yang diketahui jumlahnya, `while` untuk kondisi yang tidak pasti, dan `foreach` untuk array.

Fungsi adalah kunci untuk membuat kode yang dapat digunakan kembali. Pertimbangkan untuk membuat fungsi setiap kali Anda menemukan blok kode yang sama diulang di beberapa tempat. Ini akan membuat kode Anda lebih mudah untuk di-debug dan di-maintain.

4. Contoh Kode

File: kontrol_fungsi.php

```
<!DOCTYPE html>
<html>
<head>
    <title>Struktur Kontrol dan Fungsi</title>
</head>
<body>

    <h1>Struktur Kontrol PHP</h1>

    <?php
        // Contoh If-Elseif-Else
        $skor = 85;
        echo "<h2>Nilai: " . $skor . "</h2>";
        if ($skor >= 90) {
            echo "<p>Predikat: Sangat Baik (A)</p>";
        } elseif ($skor >= 75) {
            echo "<p>Predikat: Baik (B)</p>";
        } elseif ($skor >= 60) {
            echo "<p>Predikat: Cukup (C)</p>";
        } else {
            echo "<p>Predikat: Kurang (D)</p>";
        }

        echo "<hr>";

        // Contoh Switch
        $bulan = "Maret";
    
```

```
echo "<h2>Bulan: " . $bulan . "</h2>";
switch ($bulan) {
    case "Januari":
    case "Februari":
    case "Maret":
        echo "<p>Ini adalah kuartal pertama.</p>";
        break;
    case "April":
    case "Mei":
    case "Juni":
        echo "<p>Ini adalah kuartal kedua.</p>";
        break;
    default:
        echo "<p>Bulan tidak dikenali atau di luar kuartal pertama/kedua.</p>";
}
echo "<hr>";

<h1>Perulangan PHP</h1>

<?php
// Contoh For Loop
echo "<h2>For Loop (Angka 1 sampai 5)</h2>";
for ($i = 1; $i <= 5; $i++) {
    echo "<p>Angka: " . $i . "</p>";
}

echo "<hr>";

// Contoh While Loop
echo "<h2>While Loop (Angka Genap sampai 10)</h2>";
$j = 2;
while ($j <= 10) {
    echo "<p>Angka Genap: " . $j . "</p>";
    $j += 2;
}

echo "<hr>";

// Contoh Foreach Loop dengan Array Indeks
echo "<h2>Foreach Loop (Daftar Buah)</h2>";
$buah = array("Apel", "Pisang", "Mangga", "Jeruk");
echo "<ul>";
foreach ($buah as $item) {
```

```
    echo "<li>" . $item . "</li>";
}

echo "</ul>";

echo "<hr>";

// Contoh Foreach Loop dengan Array Asosiatif
echo "<h2>Foreach Loop (Informasi Produk)</h2>";
$produk = [
    "nama" => "Laptop",
    "harga" => 12000000,
    "stok" => 50
];
echo "<p>";
foreach ($produk as $key => $value) {
    echo ucfirst($key) . ": " . $value . "<br>";
}
echo "</p>";
?>

<hr>

<h1>Fungsi PHP</h1>

<?php
// Fungsi tanpa parameter dan nilai kembalian
function salam() {
    echo "<p>Selamat datang di modul PHP!</p>";
}
salam();

// Fungsi dengan parameter
function ucapanSalam($nama, $waktu) {
    echo "<p>Selamat " . $waktu . ", " . $nama . " !</p>";
}
ucapanSalam("Andi", "Pagi");
ucapanSalam("Siti", "Sore");

// Fungsi dengan nilai kembalian
function hitungLuasPersegiPanjang($panjang, $lebar) {
    $luas = $panjang * $lebar;
    return $luas;
}
```

```

$p = 10;
$l = 5;
$luas_persegi = hitungLuasPersegiPanjang($p, $l);
echo "<p>Luas persegi panjang dengan panjang " . $p . " dan lebar " . $l . " adalah " . $luas_persegi . "</p>";

// Fungsi dengan parameter default
function ucapanTerimaKasih($penerima = "Anda") {
    echo "<p>Terima kasih, " . $penerima . " !</p>";
}
ucapanTerimaKasih("Yamin");
ucapanTerimaKasih(); // Menggunakan nilai default
?>

</body>
</html>

```

5. Langkah-langkah Praktikum

1. Buat File Baru:

- Buka editor teks Anda.
- Salin dan tempel kode dari bagian "Contoh Kode" di atas ke dalam file baru.
- Simpan file tersebut dengan nama `kontrol_fungsi.php` di dalam folder `htdocs/praktikum_php` yang telah Anda buat sebelumnya.

2. Akses Melalui Browser:

- Buka browser web Anda.
- Ketik `http://localhost/praktikum_php/kontrol_fungsi.php` di bilah alamat dan tekan Enter.
- Amati output yang ditampilkan di browser. Pastikan semua contoh struktur kontrol dan fungsi berjalan sesuai harapan.

3. Eksplorasi Struktur Kondisional:

- Ubah nilai variabel `$skor` di awal kode dan amati bagaimana output predikat berubah.
- Coba ubah nilai variabel `$bulan` dan lihat bagaimana blok `switch` merespons.
- Tambahkan kondisi `elseif` baru atau `case` baru untuk memperluas logika.

4. Eksplorasi Struktur Perulangan:

- Ubah batas atas atau bawah pada `for` dan `while` loop.
- Tambahkan elemen baru ke array `$buah` atau `$produk` dan lihat bagaimana `foreach` loop secara otomatis memprosesnya.

- Coba gunakan `break` atau `continue` di dalam perulangan untuk mengontrol alur lebih lanjut.

5. Eksplorasi Fungsi:

- Buat fungsi baru yang menerima tiga angka sebagai parameter dan mengembalikan rataratanya.
 - Panggil fungsi tersebut dengan nilai yang berbeda dan tampilkan hasilnya.
 - Coba buat fungsi yang tidak mengembalikan nilai (void function) tetapi langsung mencetak output.
-

Referensi

- [1] W3Schools. (n.d.). *PHP If...Else...Elseif Statements*. Retrieved from https://www.w3schools.com/php/php_if_else.asp
- [2] W3Schools. (n.d.). *PHP Loops*. Retrieved from https://www.w3schools.com/php/php_looping.asp
- [3] W3Schools. (n.d.). *PHP Functions*. Retrieved from https://www.w3schools.com/php/php_functions.asp

Praktikum 3: Pengenalan MySQL dan Operasi Dasar Database

1. Tujuan Pembelajaran

Setelah menyelesaikan praktikum ini, mahasiswa diharapkan mampu:

- Memahami konsep dasar database relasional.
- Mengenal komponen-komponen dasar dalam database (tabel, kolom, baris).
- Melakukan operasi DDL (Data Definition Language) untuk membuat dan memodifikasi struktur database dan tabel.
- Melakukan operasi DML (Data Manipulation Language) untuk memasukkan, memilih, memperbarui, dan menghapus data.
- Menggunakan phpMyAdmin sebagai alat bantu pengelolaan database.

2. Materi Pembelajaran

2.1. Konsep Dasar Database Relasional

Database relasional adalah jenis database yang menyimpan dan menyediakan akses ke titik data yang saling terkait. Ini didasarkan pada model relasional, cara intuitif dan lugas untuk merepresentasikan data dalam tabel. Dalam database relasional, setiap baris dalam tabel adalah catatan dengan ID unik yang disebut kunci. Kolom tabel berisi atribut data, dan setiap catatan biasanya memiliki nilai untuk setiap atribut, membuat hubungan antar titik data mudah teridentifikasi. [1]

2.2. Komponen Dasar Database

- **Database:** Kumpulan terorganisir dari data yang saling terkait.
- **Tabel:** Struktur yang menyimpan data dalam baris dan kolom. Setiap tabel mewakili entitas tertentu (misalnya, pengguna, produk).
- **Kolom (Field/Attribute):** Bagian vertikal dari tabel yang menyimpan jenis data tertentu (misalnya, nama, umur, email). Setiap kolom memiliki tipe data (misalnya, VARCHAR, INT, DATE).
- **Baris (Record/Tuple):** Bagian horizontal dari tabel yang mewakili satu entri data lengkap untuk entitas tersebut.
- **Primary Key:** Kolom atau set kolom yang secara unik mengidentifikasi setiap baris dalam tabel. Nilainya harus unik dan tidak boleh NULL.
- **Foreign Key:** Kolom atau set kolom dalam satu tabel yang merujuk ke Primary Key di tabel lain, digunakan untuk membuat hubungan antar tabel.

2.3. Tipe Data MySQL

MySQL mendukung berbagai tipe data untuk kolom, antara lain:

- **Numerik:** INT, TINYINT, BIGINT, FLOAT, DOUBLE, DECIMAL .

- **String:** `VARCHAR` , `CHAR` , `TEXT` , `BLOB` .
- **Tanggal dan Waktu:** `DATE` , `TIME` , `DATETIME` , `TIMESTAMP` .

2.4. Data Definition Language (DDL)

DDL digunakan untuk mendefinisikan atau memodifikasi struktur database dan objeknya.

- `CREATE DATABASE` : Membuat database baru.



```
CREATE DATABASE nama_database;
```

- `CREATE TABLE` : Membuat tabel baru dalam database.



```
CREATE TABLE nama_tabel (
    kolom1 TIPE_DATA(ukuran) BATASAN,
    kolom2 TIPE_DATA(ukuran) BATASAN,
    PRIMARY KEY (kolom1)
);
```

- `ALTER TABLE` : Memodifikasi struktur tabel yang sudah ada (menambah/menghapus kolom, mengubah tipe data).



```
ALTER TABLE nama_tabel ADD kolom_baru TIPE_DATA;
ALTER TABLE nama_tabel DROP COLUMN kolom_lama;
ALTER TABLE nama_tabel MODIFY COLUMN kolom_lama TIPE_DATA_BARU;
```

- `DROP DATABASE` : Menghapus database.



```
DROP DATABASE nama_database;
```

- `DROP TABLE` : Menghapus tabel.



```
DROP TABLE nama_tabel;
```

2.5. Data Manipulation Language (DML)

DML digunakan untuk memanipulasi data di dalam tabel.

- **INSERT INTO** : Memasukkan baris data baru ke dalam tabel.



```
INSERT INTO nama_tabel (kolom1, kolom2) VALUES (nilai1, nilai2);
```

- **SELECT** : Mengambil data dari satu atau lebih tabel.



```
SELECT kolom1, kolom2 FROM nama_tabel WHERE kondisi;  
SELECT * FROM nama_tabel;
```

- **UPDATE** : Memperbarui data yang sudah ada dalam tabel.



```
UPDATE nama_tabel SET kolom1 = nilai_baru WHERE kondisi;
```

- **DELETE FROM** : Menghapus baris data dari tabel.



```
DELETE FROM nama_tabel WHERE kondisi;
```

2.6. phpMyAdmin

phpMyAdmin adalah alat berbasis web yang ditulis dalam PHP, dirancang untuk menangani administrasi MySQL melalui browser web. Ini menyediakan antarmuka grafis untuk melakukan berbagai operasi database

seperti membuat/menghapus database, membuat/memodifikasi/menghapus tabel, menjalankan query SQL, mengelola pengguna, dan banyak lagi. [2]

3. Pembahasan

Pada praktikum ini, Anda akan berinteraksi langsung dengan MySQL menggunakan phpMyAdmin. Meskipun nantinya kita akan terhubung ke MySQL melalui PHP, memahami dasar-dasar SQL dan cara mengelola database secara manual melalui phpMyAdmin adalah langkah penting. Ini akan membantu Anda memvisualisasikan struktur data dan memahami bagaimana data disimpan dan diambil.

Pastikan XAMPP/LAMPP Anda berjalan dan modul MySQL aktif. Anda dapat mengakses phpMyAdmin melalui browser dengan mengetik `http://localhost/phpmyadmin`.

Saat menulis query SQL, perhatikan sintaksnya. Kesalahan kecil seperti tanda kutip yang salah atau koma yang hilang dapat menyebabkan error. Selalu mulai dengan operasi `SELECT` untuk memastikan Anda memahami data sebelum melakukan `UPDATE` atau `DELETE` yang dapat mengubah data secara permanen.

4. Contoh Kode (SQL)

Berikut adalah contoh-contoh query SQL yang akan Anda gunakan di phpMyAdmin:

1. Membuat Database:

```
CREATE DATABASE db_akademik;
```

2. Menggunakan Database:

```
USE db_akademik;
```

3. Membuat Tabel `mahasiswa`:

```
CREATE TABLE mahasiswa (
    id INT(11) NOT NULL AUTO_INCREMENT,
    nim VARCHAR(10) NOT NULL UNIQUE,
    nama VARCHAR(100) NOT NULL,
    jurusan VARCHAR(50),
    email VARCHAR(100) UNIQUE,
```

```
PRIMARY KEY (id)
);
```

4. Memasukkan Data ke Tabel `mahasiswa`:

```
INSERT INTO mahasiswa (nim, nama, jurusan, email) VALUES
('2023001', 'Yamin bae', 'Teknik Informatika', 'yamin.s@example.com'),
('2023002', 'Siti Aminah', 'Sistem Informasi', 'siti.a@example.com'),
('2023003', 'Agus Salim', 'Teknik Informatika', 'agus.s@example.com');
```

5. Menampilkan Semua Data dari Tabel `mahasiswa`:

```
SELECT * FROM mahasiswa;
```

6. Menampilkan Data Mahasiswa dengan Jurusan Tertentu:

```
SELECT nim, nama FROM mahasiswa WHERE jurusan = 'Teknik Informatika';
```

7. Memperbarui Data Mahasiswa:

```
UPDATE mahasiswa SET email = 'yamin.bae@example.com' WHERE nim = '2023001';
```

8. Menghapus Data Mahasiswa:

```
DELETE FROM mahasiswa WHERE nim = '2023003';
```

9. Menghapus Tabel:



```
DROP TABLE mahasiswa;
```

10. Menghapus Database:



```
DROP DATABASE db_akademik;
```

5. Langkah-langkah Praktikum

1. Akses phpMyAdmin:

- Pastikan Apache dan MySQL di XAMPP Control Panel Anda berjalan.
- Buka browser web Anda dan ketik `http://localhost/phpmyadmin` di bilah alamat.

2. Membuat Database Baru:

- Di halaman phpMyAdmin, klik tab `Databases` atau klik `New` di sidebar kiri.
- Masukkan nama database `db_akademik` dan klik `Create`.

3. Membuat Tabel `mahasiswa`:

- Setelah database `db_akademik` dibuat, klik nama database tersebut di sidebar kiri.
- Di bagian `Create table`, masukkan nama tabel `mahasiswa` dan jumlah kolom (misalnya 5).
- Klik `Create` atau `Go`.
- Isi detail kolom sesuai dengan contoh `CREATE TABLE mahasiswa` di bagian "Contoh Kode".
Pastikan untuk mengatur `id` sebagai `PRIMARY KEY` dan `AUTO_INCREMENT`.
- Klik `Save`.

4. Memasukkan Data:

- Setelah tabel `mahasiswa` dibuat, klik tab `Insert`.
- Masukkan data untuk beberapa mahasiswa sesuai dengan contoh `INSERT INTO` di bagian "Contoh Kode". Anda bisa menambahkan lebih banyak baris jika diinginkan.
- Klik `Go` untuk setiap baris data yang dimasukkan.

5. Melihat Data:

- Klik tab `Browse` pada tabel `mahasiswa` untuk melihat semua data yang telah Anda masukkan.
- Coba jalankan query `SELECT * FROM mahasiswa;` di tab `SQL` untuk melihat hasil yang sama.

6. Mengambil Data dengan Kondisi:

- Di tab **SQL**, jalankan query
`SELECT nim, nama FROM mahasiswa WHERE jurusan = 'Teknik Informatika';`.
- Amati hasil yang hanya menampilkan mahasiswa dari jurusan Teknik Informatika.

7. Memperbarui Data:

- Di tab **SQL**, jalankan query
`UPDATE mahasiswa SET email = 'Yamin.bae@example.com' WHERE nim = '2023001';`.
- Kembali ke tab **Browse** untuk memverifikasi bahwa email mahasiswa dengan NIM 2023001 telah diperbarui.

8. Menghapus Data:

- Di tab **SQL**, jalankan query `DELETE FROM mahasiswa WHERE nim = '2023003';`.
- Kembali ke tab **Browse** untuk memverifikasi bahwa mahasiswa dengan NIM 2023003 telah dihapus.

9. Eksplorasi DDL Lanjutan (Opsional):

- Coba tambahkan kolom baru ke tabel **mahasiswa** menggunakan **ALTER TABLE** (misalnya, kolom **tanggal_lahir** dengan tipe **DATE**).
- Coba hapus kolom yang sudah ada.
- **PERHATIAN:** Jangan menghapus tabel atau database kecuali Anda yakin dan telah mencadangkan data penting.

Referensi

- [1] Oracle. (n.d.). *What is a Relational Database?*. Retrieved from <https://www.oracle.com/database/what-is-relational-database/>
- [2] phpMyAdmin. (n.d.). *About*. Retrieved from <https://www.phpmyadmin.net/about/>

Praktikum 4: Form Handling dan Validasi

1. Tujuan Pembelajaran

Setelah menyelesaikan praktikum ini, mahasiswa diharapkan mampu:

- Memahami cara kerja form HTML dalam mengirimkan data ke server PHP.
- Menggunakan superglobal `$_GET` dan `$_POST` untuk mengambil data dari form.
- Melakukan validasi data sederhana di sisi server untuk memastikan integritas data.
- Menampilkan pesan kesalahan (error messages) kepada pengguna.

2. Materi Pembelajaran

2.1. Form HTML dan Metode Pengiriman Data

Form HTML adalah elemen penting untuk mengumpulkan input dari pengguna. Data yang dimasukkan ke dalam form dapat dikirimkan ke server menggunakan dua metode utama: `GET` dan `POST`.

- **Metode `GET`:**

- Mengirimkan data form sebagai bagian dari URL (query string).
- Data terlihat di URL, sehingga tidak cocok untuk informasi sensitif.
- Jumlah data yang dapat dikirim terbatas.
- Cocok untuk pencarian atau filter data.

- **Metode `POST`:**

- Mengirimkan data form dalam badan (body) permintaan HTTP.
- Data tidak terlihat di URL, lebih aman untuk informasi sensitif.
- Tidak ada batasan ukuran data yang dikirim.
- Cocok untuk pengiriman data form yang besar atau sensitif (misalnya, login, pendaftaran).

Sintaks Dasar Form HTML:

```
<form action="proses.php" method="POST">
    <label for="nama">Nama:</label><br>
    <input type="text" id="nama" name="nama"><br>
    <label for="email">Email:</label><br>
    <input type="email" id="email" name="email"><br>
    <input type="submit" value="Kirim">
</form>
```

- `action` : Menentukan URL tujuan tempat data form akan dikirim.
- `method` : Menentukan metode HTTP (`GET` atau `POST`) yang digunakan untuk mengirim data.
- `name` : Atribut penting untuk input field, digunakan oleh PHP untuk mengidentifikasi data yang dikirim.

2.2. Mengambil Data Form dengan PHP Superglobals

PHP menyediakan variabel superglobal untuk mengakses data yang dikirimkan melalui form:

- `$_GET` : Array asosiatif yang berisi semua data yang dikirimkan melalui metode `GET`.
- `$_POST` : Array asosiatif yang berisi semua data yang dikirimkan melalui metode `POST`.
- `$_REQUEST` : Array asosiatif yang berisi data dari `$_GET`, `$_POST`, dan `$_COOKIE`. Sebaiknya hindari penggunaannya untuk menghindari ambiguitas dan potensi masalah keamanan.

Contoh Mengambil Data:

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $nama = $_POST["nama"];
    $email = $_POST["email"];

    echo "Nama Anda: " . $nama . "<br>";
    echo "Email Anda: " . $email;
}
?>
```

- `$_SERVER["REQUEST_METHOD"]` : Digunakan untuk memeriksa apakah permintaan HTTP adalah `POST` atau `GET`.

2.3. Validasi Data Sisi Server

Validasi data adalah proses memastikan bahwa data yang dimasukkan oleh pengguna memenuhi kriteria tertentu sebelum diproses atau disimpan. Validasi sisi server sangat penting karena validasi sisi klien (JavaScript) dapat dilewati.

Beberapa fungsi PHP yang berguna untuk validasi:

- `empty()` : Memeriksa apakah variabel kosong (nilai `0`, `false`, `NULL`, string kosong, array kosong dianggap kosong).
- `isset()` : Memeriksa apakah variabel telah diatur dan bukan `NULL`.

- `trim()` : Menghapus spasi putih (spasi, tab, newline) dari awal dan akhir string.
- `stripslashes()` : Menghapus backslash yang ditambahkan oleh fungsi `addslashes()`.
- `htmlspecialchars()` : Mengubah karakter khusus menjadi entitas HTML untuk mencegah serangan XSS (Cross-Site Scripting).
- `filter_var()` : Memfilter variabel dengan filter tertentu (misalnya, `FILTER_VALIDATE_EMAIL`, `FILTER_VALIDATE_INT`).

Contoh Validasi Sederhana:

```
<?php
$namaErr = $emailErr = "";
$nama = $email = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (empty($_POST["nama"])) {
        $namaErr = "Nama harus diisi";
    } else {
        $nama = htmlspecialchars(trim($_POST["nama"]));
        // Tambahkan validasi lain jika diperlukan (misal: hanya huruf)
        if (!preg_match("/^([a-zA-Z ])*$/",$nama)) {
            $namaErr = "Hanya huruf dan spasi yang diizinkan";
        }
    }

    if (empty($_POST["email"])) {
        $emailErr = "Email harus diisi";
    } else {
        $email = htmlspecialchars(trim($_POST["email"]));
        // Validasi format email
        if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
            $emailErr = "Format email tidak valid";
        }
    }
}

// Jika tidak ada error, proses data
if (empty($namaErr) && empty($emailErr)) {
    echo "Data berhasil divalidasi:<br>";
    echo "Nama: " . $nama . "<br>";
    echo "Email: " . $email;
} else {
    echo "Ada kesalahan validasi.<br>";
}
```

```
    }
}
```

3. Pembahasan

Pada praktikum ini, Anda akan membangun form HTML sederhana dan mengolah data yang dikirimkan menggunakan PHP. Fokus utama adalah pada bagaimana data form diterima oleh server dan bagaimana melakukan validasi dasar untuk memastikan data yang masuk bersih dan sesuai format yang diharapkan. Ini adalah langkah krusial dalam membangun aplikasi web yang aman dan andal.

Penting untuk selalu melakukan validasi sisi server, bahkan jika Anda sudah melakukan validasi sisi klien dengan JavaScript. Validasi sisi klien hanya untuk kenyamanan pengguna, sedangkan validasi sisi server adalah lapisan keamanan yang sebenarnya. Selalu bersihkan input pengguna menggunakan fungsi seperti `htmlspecialchars()` untuk mencegah serangan XSS.

4. Contoh Kode

File: `form_pendaftaran.php`

```
<!DOCTYPE html>
<html>
<head>
    <title>Form Pendaftaran</title>
    <style>
        .error {color: #FF0000;}
    </style>
</head>
<body>

    <?php
        // Mendefinisikan variabel dan menginisialisasi dengan nilai kosong
        $namaErr = $emailErr = $websiteErr = $komentarErr = $genderErr = "";
        $nama = $email = $website = $komentar = $gender = "";

        if ($_SERVER["REQUEST_METHOD"] == "POST") {
            // Validasi Nama
            if (empty($_POST["nama"])) {
                $namaErr = "Nama harus diisi";
            } else {
                $nama = test_input($_POST["nama"]);
            }
        }
    </?php>
```

```
// Cek apakah nama hanya mengandung huruf dan spasi
if (!preg_match("/^[\a-zA-Z ]*$/,$nama)) {
    $namaErr = "Hanya huruf dan spasi yang diizinkan";
}

// Validasi Email
if (empty($_POST["email"])) {
    $emailErr = "Email harus diisi";
} else {
    $email = test_input($_POST["email"]);
    // Cek apakah format email valid
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        $emailErr = "Format email tidak valid";
    }
}

// Validasi Website (opsional)
if (empty($_POST["website"])) {
    $website = "";
} else {
    $website = test_input($_POST["website"]);
    // Cek apakah format URL valid
    if (!preg_match("/\b(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\%?=~_|!:,.,.;]/i",
        $websiteErr = "Format URL tidak valid";
    }
}

// Validasi Komentar (opsional)
if (empty($_POST["komentar"])) {
    $komentar = "";
} else {
    $komentar = test_input($_POST["komentar"]);
}

// Validasi Gender
if (empty($_POST["gender"])) {
    $genderErr = "Gender harus dipilih";
} else {
    $gender = test_input($_POST["gender"]);
}

function test_input($data) {
```

```
$data = trim($data);
$data = stripslashes($data);
$data = htmlspecialchars($data);
return $data;
}

?>

<h2>Form Pendaftaran Sederhana</h2>
<p><span class="error">* Bidang yang wajib diisi</span></p>
<form method="post" action=<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>
    Nama: <input type="text" name="nama" value=<?php echo $nama;?>">
    <span class="error">* <?php echo $namaErr;?></span>
    <br><br>
    E-mail: <input type="text" name="email" value=<?php echo $email;?>">
    <span class="error">* <?php echo $emailErr;?></span>
    <br><br>
    Website: <input type="text" name="website" value=<?php echo $website;?>">
    <span class="error"><?php echo $websiteErr;?></span>
    <br><br>
    Komentar: <textarea name="komentar" rows="5" cols="40"><?php echo $komentar;
    <br><br>
    Gender:
    <input type="radio" name="gender" <?php if (isset($gender) && $gender=="female") { ?>
    <input type="radio" name="gender" <?php if (isset($gender) && $gender=="male") { ?>
    <span class="error">* <?php echo $genderErr;?></span>
    <br><br>
    <input type="submit" name="submit" value="Submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST" && empty($namaErr) && empty($emailErr) && empty($websiteErr)) {
    echo "<h2>Input Anda:</h2>";
    echo "Nama: " . $nama . "<br>";
    echo "Email: " . $email . "<br>";
    echo "Website: " . $website . "<br>";
    echo "Komentar: " . $komentar . "<br>";
    echo "Gender: " . $gender . "<br>";
}
?>

</body>
</html>
```

5. Langkah-langkah Praktikum

1. Buat File Form:

- Buka editor teks Anda.
- Salin dan tempel kode dari bagian "Contoh Kode" di atas ke dalam file baru.
- Simpan file tersebut dengan nama `form_pendaftaran.php` di dalam folder `htdocs/praktikum_php`.

2. Akses Melalui Browser:

- Buka browser web Anda dan ketik `http://localhost/praktikum_php/form_pendaftaran.php`.
- Amati form yang ditampilkan.

3. Uji Validasi Kosong:

- Coba langsung klik tombol `Submit` tanpa mengisi data apapun.
- Perhatikan pesan kesalahan yang muncul untuk bidang yang wajib diisi.

4. Uji Validasi Format:

- Isi nama dengan angka atau karakter khusus (misalnya `123Nama`). Amati pesan kesalahan.
- Isi email dengan format yang salah (misalnya `emailku@`). Amati pesan kesalahan.
- Isi website dengan format URL yang salah. Amati pesan kesalahan.

5. Uji Pengiriman Data Berhasil:

- Isi semua bidang dengan data yang valid.
- Klik `Submit`.
- Amati output yang menampilkan data yang telah Anda masukkan di bawah form.

6. Eksplorasi Kode:

- Pelajari fungsi `test_input()` yang digunakan untuk membersihkan dan memvalidasi input.
- Coba tambahkan bidang input baru ke form (misalnya, nomor telepon) dan tambahkan validasi yang sesuai di PHP.
- Eksplorasi penggunaan `$_GET` dengan mengubah `method="post"` menjadi `method="get"` di tag form dan amati perubahan pada URL setelah submit.

Referensi

- [1] W3Schools. (n.d.). *PHP Forms*. Retrieved from https://www.w3schools.com/php/php_forms.asp
- [2] W3Schools. (n.d.). *PHP Form Validation*. Retrieved from https://www.w3schools.com/php/php_form_validation.asp

Praktikum 5: Koneksi PHP ke MySQL dan CRUD

1. Tujuan Pembelajaran

Setelah menyelesaikan praktikum ini, mahasiswa diharapkan mampu:

- Membuat koneksi ke database MySQL dari aplikasi PHP.
- Melakukan operasi Create, Read, Update, dan Delete (CRUD) pada data di database menggunakan PHP.
- Memahami pentingnya prepared statements untuk keamanan.

2. Materi Pembelajaran

2.1. Koneksi PHP ke MySQL

PHP menyediakan beberapa ekstensi untuk berinteraksi dengan database MySQL. Dua yang paling umum adalah MySQLi (MySQL Improved) dan PDO (PHP Data Objects). PDO lebih fleksibel karena mendukung berbagai jenis database, sedangkan MySQLi khusus untuk MySQL.

MySQLi (Object-Oriented Style):

```
<?php
$servername = "localhost";
$username = "root";
$password = ""; // Kosong jika tidak ada password
$dbname = "db_akademik";

// Membuat koneksi
$conn = new mysqli($servername, $username, $password, $dbname);

// Mengecek koneksi
if ($conn->connect_error) {
    die("Koneksi gagal: " . $conn->connect_error);
}
echo "Koneksi berhasil";

// Menutup koneksi
$conn->close();
?>
```

PDO (PHP Data Objects):

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "db_akademik";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
    // Set mode error PDO ke exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Koneksi berhasil";
} catch(PDOException $e) {
    echo "Koneksi gagal: " . $e->getMessage();
}

// Menutup koneksi (tidak perlu secara eksplisit untuk PDO, koneksi akan ditutup saat
$conn = null;
?>
```

PDO direkomendasikan karena mendukung berbagai database dan memiliki fitur keamanan yang lebih baik (prepared statements secara default).

2.2. Operasi CRUD (Create, Read, Update, Delete)**2.2.1. Create (INSERT)**

Memasukkan data baru ke dalam tabel.

MySQLi (Prepared Statements):

```
<?php
// ... kode koneksi ...

$stmt = $conn->prepare("INSERT INTO mahasiswa (nim, nama, jurusan, email) VALUES (?, ?, ?, ?)");
$stmt->bind_param("ssss", $nim, $nama, $jurusan, $email);

// Set parameter dan eksekusi
```

```
$nim = "2023004";
$nama = "Dian Pertiwi";
$jurusan = "Sistem Informasi";
$email = "dian.p@example.com";
$stmt->execute();

echo "Data baru berhasil ditambahkan";

$stmt->close();
$conn->close();
?>
```

PDO (Prepared Statements):

```
<?php
// ... kode koneksi ...

$stmt = $conn->prepare("INSERT INTO mahasiswa (nim, nama, jurusan, email) VALUES (:nim, :nama, :jurusan, :email)");
$stmt->bindParam(":nim", $nim);
$stmt->bindParam(":nama", $nama);
$stmt->bindParam(":jurusan", $jurusan);
$stmt->bindParam(":email", $email);

// Set parameter dan eksekusi
nim = "2023005";
$nama = "Eko Prasetyo";
$jurusan = "Teknik Informatika";
$email = "eko.p@example.com";
$stmt->execute();

echo "Data baru berhasil ditambahkan";

$conn = null;
?>
```

2.2.2. Read (SELECT)

Mengambil data dari tabel.

MySQLi (Object-Oriented Style):

```
<?php
// ... kode koneksi ...

$sql = "SELECT id, nim, nama, jurusan, email FROM mahasiswa";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // Output data dari setiap baris
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - NIM: " . $row["nim"]. " - Nama: " . $row["nama"]
    }
} else {
    echo "0 hasil";
}
$conn->close();
?>
```

PDO:

```
<?php
// ... kode koneksi ...

$stmt = $conn->prepare("SELECT id, nim, nama, jurusan, email FROM mahasiswa");
$stmt->execute();

// Set the resulting array to associative
$result = $stmt->setFetchMode(PDO::FETCH_ASSOC);

foreach(new RecursiveArrayIterator($stmt->fetchAll()) as $k=>$v) {
    echo "id: " . $v["id"]. " - NIM: " . $v["nim"]. " - Nama: " . $v["nama"]. " - Ju
}

$conn = null;
?>
```

2.2.3. Update (UPDATE)

Memperbarui data yang sudah ada.

MySQLi (Prepared Statements):

```
<?php
// ... kode koneksi ...

$stmt = $conn->prepare("UPDATE mahasiswa SET jurusan = ? WHERE nim = ?");
$stmt->bind_param("ss", $jurusan, $nim);

$jurusan = "Manajemen Informatika";
$nim = "2023001";
$stmt->execute();

echo $stmt->affected_rows . " baris berhasil diperbarui";

$stmt->close();
$conn->close();
?>
```

PDO (Prepared Statements):

```
<?php
// ... kode koneksi ...

$stmt = $conn->prepare("UPDATE mahasiswa SET jurusan = :jurusan WHERE nim = :nim");
$stmt->bindParam(":jurusan", $jurusan);
$stmt->bindParam(":nim", $nim);

$jurusan = "Manajemen Informatika";
nim = "2023002";
$stmt->execute();

echo $stmt->rowCount() . " baris berhasil diperbarui";

$conn = null;
?>
```

2.2.4. Delete (DELETE)

Menghapus data dari tabel.

MySQLi (Prepared Statements):

```
<?php
// ... kode koneksi ...

$stmt = $conn->prepare("DELETE FROM mahasiswa WHERE nim = ?");
$stmt->bind_param("s", $nim);

$nim = "2023004";
$stmt->execute();

echo $stmt->affected_rows . " baris berhasil dihapus";

$stmt->close();
$conn->close();
?>
```

PDO (Prepared Statements):

```
<?php
// ... kode koneksi ...

$stmt = $conn->prepare("DELETE FROM mahasiswa WHERE nim = :nim");
$stmt->bindParam(":nim", $nim);

$nim = "2023005";
$stmt->execute();

echo $stmt->rowCount() . " baris berhasil dihapus";

$conn = null;
?>
```

2.3. Prepared Statements

Prepared statements adalah fitur yang digunakan untuk mengeksekusi pernyataan SQL yang sama berulang kali dengan efisiensi tinggi dan, yang lebih penting, dengan aman. Mereka sangat penting untuk mencegah serangan SQL Injection. [1]

Bagaimana cara kerjanya?

1. **Prepare:** Template SQL dikirim ke database server dengan placeholder (misalnya `?` atau `:nama`). Server mengkompilasi template ini dan mengembalikan objek statement.
2. **Bind:** Nilai-nilai aktual untuk placeholder diikat ke objek statement.
3. **Execute:** Statement dieksekusi dengan nilai-nilai yang terikat. Nilai-nilai ini dikirim secara terpisah dari template SQL, sehingga tidak mungkin bagi penyerang untuk menyuntikkan kode SQL berbahaya.

3. Pembahasan

Praktikum ini adalah inti dari pengembangan aplikasi web dinamis, yaitu bagaimana PHP berinteraksi dengan database. Anda akan belajar cara membuat koneksi yang stabil dan aman, serta bagaimana melakukan semua operasi dasar (CRUD) yang diperlukan untuk mengelola data pengguna atau aplikasi. Penggunaan prepared statements adalah praktik terbaik yang harus selalu Anda terapkan untuk melindungi aplikasi Anda dari SQL Injection, salah satu kerentanan keamanan web paling umum.

Disarankan untuk menggunakan PDO karena fleksibilitas dan fitur keamanannya yang lebih baik. Pastikan Anda telah membuat database `db_akademik` dan tabel `mahasiswa` dari Praktikum 3 sebelum memulai praktikum ini. Jika Anda belum memiliki data di tabel `mahasiswa`, Anda bisa menambahkannya melalui phpMyAdmin atau menggunakan skrip `INSERT` yang akan Anda buat.

4. Contoh Kode

Untuk praktikum ini, kita akan membuat beberapa file PHP terpisah untuk setiap operasi CRUD, dan satu file konfigurasi koneksi.

File: `koneksi.php`

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "db_akademik";

$conn = null;

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

```
// echo "Koneksi berhasil"; // Komentar ini setelah berhasil koneksi
} catch(PDOException $e) {
    die("Koneksi gagal: " . $e->getMessage());
}
?>
```

File: tambah_mahasiswa.php

```
<?php
include 'koneksi.php';

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $nim = $_POST['nim'];
    $nama = $_POST['nama'];
    $jurusan = $_POST['jurusan'];
    $email = $_POST['email'];

    try {
        $stmt = $conn->prepare("INSERT INTO mahasiswa (nim, nama, jurusan, email) VA
        $stmt->bindParam(':nim', $nim);
        $stmt->bindParam(':nama', $nama);
        $stmt->bindParam(':jurusan', $jurusan);
        $stmt->bindParam(':email', $email);
        $stmt->execute();

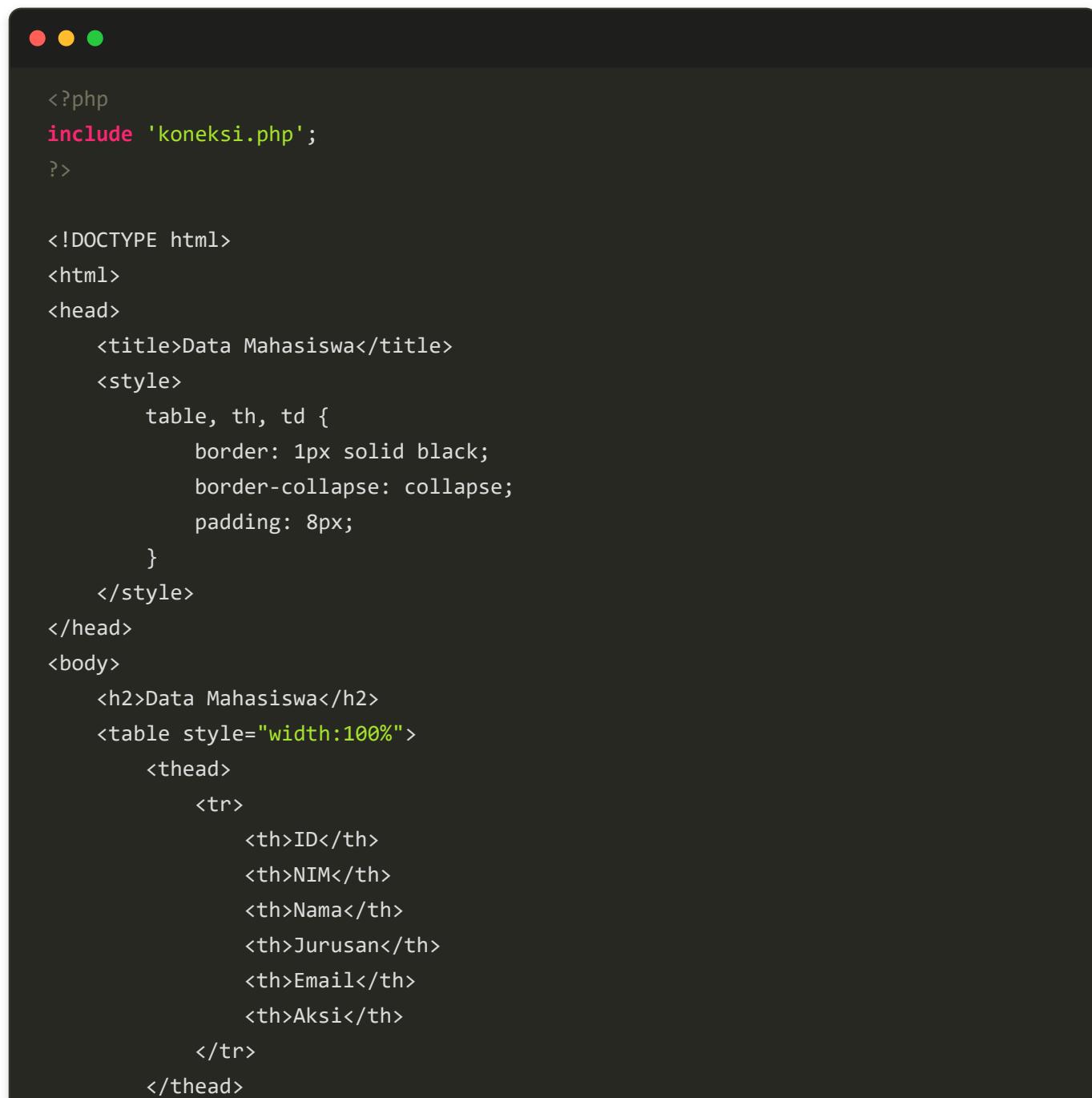
        echo "<p>Data mahasiswa berhasil ditambahkan!</p>";
    } catch(PDOException $e) {
        echo "Error: " . $e->getMessage();
    }
}

?>

<!DOCTYPE html>
<html>
<head>
    <title>Tambah Mahasiswa</title>
</head>
<body>
    <h2>Tambah Data Mahasiswa</h2>
    <form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
        NIM: <input type="text" name="nim" required><br><br>
```

```
Nama: <input type="text" name="nama" required><br><br>
Jurusan: <input type="text" name="jurusan"><br><br>
Email: <input type="email" name="email"><br><br>
<input type="submit" value="Tambah">
</form>
<br>
<a href="lihat_mahasiswa.php">Lihat Data Mahasiswa</a>
</body>
</html>
```

File: lihat_mahasiswa.php



```
<?php
include 'koneksi.php';
?>

<!DOCTYPE html>
<html>
<head>
<title>Data Mahasiswa</title>
<style>
table, th, td {
    border: 1px solid black;
    border-collapse: collapse;
    padding: 8px;
}
</style>
</head>
<body>
<h2>Data Mahasiswa</h2>
<table style="width:100%">
<thead>
<tr>
<th>ID</th>
<th>NIM</th>
<th>Nama</th>
<th>Jurusan</th>
<th>Email</th>
<th>Aksi</th>
</tr>
</thead>
```

```
<tbody>
<?php
try {
    $stmt = $conn->prepare("SELECT id, nim, nama, jurusan, email FROM mahasiswa");
    $stmt->execute();
    $result = $stmt->fetchAll(PDO::FETCH_ASSOC);

    if (count($result) > 0) {
        foreach ($result as $row) {
            echo "<tr>";
            echo "<td>" . $row["id"] . "</td>";
            echo "<td>" . $row["nim"] . "</td>";
            echo "<td>" . $row["nama"] . "</td>";
            echo "<td>" . $row["jurusan"] . "</td>";
            echo "<td>" . $row["email"] . "</td>";
            echo "<td>";
            echo "<a href='edit_mahasiswa.php?id=" . $row["id"] . "'>Edit</a>";
            echo "<a href='hapus_mahasiswa.php?id=" . $row["id"] . "'>Hapus</a>";
            echo "</td>";
            echo "</tr>";
        }
    } else {
        echo "<tr><td colspan='6'>Tidak ada data mahasiswa.</td></tr>";
    }
} catch(PDOException $e) {
    echo "Error: " . $e->getMessage();
}
$conn = null;
?>
</tbody>
</table>
<br>
<a href="tambah_mahasiswa.php">Tambah Mahasiswa Baru</a>
</body>
</html>
```

File: edit_mahasiswa.php

```
<?php
include 'koneksi.php';
```

```
$id = $_GET['id'] ?? '';
$nim = $nama = $jurusan = $email = '';

if ($id) {
    try {
        $stmt = $conn->prepare("SELECT nim, nama, jurusan, email FROM mahasiswa WHERE id = :id");
        $stmt->bindParam(':id', $id);
        $stmt->execute();
        $mahasiswa = $stmt->fetch(PDO::FETCH_ASSOC);

        if ($mahasiswa) {
            $nim = $mahasiswa['nim'];
            $nama = $mahasiswa['nama'];
            $jurusan = $mahasiswa['jurusan'];
            $email = $mahasiswa['email'];
        } else {
            echo "<p>Data mahasiswa tidak ditemukan.</p>";
        }
    } catch(PDOException $e) {
        echo "Error: " . $e->getMessage();
    }
}

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $id = $_POST['id'];
    $nim = $_POST['nim'];
    $nama = $_POST['nama'];
    $jurusan = $_POST['jurusan'];
    $email = $_POST['email'];

    try {
        $stmt = $conn->prepare("UPDATE mahasiswa SET nim = :nim, nama = :nama, jurusan = :jurusan, email = :email WHERE id = :id");
        $stmt->bindParam(':nim', $nim);
        $stmt->bindParam(':nama', $nama);
        $stmt->bindParam(':jurusan', $jurusan);
        $stmt->bindParam(':email', $email);
        $stmt->bindParam(':id', $id);
        $stmt->execute();

        echo "<p>Data mahasiswa berhasil diperbarui!</p>";
        header("Location: lihat_mahasiswa.php"); // Redirect setelah update
        exit();
    } catch(PDOException $e) {
        echo "Error: " . $e->getMessage();
    }
}
```

```

    }
}

?>

<!DOCTYPE html>
<html>
<head>
    <title>Edit Mahasiswa</title>
</head>
<body>
    <h2>Edit Data Mahasiswa</h2>
    <form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
        <input type="hidden" name="id" value="<?php echo $id;?>">
        NIM: <input type="text" name="nim" value="<?php echo $nim;?>" required><br><br>
        Nama: <input type="text" name="nama" value="<?php echo $nama;?>" required><br><br>
        Jurusan: <input type="text" name="jurusan" value="<?php echo $jurusan;?>"><br><br>
        Email: <input type="email" name="email" value="<?php echo $email;?>"><br><br>
        <input type="submit" value="Update">
    </form>
    <br>
    <a href="lihat_mahasiswa.php">Kembali ke Data Mahasiswa</a>
</body>
</html>

```

File: hapus_mahasiswa.php

```

?●?●?●

<?php
include 'koneksi.php';

$id = $_GET['id'] ?? '';

if ($id) {
    try {
        $stmt = $conn->prepare("DELETE FROM mahasiswa WHERE id = :id");
        $stmt->bindParam(':id', $id);
        $stmt->execute();

        echo "<p>Data mahasiswa berhasil dihapus!</p>";
    } catch(PDOException $e) {
        echo "Error: " . $e->getMessage();
    }
}

```

```
}

header("Location: lihat_mahasiswa.php"); // Redirect kembali ke halaman daftar
exit();
?>
```

5. Langkah-langkah Praktikum

1. Pastikan Database dan Tabel Siap:

- Pastikan Anda telah membuat database **db_akademik** dan tabel **mahasiswa** dari Praktikum 3. Jika belum, kembali ke Praktikum 3 dan selesaikan langkah-langkahnya.
- Pastikan modul Apache dan MySQL di XAMPP Control Panel Anda berjalan.

2. Buat File **koneksi.php** :

- Buat file baru bernama **koneksi.php** di folder **htdocs/praktikum_php**.
- Salin dan tempel kode **koneksi.php** dari bagian "Contoh Kode" di atas.

3. Buat File **tambah_mahasiswa.php** :

- Buat file baru bernama **tambah_mahasiswa.php** di folder yang sama.
- Salin dan tempel kode **tambah_mahasiswa.php**.
- Akses melalui browser: **http://localhost/praktikum_php/tambah_mahasiswa.php**.
- Coba tambahkan beberapa data mahasiswa baru.

4. Buat File **lihat_mahasiswa.php** :

- Buat file baru bernama **lihat_mahasiswa.php** di folder yang sama.
- Salin dan tempel kode **lihat_mahasiswa.php**.
- Akses melalui browser: **http://localhost/praktikum_php/lihat_mahasiswa.php**.
- Verifikasi bahwa data mahasiswa yang Anda tambahkan (dan data dari Praktikum 3 jika ada) ditampilkan dalam tabel.

5. Buat File **edit_mahasiswa.php** :

- Buat file baru bernama **edit_mahasiswa.php** di folder yang sama.
- Salin dan tempel kode **edit_mahasiswa.php**.
- Dari halaman **lihat_mahasiswa.php**, klik link **Edit** pada salah satu baris data.
- Ubah beberapa informasi mahasiswa dan klik **Update**.
- Verifikasi perubahan di halaman **lihat_mahasiswa.php**.

6. Buat File **hapus_mahasiswa.php** :

- Buat file baru bernama `hapus_mahasiswa.php` di folder yang sama.
- Salin dan tempel kode `hapus_mahasiswa.php`.
- Dari halaman `lihat_mahasiswa.php`, klik link `Hapus` pada salah satu baris data.
- Konfirmasi penghapusan jika diminta.
- Verifikasi bahwa data mahasiswa telah dihapus dari tabel.

7. Eksplorasi dan Modifikasi:

- Coba tambahkan kolom baru ke tabel `mahasiswa` (misalnya, `tanggal_lahir`) melalui phpMyAdmin.
- Modifikasi semua file CRUD (`tambah_mahasiswa.php`, `lihat_mahasiswa.php`, `edit_mahasiswa.php`) untuk menyertakan kolom baru tersebut.
- Eksplorasi penggunaan `try-catch` block untuk penanganan error database yang lebih baik.

Referensi

- [1] PHP.net. (n.d.). *PDO - PHP Data Objects*. Retrieved from <https://www.php.net/manual/en/book pdo.php>
- [2] W3Schools. (n.d.). *PHP MySQL Connect*. Retrieved from https://www.w3schools.com/php/php_mysql_connect.asp
- [3] W3Schools. (n.d.). *PHP MySQL Insert Data*. Retrieved from https://www.w3schools.com/php/php_mysql_insert.asp
- [4] W3Schools. (n.d.). *PHP MySQL Select Data*. Retrieved from https://www.w3schools.com/php/php_mysql_select.asp
- [5] W3Schools. (n.d.). *PHP MySQL Update Data*. Retrieved from https://www.w3schools.com/php/php_mysql_update.asp
- [6] W3Schools. (n.d.). *PHP MySQL Delete Data*. Retrieved from https://www.w3schools.com/php/php_mysql_delete.asp

Praktikum 6: Session dan Cookie Management

1. Tujuan Pembelajaran

Setelah menyelesaikan praktikum ini, mahasiswa diharapkan mampu:

- Memahami konsep dan perbedaan antara session dan cookie.
- Menggunakan session untuk menyimpan data pengguna antar halaman.
- Menggunakan cookie untuk menyimpan data pengguna di sisi klien.
- Mengimplementasikan otentifikasi sederhana menggunakan session.

2. Materi Pembelajaran

2.1. Konsep Session dan Cookie

HTTP adalah protokol stateless, artinya server tidak "mengingat" apa pun tentang permintaan sebelumnya dari klien yang sama. Untuk mengatasi keterbatasan ini dan mempertahankan status pengguna antar permintaan, kita menggunakan session dan cookie.

- **Cookie:**

- Data disimpan di sisi klien (browser pengguna).
- Ukuran terbatas (sekitar 4KB per cookie).
- Dapat memiliki tanggal kedaluwarsa.
- Tidak aman untuk menyimpan informasi sensitif karena dapat diakses dan dimodifikasi oleh pengguna.
- Digunakan untuk "mengingat" preferensi pengguna, keranjang belanja, atau melacak aktivitas.

- **Session:**

- Data disimpan di sisi server.
- Setiap session memiliki ID unik yang biasanya disimpan dalam cookie di sisi klien.
- Lebih aman untuk menyimpan informasi sensitif karena data tidak langsung diakses oleh klien.
- Data session akan hilang ketika browser ditutup atau setelah waktu tertentu (timeout).
- Digunakan untuk otentifikasi pengguna, data login, atau informasi sementara lainnya.

2.2. Mengelola Cookie di PHP

- **Membuat Cookie:** Menggunakan fungsi `setcookie()`.



```
setcookie(name, value, expire, path, domain, secure, httponly);
```

- `name` : Nama cookie.
- `value` : Nilai cookie.
- `expire` : Waktu kedaluwarsa (timestamp Unix). `time() + (86400 * 30)` untuk 30 hari.
- `path` : Jalur server tempat cookie akan tersedia (misalnya `/` untuk seluruh domain).
- `domain` : Domain tempat cookie tersedia.
- `secure` : `true` jika cookie hanya dikirim melalui koneksi HTTPS.
- `httponly` : `true` jika cookie hanya dapat diakses melalui HTTP (tidak melalui JavaScript).

Contoh:

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>
```

- **Mengambil Nilai Cookie:** Menggunakan superglobal `$_COOKIE`.

```
<?php
if(isset($_COOKIE["user"])) {
    echo "Nilai cookie user: " . $_COOKIE["user"];
} else {
    echo "Cookie user belum diatur.";
}
?>
```

- **Menghapus Cookie:** Mengatur waktu kedaluwarsa ke masa lalu.

```
<?php
setcookie("user", "", time() - 3600); // Set waktu kedaluwarsa ke satu jam yang
echo "Cookie user telah dihapus.";
?>
```

2.3. Mengelola Session di PHP

- **Memulai Session:** Menggunakan fungsi `session_start()`. Harus dipanggil di awal setiap skrip yang menggunakan session.

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

- **Menyimpan Data Session:** Menggunakan superglobal `$_SESSION`.

```
<?php  
session_start();  
$_SESSION["username"] = "admin";  
$_SESSION["favcolor"] = "green";  
echo "Data session telah diatur.";  
?>
```

- **Mengambil Data Session:** Menggunakan superglobal `$_SESSION`.

```
<?php  
session_start();  
if(isset($_SESSION["username"])) {  
    echo "Username: " . $_SESSION["username"] . "<br>";  
    echo "Warna favorit: " . $_SESSION["favcolor"];  
} else {  
    echo "Tidak ada data session.";  
}  
?>
```

- **Menghapus Data Session:**

- `unset()`: Menghapus satu variabel session.

```
<?php  
session_start();  
unset($_SESSION["favcolor"]);  
echo "Variabel favcolor telah dihapus.";  
?>
```

- `session_destroy()` : Menghapus semua data session dan mengakhiri session.

```
<?php  
session_start();  
session_destroy();  
echo "Semua data session telah dihapus.";  
?>
```

3. Pembahasan

Praktikum ini akan mengajarkan Anda bagaimana mempertahankan status pengguna di aplikasi web Anda. Anda akan belajar kapan harus menggunakan cookie (untuk preferensi non-sensitif) dan kapan harus menggunakan session (untuk data sensitif seperti status login). Memahami perbedaan dan cara kerja keduanya sangat penting untuk membangun aplikasi web yang fungsional dan aman.

Inginlah bahwa `session_start()` harus dipanggil di awal setiap halaman PHP yang ingin menggunakan session. Jika tidak, PHP tidak akan dapat mengakses atau membuat session. Untuk cookie, `setcookie()` harus dipanggil sebelum ada output HTML atau teks lain ke browser, karena cookie dikirim sebagai bagian dari header HTTP.

4. Contoh Kode

Kita akan membuat contoh sederhana sistem login/logout menggunakan session.

File: `login.php`

```
<?php  
session_start();  
  
$username = "";
```

```
$password = "";
$error = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST["username"];
    $password = $_POST["password"];

    // Contoh validasi sederhana (dalam aplikasi nyata, gunakan database)
    if ($username == "admin" && $password == "password123") {
        $_SESSION["loggedin"] = true;
        $_SESSION["username"] = $username;
        header("Location: dashboard.php");
        exit;
    } else {
        $error = "Username atau password salah!";
    }
}

// Jika sudah login, redirect ke dashboard
if (isset($_SESSION["loggedin"]) && $_SESSION["loggedin"] === true) {
    header("Location: dashboard.php");
    exit;
}
?>

<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
    <style>
        .error {color: red;}
    </style>
</head>
<body>
    <h2>Login Pengguna</h2>
    <form method="post" action="php echo htmlspecialchars($_SERVER["PHP_SELF"]);?&gt;
        Username: &lt;input type="text" name="username" required&gt;&lt;br&gt;&lt;br&gt;
        Password: &lt;input type="password" name="password" required&gt;&lt;br&gt;&lt;br&gt;
        &lt;span class="error"&gt;&lt;?php echo $error;?&gt;&lt;/span&gt;&lt;br&gt;&lt;br&gt;
        &lt;input type="submit" value="Login"&gt;
    &lt;/form&gt;
&lt;/body&gt;
&lt;/html&gt;</pre
```

File: dashboard.php

```
<?php
session_start();

// Cek apakah pengguna sudah login
if (!isset($_SESSION["loggedin"]) || $_SESSION["loggedin"] !== true) {
    header("Location: login.php");
    exit;
}

?>

<!DOCTYPE html>
<html>
<head>
    <title>Dashboard</title>
</head>
<body>
    <h2>Selamat Datang, <?php echo htmlspecialchars($_SESSION["username"]); ?>!</h2>
    <p>Ini adalah halaman dashboard Anda.</p>
    <p><a href="logout.php">Logout</a></p>

    <h3>Contoh Penggunaan Cookie</h3>
    <?php
        $cookie_name = "last_visit";
        $current_time = date("Y-m-d H:i:s");

        if(isset($_COOKIE[$cookie_name])) {
            echo "<p>Kunjungan terakhir Anda: " . $_COOKIE[$cookie_name] . "</p>";
        } else {
            echo "<p>Ini adalah kunjungan pertama Anda.</p>";
        }

        // Set cookie untuk kunjungan saat ini, berlaku 1 jam
        setcookie($cookie_name, $current_time, time() + 3600, "/");
    ?>
</body>
</html>
```

File: logout.php

```
<?php
session_start();

// Hapus semua variabel session
$_SESSION = array();

// Hapus cookie session
if (ini_get("session.use_cookies")) {
    $params = session_get_cookie_params();
    setcookie(session_name(),
        '',
        time() - 42000,
        $params[ "path" ],
        $params[ "domain" ],
        $params[ "secure" ],
        $params[ "httponly" ]
    );
}

// Hancurkan session
session_destroy();

header("Location: login.php");
exit;
?>
```

5. Langkah-langkah Praktikum

1. Buat File `login.php` :

- Buat file baru bernama `login.php` di folder `htdocs/praktikum_php`.
- Salin dan tempel kode `login.php` dari bagian "Contoh Kode" di atas.

2. Buat File `dashboard.php` :

- Buat file baru bernama `dashboard.php` di folder yang sama.
- Salin dan tempel kode `dashboard.php`.

3. Buat File `logout.php` :

- Buat file baru bernama `logout.php` di folder yang sama.

- Salin dan tempel kode `logout.php`.

4. Uji Sistem Login:

- Akses `http://localhost/praktikum_php/login.php` di browser Anda.
- Coba login dengan username `admin` dan password `password123`.
- Jika berhasil, Anda akan diarahkan ke `dashboard.php`.
- Coba akses `dashboard.php` langsung tanpa login terlebih dahulu. Anda seharusnya diarahkan kembali ke `login.php`.

5. Uji Logout:

- Dari `dashboard.php`, klik link `Logout`.
- Anda seharusnya diarahkan kembali ke `login.php`.
- Coba akses `dashboard.php` lagi untuk memastikan Anda sudah logout.

6. Uji Cookie:

- Perhatikan pesan "Kunjungan terakhir Anda:" di `dashboard.php`.
- Tutup browser Anda sepenuhnya, lalu buka kembali dan akses `dashboard.php` (setelah login).
- Perhatikan apakah waktu kunjungan terakhir telah diperbarui.
- Anda juga bisa memeriksa cookie di pengaturan browser Anda untuk melihat cookie `last_visit`.

7. Eksplorasi dan Modifikasi:

- Coba tambahkan fitur "Remember Me" menggunakan cookie untuk menyimpan username pengguna (bukan password!).
- Implementasikan penghitung kunjungan halaman menggunakan session.
- Modifikasi sistem login agar username dan password diambil dari database yang telah Anda buat di Praktikum 3 dan 5.

Referensi

- [1] W3Schools. (n.d.). *PHP Cookies*. Retrieved from https://www.w3schools.com/php/php_cookies.asp
- [2] W3Schools. (n.d.). *PHP Sessions*. Retrieved from https://www.w3schools.com/php/php_sessions.asp

Praktikum 7: Konsep Dasar OOP di PHP

1. Tujuan Pembelajaran

Setelah menyelesaikan praktikum ini, mahasiswa diharapkan mampu:

- Memahami konsep dasar Object-Oriented Programming (OOP).
- Mendefinisikan dan menggunakan kelas (class) dan objek (object) di PHP.
- Mengelola properti (properties) dan metode (methods) dalam sebuah kelas.
- Memahami dan mengimplementasikan konstruktor dan destruktur.
- Menerapkan konsep enkapsulasi (public, private, protected).
- Memahami dasar-dasar pewarisan (inheritance).

2. Materi Pembelajaran

2.1. Pengantar Object-Oriented Programming (OOP)

Object-Oriented Programming (OOP) adalah paradigma pemrograman yang didasarkan pada konsep "objek", yang dapat berisi data (dalam bentuk field atau properti) dan kode (dalam bentuk prosedur atau metode).

Tujuan utama OOP adalah untuk meningkatkan fleksibilitas dan pemeliharaan program dengan memungkinkan modularitas. [1]

Konsep Utama OOP:

- **Class (Kelas):** Blueprint atau cetak biru untuk membuat objek. Ini mendefinisikan properti (data) dan metode (fungsi) yang akan dimiliki oleh objek yang dibuat dari kelas tersebut.
- **Object (Objek):** Instance dari sebuah kelas. Setiap objek memiliki properti dan metode yang didefinisikan oleh kelasnya.
- **Property (Properti):** Variabel yang terkait dengan sebuah objek. Mereka mendefinisikan karakteristik atau atribut objek.
- **Method (Metode):** Fungsi yang terkait dengan sebuah objek. Mereka mendefinisikan perilaku atau tindakan yang dapat dilakukan objek.

2.2. Mendefinisikan Kelas dan Membuat Objek

Kelas didefinisikan menggunakan kata kunci `class`. Objek dibuat menggunakan kata kunci `new`.

Contoh:



```
<?php  
class Mobil {
```

```
// Properti
public $merk;
public $warna;

// Metode
public function __construct($merk, $warna) {
    $this->merk = $merk;
    $this->warna = $warna;
}

public function getInfo() {
    return "Mobil ini bermerk " . $this->merk . " dan berwarna " . $this->warna
}
}

// Membuat objek (instance dari kelas Mobil)
$mobill = new Mobil("Toyota", "Merah");
$mobil2 = new Mobil("Honda", "Biru");

// Mengakses properti dan metode
echo $mobill->getInfo() . "<br>"; // Output: Mobil ini bermerk Toyota dan berwarna Merah
echo $mobil2->getInfo() . "<br>"; // Output: Mobil ini bermerk Honda dan berwarna Biru
?>
```

2.3. Konstruktor dan Destruktor

- **Konstruktor** (`__construct()`): Metode khusus yang secara otomatis dipanggil saat objek baru dibuat. Digunakan untuk menginisialisasi properti objek.
- **Destruktor** (`__destruct()`): Metode khusus yang secara otomatis dipanggil saat objek dihancurkan atau skrip berakhir. Digunakan untuk membersihkan sumber daya (misalnya, menutup koneksi database).

Contoh:

```
<?php
class Produk {
    public $nama;
    public $harga;

    public function __construct($nama, $harga) {
        $this->nama = $nama;
```

```

    $this->harga = $harga;
    echo "Objek Produk '" . $this->nama . "' dibuat.<br>";
}

public function __destruct() {
    echo "Objek Produk '" . $this->nama . "' dihancurkan.<br>";
}

public function getDetail() {
    return "Produk: " . $this->nama . ", Harga: " . $this->harga;
}
}

$laptop = new Produk("Laptop Gaming", 15000000);
echo $laptop->getDetail() . "<br>";

// Objek akan dihancurkan secara otomatis saat skrip berakhir
?>

```

2.4. Enkapsulasi (Visibility)

Enkapsulasi adalah prinsip OOP yang membungkus data (properti) dan metode yang beroperasi pada data tersebut ke dalam satu unit (kelas). Ini mengontrol akses ke properti dan metode dari luar kelas.

- **public** : Properti atau metode dapat diakses dari mana saja (dari dalam kelas, dari objek, atau dari kelas turunan).
- **private** : Properti atau metode hanya dapat diakses dari dalam kelas itu sendiri.
- **protected** : Properti atau metode dapat diakses dari dalam kelas itu sendiri dan dari kelas-kelas yang mewarisinya (kelas turunan).

Contoh:



```

<?php
class AkunBank {
    private $saldo; // Properti private
    public $pemilik;

    public function __construct($pemilik, $saldoAwal) {
        $this->pemilik = $pemilik;
        $this->saldo = $saldoAwal;
    }
}

```

```

public function getSaldo() {
    return $this->saldo;
}

public function setor($jumlah) {
    if ($jumlah > 0) {
        $this->saldo += $jumlah;
        echo "Setor " . $jumlah . " berhasil. Saldo sekarang: " . $this->saldo .
    } else {
        echo "Jumlah setor harus positif.<br>";
    }
}

protected function cekInternal() {
    echo "Melakukan cek internal akun.<br>";
}
}

class AkunPremium extends AkunBank {
    public function cekSaldoPremium() {
        $this->cekInternal(); // Bisa mengakses metode protected
        echo "Saldo akun premium: " . $this->getSaldo() . "<br>";
    }
}

$akun1 = new AkunBank("Yamin", 1000000);
echo "Pemilik: " . $akun1->pemilik . "<br>";
echo "Saldo: " . $akun1->getSaldo() . "<br>";
$akun1->setor(500000);
// echo $akun1->saldo; // Ini akan error karena saldo adalah private

$akunPremium = new AkunPremium("Siti", 5000000);
$akunPremium->cekSaldoPremium();
?>

```

2.5. Pewarisan (Inheritance)

Pewarisan memungkinkan sebuah kelas (kelas anak/subclass) untuk mewarisi properti dan metode dari kelas lain (kelas induk/superclass). Ini mempromosikan penggunaan kembali kode dan membentuk hubungan "is-a" (misalnya, "Mobil adalah Kendaraan"). Kata kunci `extends` digunakan untuk pewarisan.

Contoh:

```
<?php

class Kendaraan {
    public $roda;
    public $kecepatan;

    public function __construct($roda, $kecepatan) {
        $this->roda = $roda;
        $this->kecepatan = $kecepatan;
    }

    public function bergerak() {
        return "Kendaraan bergerak dengan kecepatan " . $this->kecepatan . " km/jam.";
    }
}

class Mobil extends Kendaraan {
    public $merk;

    public function __construct($roda, $kecepatan, $merk) {
        parent::__construct($roda, $kecepatan); // Memanggil konstruktor kelas induk
        $this->merk = $merk;
    }

    public function klakson() {
        return "Beep beep!";
    }

    // Override metode dari kelas induk
    public function bergerak() {
        return "Mobil " . $this->merk . " bergerak dengan kecepatan " . $this->kecepatan;
    }
}

$mobilSaya = new Mobil(4, 120, "Toyota");
echo $mobilSaya->bergerak() . "<br>";
echo $mobilSaya->klakson() . "<br>";
?>
```

3. Pembahasan

Praktikum ini memperkenalkan Anda pada fondasi Object-Oriented Programming di PHP. Memahami konsep kelas, objek, properti, dan metode adalah langkah pertama yang krusial. Enkapsulasi, melalui penggunaan `public`, `private`, dan `protected`, membantu Anda mengontrol akses ke bagian-bagian internal objek, yang penting untuk menjaga integritas data dan membuat kode lebih mudah dikelola.

Pewarisan adalah alat yang ampuh untuk penggunaan kembali kode dan membangun hierarki objek yang logis. Anda akan melihat bagaimana kelas anak dapat memperluas fungsionalitas kelas induk dan bahkan mengganti (override) metode yang ada. Latihan ini akan membantu Anda berpikir dalam paradigma objek, yang akan sangat berguna saat Anda beralih ke framework seperti CodeIgniter yang sangat bergantung pada OOP.

4. Contoh Kode

File: oop_dasar.php

```
<!DOCTYPE html>
<html>
<head>
    <title>OOP Dasar PHP</title>
</head>
<body>

    <h1>Contoh OOP Dasar</h1>

    <?php
        // Definisi Kelas Karyawan
        class Karyawan {
            // Properti
            public $nama;
            public $jabatan;
            private $gajiPokok; // Private: hanya bisa diakses dari dalam kelas
            protected $tunjangan; // Protected: bisa diakses dari kelas turunan

            // Konstruktor
            public function __construct($nama, $jabatan, $gajiPokok) {
                $this->nama = $nama;
                $this->jabatan = $jabatan;
                $this->gajiPokok = $gajiPokok;
                $this->tunjangan = 0; // Inisialisasi tunjangan
                echo "<p>Objek Karyawan '" . $this->nama . "' dibuat.</p>";
            }
        }
    </?php>
```

```
// Metode Publik
public function getInfo() {
    return "Nama: " . $this->nama . ", Jabatan: " . $this->jabatan;
}

public function hitungGajiTotal() {
    // Metode publik bisa mengakses properti private
    return $this->gajiPokok + $this->tunjangan;
}

// Metode untuk mengatur tunjangan (akses dari luar)
public function setTunjangan($tunjangan) {
    if ($tunjangan >= 0) {
        $this->tunjangan = $tunjangan;
    }
}

// Destruktor
public function __destruct() {
    echo "<p>Objek Karyawan '" . $this->nama . "' dihancurkan.</p>";
}
}

// Definisi Kelas Manajer (turunan dari Karyawan)
class Manajer extends Karyawan {
    public $departemen;

    public function __construct($nama, $jabatan, $gajiPokok, $departemen) {
        parent::__construct($nama, $jabatan, $gajiPokok); // Panggil konstruktor
        $this->departemen = $departemen;
        // Manajer mendapatkan tunjangan lebih
        $this->tunjangan = 1000000; // Bisa mengakses properti protected
    }
}

// Override metode getInfo dari kelas induk
public function getInfo() {
    return parent::getInfo() . ", Departemen: " . $this->departemen;
}

public function rapat() {
    return "Manajer " . $this->nama . " sedang memimpin rapat di departemen
}
}
```

```
// Membuat Objek Karyawan
$karyawan1 = new Karyawan("Yamin bae", "Staff IT", 5000000);
$karyawan1->setTunjangan(200000);
echo $karyawan1->getInfo() . "<br>";
echo "Gaji Total Karyawan 1: " . $karyawan1->hitungGajiTotal() . "<br>";
// echo $karyawan1->gajiPokok; // Ini akan error karena gajiPokok private

echo "<hr>";

// Membuat Objek Manajer
$manajer1 = new Manajer("Siti Aminah", "Manajer Pemasaran", 8000000, "Pemasaran");
echo $manajer1->getInfo() . "<br>";
echo "Gaji Total Manajer 1: " . $manajer1->hitungGajiTotal() . "<br>";
echo $manajer1->rapat() . "<br>";

echo "<hr>";

// Contoh lain tanpa konstruktor
class Mahasiswa {
    public $nama;
    public $nim;

    public function setNama($nama) {
        $this->nama = $nama;
    }

    public function setNim($nim) {
        $this->nim = $nim;
    }

    public function getNama() {
        return $this->nama;
    }

    public function getNim() {
        return $this->nim;
    }
}

$mhs1 = new Mahasiswa();
$mhs1->setNama("Dewi Lestari");
$mhs1->setNim("2023007");
echo "<p>Mahasiswa: " . $mhs1->getNama() . " (NIM: " . $mhs1->getNim() . ")</p>"
```

```
?>

</body>
</html>
```

5. Langkah-langkah Praktikum

1. Buat File `oop_dasar.php` :

- Buka editor teks Anda.
- Salin dan tempel kode dari bagian "Contoh Kode" di atas ke dalam file baru.
- Simpan file tersebut dengan nama `oop_dasar.php` di dalam folder `htdocs/praktikum_php`.

2. Akses Melalui Browser:

- Buka browser web Anda dan ketik `http://localhost/praktikum_php/oop_dasar.php`.
- Amati output yang ditampilkan. Perhatikan pesan konstruktur dan destruktur, serta informasi karyawan dan manajer.

3. Eksplorasi Properti dan Metode:

- Coba ubah nilai properti `public` secara langsung (misalnya `$karyawan1->jabatan = "Senior Staff";`). Simpan dan refresh browser.
- Coba akses properti `private` (`$karyawan1->gajiPokok`) atau `protected` (`$karyawan1->tunjangan`) secara langsung dari luar kelas (misalnya di bawah baris `echo`

Gaji Total Karyawan 1: " . \$karyawan1->hitungGajiTotal() . "

";). Amati error yang muncul. * Pahami mengapa metode `setTunjangan` diperlukan untuk mengubah tunjangan`.

4. Eksplorasi Pewarisan:

- Perhatikan bagaimana kelas `Manajer` mewarisi properti dan metode dari `Karyawan`.
- Amati bagaimana metode `getInfo()` di kelas `Manajer` meng-override metode dari kelas `Karyawan`.
- Coba buat kelas baru yang mewarisi dari `Manajer` (misalnya `Direktur`) dan tambahkan properti atau metode spesifik untuk `Direktur`.

5. Konstruktur dan Destruktor:

- Perhatikan urutan pemanggilan konstruktur dan destruktur. Konstruktur dipanggil saat objek dibuat, destruktur saat objek dihancurkan (biasanya di akhir skrip).
- Coba buat lebih banyak objek `Karyawan` atau `Manajer` dan amati bagaimana pesan konstruktur/destruktur muncul.

Referensi

- [1] W3Schools. (n.d.). *PHP OOP*. Retrieved from https://www.w3schools.com/php/php_oop_intro.asp
- [2] PHP.net. (n.d.). *Classes and Objects*. Retrieved from <https://www.php.net/manual/en/language.oop5.basic.php>

Praktikum 8: Pengenalan Konsep MVC

1. Tujuan Pembelajaran

Setelah menyelesaikan praktikum ini, mahasiswa diharapkan mampu:

- Memahami arsitektur Model-View-Controller (MVC) sebagai pola desain perangkat lunak.
- Mengenali peran dan tanggung jawab masing-masing komponen (Model, View, Controller).
- Menerapkan konsep MVC dalam proyek PHP sederhana tanpa menggunakan framework.
- Meningkatkan modularitas dan keterbacaan kode dengan memisahkan logika aplikasi.

2. Materi Pembelajaran

2.1. Apa itu MVC?

Model-View-Controller (MVC) adalah pola arsitektur perangkat lunak yang memisahkan aplikasi menjadi tiga komponen utama yang saling terhubung. Tujuannya adalah untuk memisahkan representasi informasi dari interaksi pengguna, sehingga memungkinkan pengembangan yang lebih efisien dan pemeliharaan yang lebih mudah. [1]

Komponen-komponen MVC:

- **Model:** Merepresentasikan data dan logika bisnis aplikasi. Model bertanggung jawab untuk mengelola data, berinteraksi dengan database (CRUD), dan menerapkan aturan bisnis. Model tidak memiliki pengetahuan tentang antarmuka pengguna (View) atau bagaimana data ditampilkan.
- **View:** Bertanggung jawab untuk menampilkan data kepada pengguna. View adalah antarmuka pengguna (UI) dari aplikasi. Ini mengambil data dari Model dan menampilkannya dalam format yang sesuai (misalnya, HTML). View tidak memiliki logika bisnis atau pengetahuan tentang bagaimana data diambil atau disimpan.
- **Controller:** Bertindak sebagai perantara antara Model dan View. Controller menerima input dari pengguna (melalui View), memprosesnya (seringkali dengan berinteraksi dengan Model), dan kemudian memilih View yang tepat untuk menampilkan hasilnya. Controller mengelola alur aplikasi.

2.2. Alur Kerja MVC

1. **Pengguna berinteraksi dengan View:** Misalnya, mengklik tombol atau mengisi form.

2. **View mengirimkan permintaan ke Controller:** Controller menerima input pengguna.

3. **Controller memproses permintaan:**

- Jika permintaan memerlukan data atau logika bisnis, Controller berinteraksi dengan Model.
- Model mengambil atau memanipulasi data dan mengembalikan hasilnya ke Controller.

4. **Controller memilih View yang sesuai:** Berdasarkan hasil dari Model atau logika internal, Controller menentukan View mana yang akan digunakan untuk menampilkan respons.

5. **View menampilkan data:** View menerima data dari Controller (yang mungkin berasal dari Model) dan menampilkannya kepada pengguna.



Gambar 8.1: Alur Kerja Model-View-Controller (Sumber: Wikimedia Commons)

2.3. Keuntungan Menggunakan MVC

- **Pemisahan Kekhawatiran (Separation of Concerns):** Setiap komponen memiliki tanggung jawab yang jelas, membuat kode lebih terorganisir dan mudah dipahami.
- **Modularitas:** Perubahan pada satu komponen tidak terlalu memengaruhi komponen lain.
- **Penggunaan Kembali Kode:** Logika bisnis di Model dapat digunakan kembali di berbagai View atau Controller.
- **Pengembangan Paralel:** Tim yang berbeda dapat bekerja pada Model, View, dan Controller secara bersamaan.
- **Kemudahan Pengujian:** Setiap komponen dapat diuji secara independen.

3. Pembahasan

Pada praktikum ini, Anda akan membangun aplikasi PHP sederhana yang mengimplementasikan pola MVC secara manual. Ini akan membantu Anda memahami bagaimana setiap bagian (Model, View, Controller) bekerja sama tanpa bantuan framework yang mungkin menyembunyikan detail implementasi. Meskipun ini adalah implementasi yang sangat dasar, prinsip-prinsip yang Anda pelajari akan menjadi fondasi yang kuat untuk memahami framework MVC yang lebih kompleks seperti CodeIgniter.

Kita akan membuat aplikasi daftar tugas (To-Do List) sederhana. Model akan mengelola data tugas, View akan menampilkan daftar tugas dan form, dan Controller akan menangani permintaan pengguna (menambah, menghapus, menampilkan tugas).

4. Contoh Kode

Kita akan membuat struktur folder sebagai berikut di dalam `htdocs/praktikum_php/mvc_app` :

```
index.php
controllers/
|   TaskController.php
models/
|   TaskModel.php
views/
|   task_list.php
|   task_form.php
```

File: mvc_app/index.php (Front Controller)

```
<?php
// index.php adalah front controller yang mengarahkan semua permintaan

require_once __DIR__ . "/controllers/TaskController.php";
require_once __DIR__ . "/models/TaskModel.php";

// Inisialisasi Controller
$controller = new TaskController();

// Routing sederhana
$action = $_GET["action"] ?? "list"; // Default action adalah 'list'

switch ($action) {
    case "list":
        $controller->index();
        break;
    case "add":
        $controller->add();
        break;
    case "delete":
        $controller->delete();
        break;
    default:
        echo "Halaman tidak ditemukan.";
        break;
}
?>
```

File: mvc_app/models/TaskModel.php

```
<?php
class TaskModel {
    private $tasks = [];
    private $filePath = __DIR__ . "/../data/tasks.json";

    public function __construct() {
```

```
// Pastikan direktori data ada
if (!is_dir(__DIR__ . "/../data")) {
    mkdir(__DIR__ . "/../data", 0777, true);
}

// Muat tugas dari file saat model diinisialisasi
if (file_exists($this->filePath)) {
    $json = file_get_contents($this->filePath);
    $this->tasks = json_decode($json, true) ?? [];
}

public function getAllTasks() {
    return $this->tasks;
}

public function addTask($description) {
    $newTask = [
        "id" => uniqid(),
        "description" => $description,
        "completed" => false
    ];
    $this->tasks[] = $newTask;
    $this->saveTasks();
    return true;
}

public function deleteTask($id) {
    $initialCount = count($this->tasks);
    $this->tasks = array_filter($this->tasks, function($task) use ($id) {
        return $task["id"] != $id;
    });
    $this->saveTasks();
    return count($this->tasks) < $initialCount;
}

private function saveTasks() {
    file_put_contents($this->filePath, json_encode($this->tasks, JSON_PRETTY_PRINT));
}

?>
```

File: mvc_app/controllers/TaskController.php

```
<?php
require_once __DIR__ . "/../models/TaskModel.php";

class TaskController {
    private $taskModel;

    public function __construct() {
        $this->taskModel = new TaskModel();
    }

    public function index() {
        $tasks = $this->taskModel->getAllTasks();
        // Memuat view untuk menampilkan daftar tugas
        include __DIR__ . "/../views/task_list.php";
    }

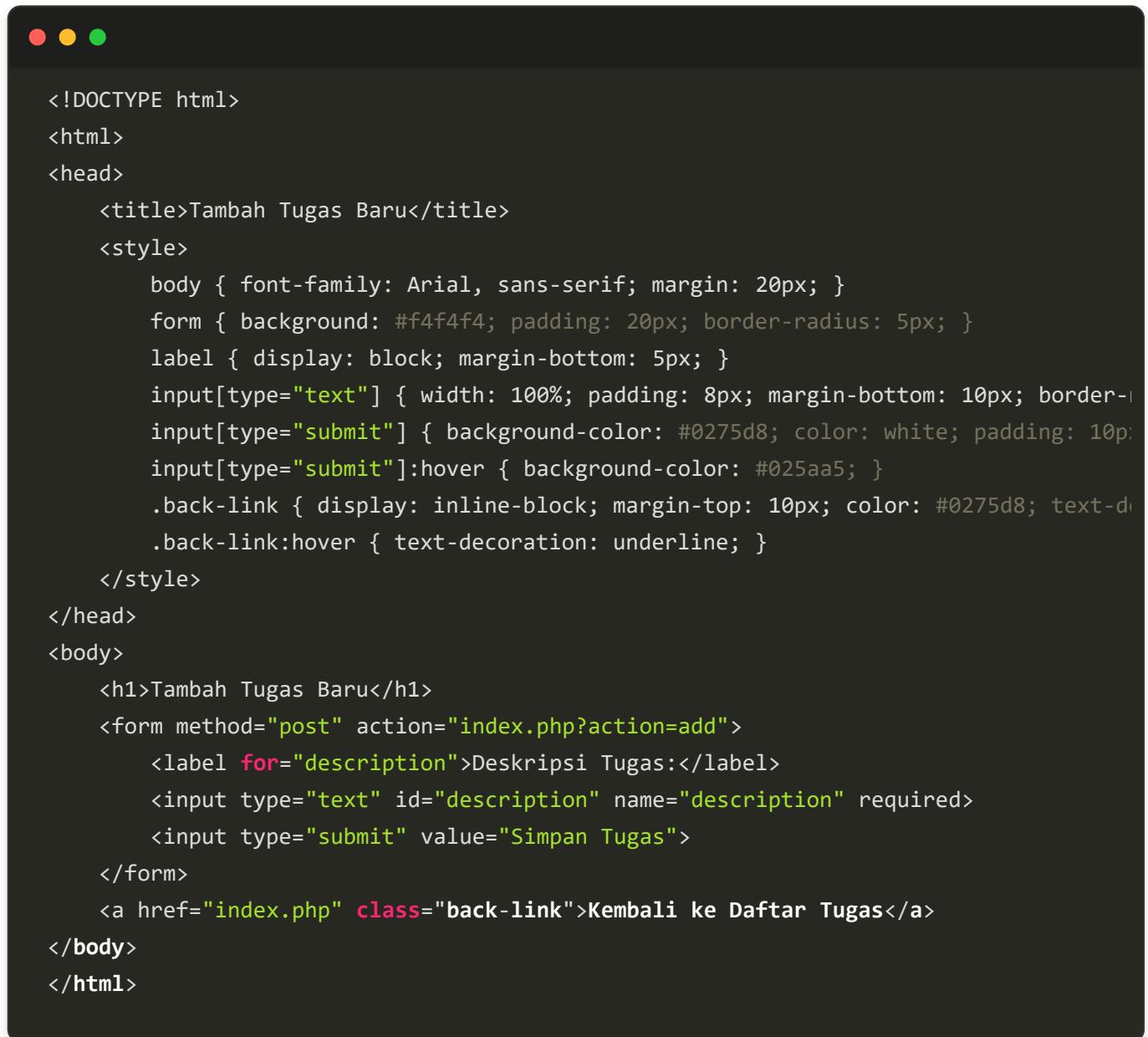
    public function add() {
        if ($_SERVER["REQUEST_METHOD"] == "POST") {
            $description = $_POST["description"] ?? "";
            if (!empty($description)) {
                $this->taskModel->addTask($description);
                header("Location: index.php"); // Redirect kembali ke daftar tugas
                exit;
            } else {
                echo "Deskripsi tugas tidak boleh kosong.";
            }
        }
        // Memuat view untuk form penambahan tugas
        include __DIR__ . "/../views/task_form.php";
    }

    public function delete() {
        $id = $_GET["id"] ?? "";
        if (!empty($id)) {
            $this->taskModel->deleteTask($id);
        }
        header("Location: index.php"); // Redirect kembali ke daftar tugas
        exit;
    }
}
?>
```

File: mvc_app/views/task_list.php

```
<!DOCTYPE html>
<html>
<head>
    <title>Daftar Tugas</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 20px; }
        ul { list-style-type: none; padding: 0; }
        li { background: #f4f4f4; margin-bottom: 5px; padding: 10px; border-radius: 5px; }
        .completed { text-decoration: line-through; color: #888; }
        a { text-decoration: none; color: #d9534f; }
        a:hover { text-decoration: underline; }
        .add-button { display: inline-block; margin-top: 20px; padding: 10px 15px; border: 2px solid #4cae4c; }
        .add-button:hover { background-color: #4cae4c; color: white; }
    </style>
</head>
<body>
    <h1>Daftar Tugas</h1>
    <ul>
        <?php if (!empty($tasks)): ?>
            <?php foreach ($tasks as $task): ?>
                <li>
                    <span class="<?php echo $task["completed"] ? "completed" : "";">
                        <?php echo htmlspecialchars($task["description"]); ?>
                    </span>
                    <a href="index.php?action=delete&id=<?php echo $task["id"]; ?>">
                </li>
            <?php endforeach; ?>
        <?php else: ?>
            <li>Tidak ada tugas.</li>
        <?php endif; ?>
    </ul>
    <a href="index.php?action=add" class="add-button">Tambah Tugas Baru</a>
</body>
</html>
```

File: mvc_app/views/task_form.php



```

<!DOCTYPE html>
<html>
<head>
    <title>Tambah Tugas Baru</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 20px; }
        form { background: #f4f4f4; padding: 20px; border-radius: 5px; }
        label { display: block; margin-bottom: 5px; }
        input[type="text"] { width: 100%; padding: 8px; margin-bottom: 10px; border: 1px solid #ccc; }
        input[type="submit"] { background-color: #0275d8; color: white; padding: 10px; border: none; }
        input[type="submit"]:hover { background-color: #025aa5; }
        .back-link { display: inline-block; margin-top: 10px; color: #0275d8; text-decoration: none; }
        .back-link:hover { text-decoration: underline; }
    </style>
</head>
<body>
    <h1>Tambah Tugas Baru</h1>
    <form method="post" action="index.php?action=add">
        <label for="description">Deskripsi Tugas:</label>
        <input type="text" id="description" name="description" required>
        <input type="submit" value="Simpan Tugas">
    </form>
    <a href="index.php" class="back-link">Kembali ke Daftar Tugas</a>
</body>
</html>

```

5. Langkah-langkah Praktikum

1. Buat Struktur Folder:

- Di dalam folder `htdocs/praktikum_php`, buat folder baru bernama `mvc_app`.
- Di dalam `mvc_app`, buat folder `controllers`, `models`, `views`, dan `data`.

2. Buat File `index.php`:

- Buat file `index.php` di dalam folder `mvc_app`.
- Salin dan tempel kode `index.php` dari bagian "Contoh Kode" di atas.

3. Buat File Model:

- Buat file `TaskModel.php` di dalam folder `mvc_app/models`.
- Salin dan tempel kode `TaskModel.php`.

4. Buat File Controller:

- Buat file `TaskController.php` di dalam folder `mvc_app/controllers`.
- Salin dan tempel kode `TaskController.php`.

5. Buat File View:

- Buat file `task_list.php` di dalam folder `mvc_app/views`.
- Salin dan tempel kode `task_list.php`.
- Buat file `task_form.php` di dalam folder `mvc_app/views`.
- Salin dan tempel kode `task_form.php`.

6. Akses Aplikasi:

- Buka browser web Anda dan ketik `http://localhost/praktikum_php/mvc_app/`.
- Anda akan melihat daftar tugas (kosong jika belum ada).

7. Uji Penambahan Tugas:

- Klik tombol "Tambah Tugas Baru".
- Masukkan deskripsi tugas dan klik "Simpan Tugas".
- Anda akan diarahkan kembali ke daftar tugas dengan tugas baru yang ditambahkan.

8. Uji Penghapusan Tugas:

- Klik link "Hapus" di samping tugas yang ingin Anda hapus.
- Tugas akan dihapus dari daftar.

9. Eksplorasi dan Modifikasi:

- Perhatikan bagaimana `index.php` bertindak sebagai satu-satunya titik masuk (front controller).
- Amati bagaimana `TaskController` mengkoordinasikan antara `TaskModel` dan `views`.
- Coba tambahkan fitur "tandai selesai" ke aplikasi. Ini akan melibatkan penambahan metode di `TaskModel` dan logika di `TaskController`, serta perubahan di `task_list.php`.
- Ganti penyimpanan data dari file JSON ke database MySQL yang telah Anda buat di praktikum sebelumnya. Ini akan melibatkan modifikasi `TaskModel.php` untuk menggunakan PDO atau MySQLi.

Referensi

[1] Wikipedia. (n.d.). *Model–view–controller*. Retrieved from

<https://en.wikipedia.org/wiki/Model%20view%20controller>

[2] GeeksforGeeks. (n.d.). *MVC Architecture in PHP*. Retrieved from <https://www.geeksforgeeks.org/mvc-architecture-in-php/>

Praktikum 9: Instalasi dan Konfigurasi CodeIgniter 4

1. Tujuan Pembelajaran

Setelah menyelesaikan praktikum ini, mahasiswa diharapkan mampu:

- Memahami konsep dasar framework PHP dan keuntungannya.
- Menginstal CodeIgniter 4 menggunakan Composer.
- Memahami struktur direktori dasar proyek CodeIgniter 4.
- Melakukan konfigurasi dasar aplikasi CodeIgniter 4, termasuk pengaturan database.
- Membuat routing dasar dan memahami cara kerja Controller di CodeIgniter 4.

2. Materi Pembelajaran

2.1. Pengantar Framework PHP

Framework PHP adalah kerangka kerja yang menyediakan struktur dasar untuk membangun aplikasi web. Mereka menyediakan kumpulan pustaka, alat, dan konvensi yang membantu pengembang membangun aplikasi lebih cepat, lebih efisien, dan dengan kode yang lebih terorganisir. Framework umumnya mengimplementasikan pola desain seperti MVC, yang telah kita pelajari di praktikum sebelumnya. [1]

Keuntungan Menggunakan Framework:

- **Pengembangan Cepat:** Menyediakan komponen siap pakai dan struktur yang terorganisir.
- **Kode Terstruktur:** Mendorong praktik terbaik dan pola desain seperti MVC.
- **Keamanan:** Banyak framework memiliki fitur keamanan bawaan (misalnya, perlindungan CSRF, XSS).
- **Pemeliharaan Mudah:** Kode yang terorganisir lebih mudah dipelihara dan di-debug.
- **Skalabilitas:** Membantu membangun aplikasi yang dapat tumbuh dan berkembang.
- **Komunitas:** Memiliki komunitas besar yang menyediakan dukungan dan sumber daya.

2.2. Pengantar CodeIgniter 4

CodeIgniter adalah framework aplikasi web PHP yang kuat dengan jejak yang sangat kecil, dibangun untuk pengembang yang membutuhkan toolkit sederhana dan elegan untuk membuat aplikasi web berfitur lengkap. CodeIgniter 4 adalah versi terbaru yang membawa banyak peningkatan, termasuk penggunaan Composer, struktur yang lebih modern, dan peningkatan kinerja. [2]

2.3. Persyaratan Sistem

Untuk menjalankan CodeIgniter 4, Anda memerlukan:

- PHP versi 7.4 atau lebih tinggi.
- Ekstensi PHP `intl`, `mbstring`, `json`, `mysqlnd` (jika menggunakan MySQL), `xml`.
- Server web (Apache atau Nginx) dengan `mod_rewrite` diaktifkan (untuk Apache).

2.4. Instalasi CodeIgniter 4 dengan Composer

Composer adalah alat manajemen dependensi untuk PHP. Ini memungkinkan Anda mendeklarasikan pustaka yang diandalkan proyek Anda, dan itu akan mengelola (menginstal/memperbarui) untuk Anda. [3]

Langkah-langkah Instalasi:

1. **Pastikan Composer Terinstal:** Jika belum, unduh dan instal Composer dari getcomposer.org.

2. **Buka Terminal/CMD:** Navigasikan ke direktori `htdocs` (atau `www` untuk LAMPP) Anda.

3. **Jalankan Perintah Composer:**



Perintah ini akan membuat folder baru bernama `project-ci4` yang berisi instalasi CodeIgniter 4.

4. **Akses Aplikasi:**

- Buka browser dan navigasikan ke `http://localhost/project-ci4/public`.
- Anda seharusnya melihat halaman selamat datang CodeIgniter 4.

2.5. Struktur Direktori Dasar

Setelah instalasi, Anda akan melihat struktur direktori seperti ini:

A screenshot of a file explorer window. The window title is 'Project-CI4'. The tree view shows the following directory structure:

```
Project-CI4
├── app/          # Kode aplikasi Anda (Controllers, Models, Views, dll.)
│   ├── Config/    # File konfigurasi
│   ├── Controllers/ # Kelas Controller
│   ├── Database/   # Migrasi dan Seeder database
│   ├── Filters/    # Filter HTTP
│   ├── Language/   # File bahasa
│   ├── Libraries/  # Pustaka kustom
│   ├── Models/     # Kelas Model
│   ├── ThirdParty/ # Pustaka pihak ketiga
│   └── Views/      # File View (template HTML)
├── public/        # Root dokumen web (file yang dapat diakses publik)
│   ├── index.php   # Titik masuk utama aplikasi
│   └── .htaccess   # Konfigurasi server web
├── system/        # Core CodeIgniter (jangan diubah)
├── tests/         # File pengujian
└── writable/     # Direktori untuk file yang dapat ditulis (cache, log, uploads)
```

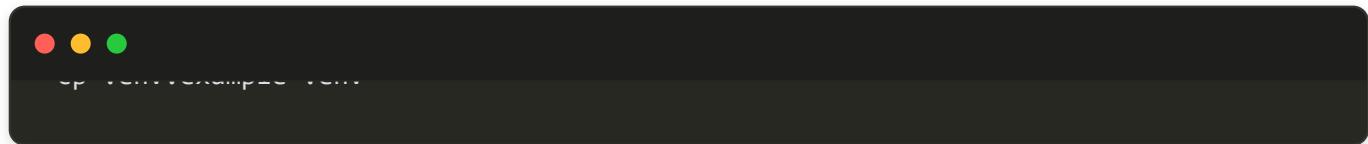
The file explorer has a dark theme with light-colored text and icons.

```
|── .env           # File konfigurasi lingkungan (environment variables)
|── composer.json # Definisi dependensi Composer
└── spark          # Alat baris perintah CodeIgniter
```

2.6. Konfigurasi Dasar

2.6.1. File .env

File `.env` digunakan untuk menyimpan konfigurasi sensitif atau spesifik lingkungan. Salin file `env` ke `.env` dan sesuaikan.



Buka file `.env` dan ubah:

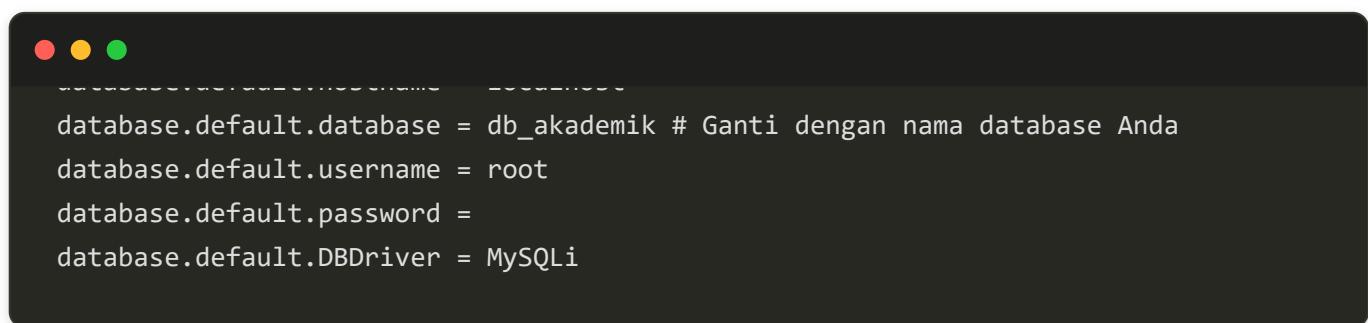
- `CI_ENVIRONMENT = development` (untuk mode pengembangan)
- `app.baseURL = 'http://localhost/project-ci4/'` (sesuaikan dengan URL proyek Anda)

2.6.2. Konfigurasi Database

Buka `app/Config/Database.php` atau atur di `.env`.

Menggunakan `.env` (Direkomendasikan):

Tambahkan baris berikut di file `.env` Anda:



2.6.3. Konfigurasi Autoload (Opsional)

Untuk memuat pustaka atau helper secara otomatis, edit `app/Config/Autoload.php`.

2.7. Routing Dasar

Routing adalah proses menentukan bagaimana URL permintaan ditangani oleh aplikasi. Di CodeIgniter, routing didefinisikan di `app/Config/Routes.php`.

Contoh Routing:

```
// app/Config/Routes.php

$routes->get('/', 'Home::index'); // Rute default

// Rute kustom
$routes->get('/about', 'Home::about');
$routes->get('/users/(:num)', 'Users::show/$1'); // Rute dengan parameter
```

2.8. Controller Dasar

Controller adalah kelas yang menangani permintaan HTTP dan mengembalikan respons. Mereka terletak di `app/Controllers`.

Contoh Controller (`app/Controllers/Home.php`):

```
<?php namespace App\Controllers;

class Home extends BaseController
{
    public function index()
    {
        return view('welcome_message');
    }

    public function about()
    {
        echo 'Ini adalah halaman About Us.';
    }
}
```

3. Pembahasan

Praktikum ini adalah langkah pertama Anda dalam menggunakan framework PHP yang sebenarnya. CodeIgniter 4 akan secara drastis mempercepat pengembangan Anda dengan menyediakan struktur yang jelas dan banyak fungsionalitas bawaan. Memahami proses instalasi dengan Composer dan konfigurasi dasar, terutama pengaturan database dan routing, adalah kunci untuk memulai proyek CodeIgniter.

Pastikan Anda telah menginstal Composer dengan benar dan semua ekstensi PHP yang diperlukan telah diaktifkan. Perhatikan bahwa semua akses ke aplikasi CodeIgniter Anda harus melalui folder `public`. Ini adalah praktik keamanan standar untuk mencegah akses langsung ke file-file sensitif di luar folder `public`.

4. Contoh Kode

Kita akan membuat controller dan view sederhana untuk menguji routing.

1. Instalasi CodeIgniter:

- Buka terminal/CMD Anda.
- Navigasikan ke direktori `htdocs` (atau `www`).
- Jalankan perintah: `composer create-project codeigniter4/appstarter ci4_app`

2. Konfigurasi `.env`:

- Navigasikan ke folder `ci4_app`.
- Salin `env` menjadi `.env` (`cp env .env` di Linux/macOS, `copy env .env` di Windows CMD).
- Buka file `.env` dan ubah baris berikut:



```
app.baseURL = 'http://localhost/ci4_app/'
```

3. Buat Controller Baru (`app\Controllers\Produk.php`):

```
<?php namespace App\Controllers;

use App\Controllers\BaseController;

class Produk extends BaseController
{
    public function index()
    {
        $data["judul"] = "Daftar Produk";
        $data["produk_list"] = [
            ["nama" => "Laptop", "harga" => 12000000],
            ["nama" => "Mouse", "harga" => 150000],
            ["nama" => "Keyboard", "harga" => 30000]
        ];
        return view("produk_view", $data);
    }
}
```

```
}

public function detail($id)
{
    echo "Detail Produk ID: " . $id;
}
}
```

4. Buat View Baru (`app/Views/produk_view.php`):

```
<!DOCTYPE html>
<html>
<head>
    <title><?= $judul ?></title>
    <style>
        body { font-family: Arial, sans-serif; margin: 20px; }
        table { width: 80%; border-collapse: collapse; margin-top: 20px; }
        th, td { border: 1px solid #ddd; padding: 8px; text-align: left; }
        th { background-color: #f2f2f2; }
    </style>
</head>
<body>
    <h1><?= $judul ?></h1>

    <table>
        <thead>
            <tr>
                <th>Nama Produk</th>
                <th>Harga</th>
            </tr>
        </thead>
        <tbody>
            <?php foreach ($produk_list as $produk): ?>
            <tr>
                <td><?= esc($produk["nama"]) ?></td>
                <td>Rp. <?= number_format	esc($produk["harga"]), 0, ',', '.') ?>
            </tr>
        <?php endforeach; ?>
        </tbody>
    </table>
```

```

<p>Kembali ke <a href="= base_url() ?&gt;"Home</a></p>
</body>
</html>

```

5. Konfigurasi Routes (`app/Config/Routes.php`):

Tambahkan baris berikut di dalam `$routes->group('/')` atau di bagian bawah file:

```

$routes->get('/produk', 'Produk::index');
$routes->get('/produk/(:num)', 'Produk::detail/$1');

```

5. Langkah-langkah Praktikum

1. Instalasi CodeIgniter 4:

- Ikuti langkah 1 di bagian "Contoh Kode" untuk menginstal CodeIgniter 4 ke folder `ci4_app` .
- Pastikan Anda dapat mengakses halaman selamat datang di http://localhost/ci4_app/public .

2. Konfigurasi `.env` :

- Ikuti langkah 2 di bagian "Contoh Kode" untuk menyalin dan mengedit file `.env` .

3. Buat Controller `Produk.php` :

- Buat file `Produk.php` di `ci4_app/app/Controllers` .
- Salin dan tempel kode `Produk.php` dari bagian "Contoh Kode".

4. Buat View `produk_view.php` :

- Buat file `produk_view.php` di `ci4_app/app/Views` .
- Salin dan tempel kode `produk_view.php` .

5. Konfigurasi Routes:

- Buka `ci4_app/app/Config/Routes.php` .
- Tambahkan baris routing untuk `/produk` dan `/produk/(:num)` seperti yang ditunjukkan di bagian "Contoh Kode".

6. Uji Aplikasi:

- Akses http://localhost/ci4_app/public/produk di browser Anda. Anda seharusnya melihat daftar produk.

- Coba akses `http://localhost/ci4_app/public/produk/123`. Anda seharusnya melihat "Detail Produk ID: 123".

7. Eksplorasi dan Modifikasi:

- Coba tambahkan lebih banyak data produk di `Produk::index()`.
 - Buat metode baru di `Produk Controller` (misalnya `tambahProduk()`) dan buat rute baru untuk itu.
 - Pelajari lebih lanjut tentang `BaseController` dan bagaimana Controller di CodeIgniter bekerja.
 - Eksplorasi file konfigurasi lain di `app/Config`.
-

Referensi

- [1] CodeIgniter. (n.d.). *What is CodeIgniter?*. Retrieved from <https://codeigniter.com/>
- [2] CodeIgniter. (n.d.). *Installation*. Retrieved from <https://codeigniter4.github.io/userguide/installation/index.html>
- [3] Composer. (n.d.). *Getting Started*. Retrieved from <https://getcomposer.org/doc/00-intro.md>

Praktikum 10: CRUD dengan CodeIgniter 4

1. Tujuan Pembelajaran

Setelah menyelesaikan praktikum ini, mahasiswa diharapkan mampu:

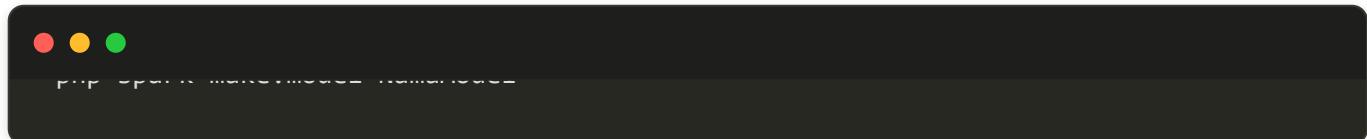
- Mengimplementasikan operasi Create, Read, Update, dan Delete (CRUD) menggunakan CodeIgniter 4.
- Membuat Model untuk berinteraksi dengan database.
- Menggunakan Query Builder CodeIgniter untuk operasi database.
- Menangani form dan validasi data di CodeIgniter 4.

2. Materi Pembelajaran

2.1. Model di CodeIgniter 4

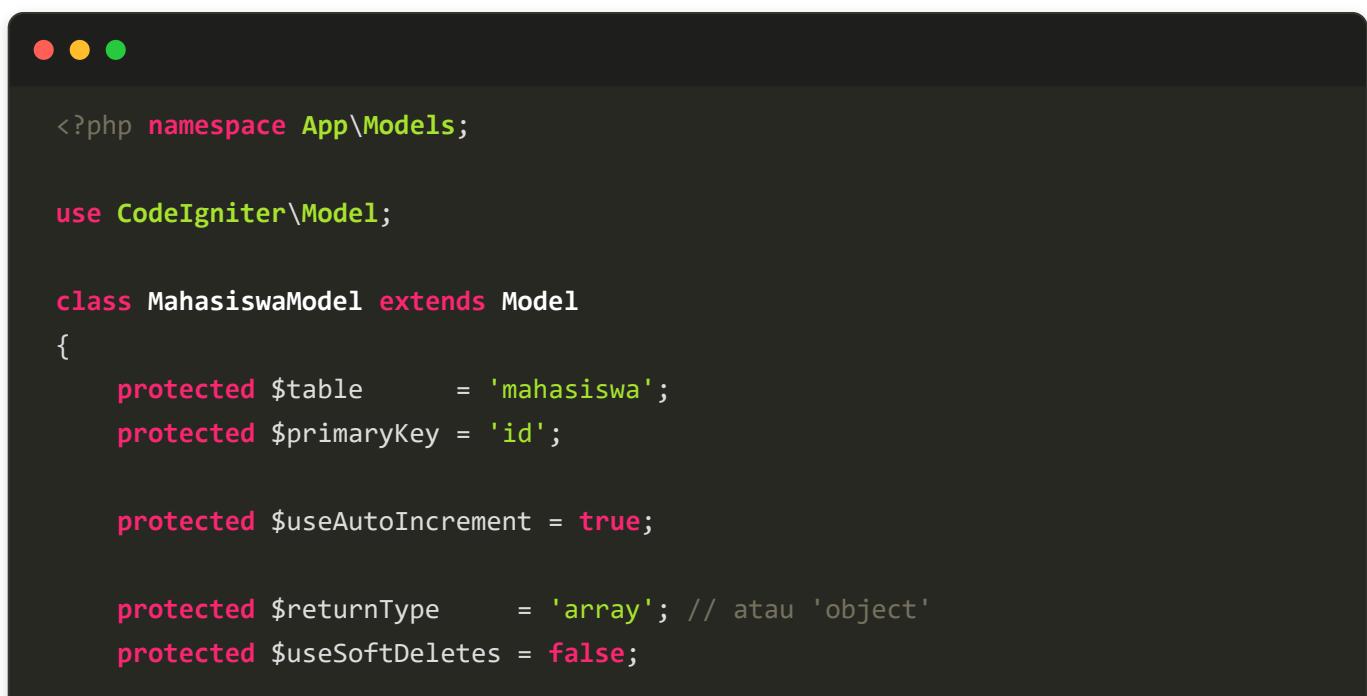
Model adalah kelas yang dirancang untuk bekerja dengan informasi dalam database. Setiap tabel database biasanya memiliki Model yang sesuai. Model di CodeIgniter 4 menyediakan fungsionalitas CRUD dasar secara otomatis, serta fitur-fitur canggih seperti validasi data, event, dan soft deletes. [1]

Untuk membuat Model, Anda dapat menggunakan perintah `spark` :



```
PSU:spark ~ % spark generate-model MahasiswaModel
```

Contoh Model (`app/Models/MahasiswaModel.php`):



```
<?php namespace App\Models;

use CodeIgniter\Model;

class MahasiswaModel extends Model
{
    protected $table      = 'mahasiswa';
    protected $primaryKey = 'id';

    protected $useAutoIncrement = true;

    protected $returnType     = 'array'; // atau 'object'
    protected $useSoftDeletes = false;
}
```

```

protected $allowedFields = ['nim', 'nama', 'jurusan', 'email'];

protected $useTimestamps = false;
protected $createdField = 'created_at';
protected $updatedField = 'updated_at';
protected $deletedField = 'deleted_at';

protected $validationRules = [];
protected $validationMessages = [];
protected $skipValidation = false;

}

```

2.2. Query Builder CodeIgniter

CodeIgniter 4 menyediakan Query Builder Class yang memungkinkan Anda membangun query database menggunakan pendekatan yang lebih berorientasi objek. Ini membantu dalam menulis query yang lebih aman (melindungi dari SQL Injection) dan lebih mudah dibaca. [2]

Contoh Penggunaan Query Builder di Controller/Model:

- **Insert:**



```

$data = [
    'nim' => '2023008',
    'nama' => 'Fajar Pratama',
    'jurusan' => 'Teknik Elektro',
    'email' => 'fajar.p@example.com'
];
$this->db->table('mahasiswa')->insert($data);
// Atau jika menggunakan Model:
$mahasiswaModel->insert($data);

```

- **Select:**



```

$query = $this->db->table('mahasiswa')->get();
// Atau jika menggunakan Model:
$mahasiswa = $mahasiswaModel->findAll(); // Mengambil semua data

```

```
$mahasiswa = $mahasiswaModel->find(1); // Mengambil data berdasarkan primary key  
$mahasiswa = $mahasiswaModel->where('jurusan', 'Teknik Informatika')->findAll()
```

- **Update:**

```
$data = [  
    'jurusan' => 'Teknik Komputer'  
];  
$this->db->table('mahasiswa')->where('nim', '2023008')->update($data);  
// Atau jika menggunakan Model:  
$mahasiswaModel->update(1, $data); // Update data dengan ID 1
```

- **Delete:**

```
$this->db->table('mahasiswa')->where('nim', '2023008')->delete();  
// Atau jika menggunakan Model:  
$mahasiswaModel->delete(1); // Hapus data dengan ID 1
```

2.3. Form Handling dan Validasi di CodeIgniter 4

CodeIgniter 4 menyediakan pustaka Form Validation yang kuat untuk memvalidasi input pengguna. Validasi dapat dilakukan di Controller atau di Model. [3]

Contoh Validasi di Controller:

```
<?php namespace App\Controllers;  
  
use App\Controllers\BaseController;  
  
class Mahasiswa extends BaseController  
{  
    public function simpan()  
    {  
        if (! $this->request->is('post')) {  
            return view('form_mahasiswa');
```

```
}

$rules = [
    'nim' => 'required|min_length[5]|max_length[10]|is_unique[mahasiswa.nim]',
    'nama' => 'required|min_length[3]|max_length[100]',
    'email' => 'required|valid_email|is_unique[mahasiswa.email]'
];

if (! $this->validate($rules)) {
    return view('form_mahasiswa', ['validation' => $this->validator]);
}

// Data valid, proses penyimpanan
$data = [
    'nim' => $this->request->getPost('nim'),
    'nama' => $this->request->getPost('nama'),
    'jurusan' => $this->request->getPost('jurusan'),
    'email' => $this->request->getPost('email'),
];
// ... simpan data ke database ...

return redirect()->to('/mahasiswa')->with('message', 'Data berhasil disimpan')
}
}
```

3. Pembahasan

Praktikum ini adalah puncak dari semua yang telah Anda pelajari, mengintegrasikan PHP, MySQL, OOP, MVC, dan sekarang framework CodeIgniter 4 untuk membangun aplikasi CRUD yang lengkap. Anda akan melihat bagaimana CodeIgniter menyederhanakan proses pengembangan dengan menyediakan struktur yang jelas dan alat bantu yang efisien untuk berinteraksi dengan database dan menangani input pengguna.

Pastikan Anda telah menyelesaikan instalasi CodeIgniter 4 dari Praktikum 9 dan telah mengkonfigurasi koneksi database di file `.env` atau `app/Config/Database.php`. Database `db_akademik` dan tabel `mahasiswa` dari Praktikum 3 dan 5 akan digunakan kembali di sini.

4. Contoh Kode

Kita akan membuat aplikasi CRUD mahasiswa sederhana menggunakan CodeIgniter 4.

1. Konfigurasi Database:

- Pastikan Anda telah mengkonfigurasi database di file `.env` atau `app/Config/Database.php` dari Praktikum 9, menggunakan database `db_akademik` dan tabel `mahasiswa`.

2. Buat Model Mahasiswa (app/Models/MahasiswaModel.php):

```
<?php namespace App\Models;

use CodeIgniter\Model;

class MahasiswaModel extends Model
{
    protected $table      = 'mahasiswa';
    protected $primaryKey = 'id';

    protected $useAutoIncrement = true;

    protected $returnType     = 'array';
    protected $useSoftDeletes = false;

    protected $allowedFields = ['nim', 'nama', 'jurusan', 'email'];

    protected $useTimestamps = false;
    protected $createdField  = 'created_at';
    protected $updatedField  = 'updated_at';
    protected $deletedField  = 'deleted_at';

    protected $validationRules     = [
        'nim'      => 'required|min_length[5]|max_length[10]|is_unique[mahasiswa.nim]',
        'nama'     => 'required|min_length[3]|max_length[100]',
        'jurusan'  => 'permit_empty|max_length[50]',
        'email'    => 'required|valid_email|is_unique[mahasiswa.email,id,{id}]'
    ];
    protected $validationMessages = [
        'nim' => [
            'required' => 'NIM harus diisi.',
            'min_length' => 'NIM minimal 5 karakter.',
            'max_length' => 'NIM maksimal 10 karakter.',
            'is_unique' => 'NIM sudah terdaftar.'
        ],
        'nama' => [
            'required' => 'Nama harus diisi.',
            'min_length' => 'Nama minimal 3 karakter.',
            'max_length' => 'Nama maksimal 100 karakter.'
        ],
        'email' => [

```

```
        'required' => 'Email harus diisi.',  
        'valid_email' => 'Format email tidak valid.',  
        'is_unique' => 'Email sudah terdaftar.'  
    ]  
];  
protected $skipValidation = false;  
}
```

3. Buat Controller Mahasiswa (app\Controllers\Mahasiswa.php):

```
<?php namespace App\Controllers;  
  
use App\Models\MahasiswaModel;  
  
class Mahasiswa extends BaseController  
{  
    protected $mahasiswaModel;  
  
    public function __construct()  
    {  
        $this->mahasiswaModel = new MahasiswaModel();  
    }  
  
    public function index()  
    {  
        $data = [  
            'mahasiswa' => $this->mahasiswaModel->findAll(),  
            'title' => 'Daftar Mahasiswa'  
        ];  
        return view('mahasiswa/index', $data);  
    }  
  
    public function create()  
    {  
        $data = [  
            'title' => 'Tambah Mahasiswa',  
            'validation' => \Config\Services::validation()  
        ];  
        return view('mahasiswa/create', $data);  
    }  
}
```

```
public function save()
{
    // Validasi input
    if (!$this->validate($this->mahasiswaModel->validationRules, $this->mahasiswa))
        return redirect()->to('/mahasiswa/create')->withInput();
}

$this->mahasiswaModel->save([
    'nim' => $this->request->getVar('nim'),
    'nama' => $this->request->getVar('nama'),
    'jurusan' => $this->request->getVar('jurusan'),
    'email' => $this->request->getVar('email'),
]);

session()->setflashdata('pesan', 'Data berhasil ditambahkan.');
return redirect()->to('/mahasiswa');
}

public function edit($id)
{
    $data = [
        'title' => 'Edit Mahasiswa',
        'validation' => \Config\Services::validation(),
        'mahasiswa' => $this->mahasiswaModel->find($id)
    ];
    return view('mahasiswa/edit', $data);
}

public function update($id)
{
    // Cek NIM lama dan baru untuk validasi unique
    $mahasiswaLama = $this->mahasiswaModel->find($id);
    if ($mahasiswaLama['nim'] == $this->request->getVar('nim')) {
        $rule_nim = 'required|min_length[5]|max_length[10]';
    } else {
        $rule_nim = 'required|min_length[5]|max_length[10]|is_unique[mahasiswa.nim]';
    }

    // Cek Email lama dan baru untuk validasi unique
    if ($mahasiswaLama['email'] == $this->request->getVar('email')) {
        $rule_email = 'required|valid_email';
    } else {
        $rule_email = 'required|valid_email|is_unique[mahasiswa.email]';
    }
}
```

```
$rules = [
    'nim' => $rule_nim,
    'nama' => 'required|min_length[3]|max_length[100]',
    'jurusan' => 'permit_empty|max_length[50]',
    'email' => $rule_email
];

if (!$this->validate($rules, $this->mahasiswaModel->validationMessages)) {
    return redirect()->to('/mahasiswa/edit/' . $id)->withInput();
}

$this->mahasiswaModel->save([
    'id' => $id,
    'nim' => $this->request->getVar('nim'),
    'nama' => $this->request->getVar('nama'),
    'jurusan' => $this->request->getVar('jurusan'),
    'email' => $this->request->getVar('email'),
]);
session()->setflashdata('pesan', 'Data berhasil diubah.');
return redirect()->to('/mahasiswa');
}

public function delete($id)
{
    $this->mahasiswaModel->delete($id);
    session()->setflashdata('pesan', 'Data berhasil dihapus.');
    return redirect()->to('/mahasiswa');
}
}
```

4. Buat Views:

File: app/Views/mahasiswa/index.php



```
<?= $this->extend('layout/template'); ?>

<?= $this->section('content'); ?>
<div class="container">
    <div class="row">
```

```
<div class="col-8">
    <h2 class="my-3">Daftar Mahasiswa</h2>
    <a href="/mahasiswa/create" class="btn btn-primary mb-3">Tambah Data Mahasiswa</a>
    <?php if (session()->getFlashdata('pesan')) : ?>
        <div class="alert alert-success" role="alert">
            <?= session()->getFlashdata('pesan'); ?>
        </div>
    <?php endif; ?>

    <table class="table">
        <thead>
            <tr>
                <th scope="col">#</th>
                <th scope="col">NIM</th>
                <th scope="col">Nama</th>
                <th scope="col">Jurusan</th>
                <th scope="col">Aksi</th>
            </tr>
        </thead>
        <tbody>
            <?php $i = 1; ?>
            <?php foreach ($mahasiswa as $m) : ?>
                <tr>
                    <th scope="row"><?= $i++; ?></th>
                    <td><?= $m['nim']; ?></td>
                    <td><?= $m['nama']; ?></td>
                    <td><?= $m['jurusan']; ?></td>
                    <td>
                        <a href="/mahasiswa/edit/<?= $m['id']; ?>" class="btn btn-warning">Edit</a>
                        <form action="/mahasiswa/<?= $m['id']; ?>" method="post" style="display: inline-block; margin-left: 10px;">
                            <?= csrf_field(); ?>
                            <input type="hidden" name="_method" value="DELETE" />
                            <button type="submit" class="btn btn-danger" onclick="return confirm('Anda yakin ingin menghapus data ini?')>Hapus</button>
                        </form>
                    </td>
                </tr>
            <?php endforeach; ?>
        </tbody>
    </table>
</div>
</div>
<?= $this->endSection(); ?>
```

File: app/Views/mahasiswa/create.php

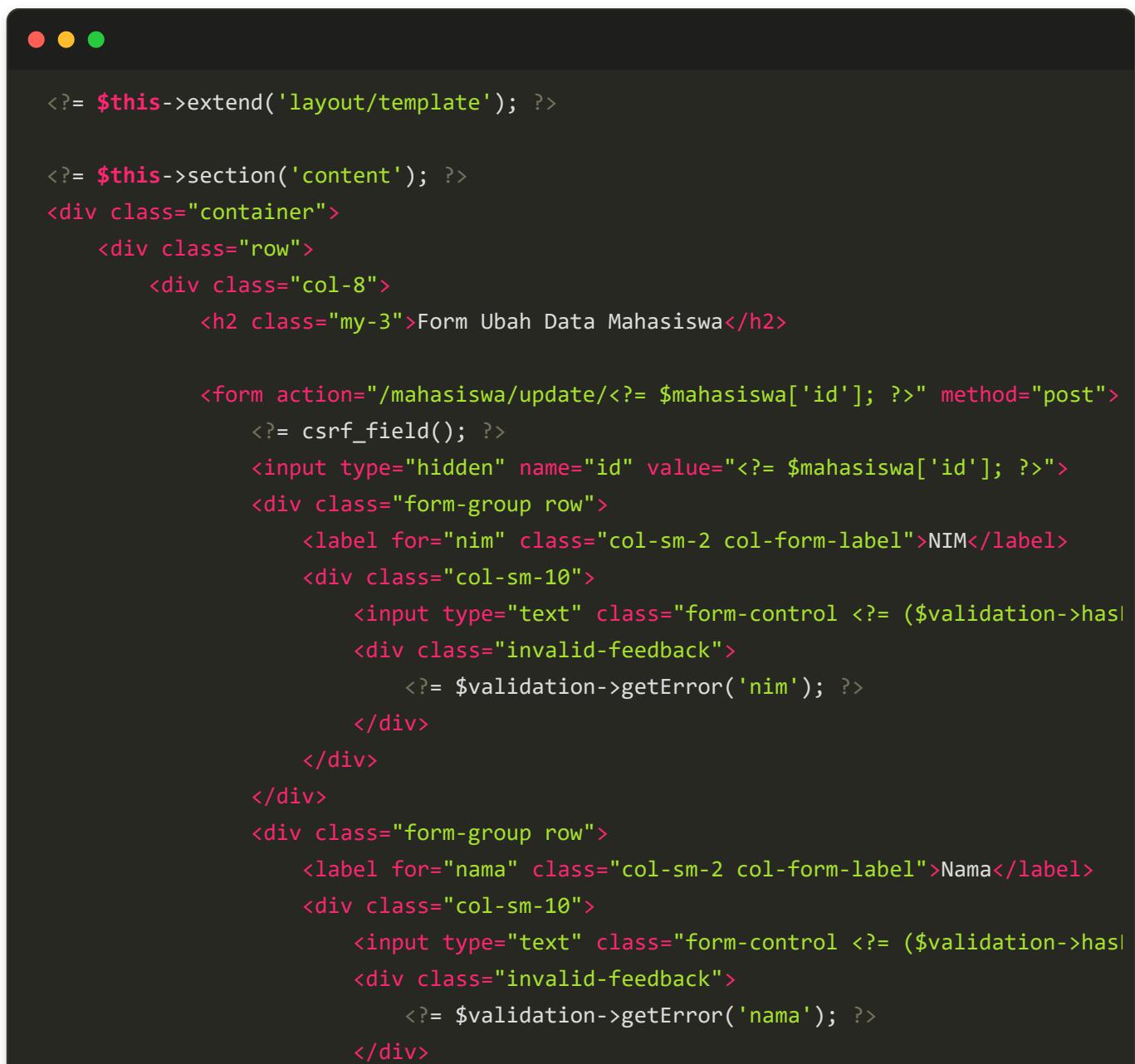
```
<?= $this->extend('layout/template'); ?>

<?= $this->section('content'); ?>
<div class="container">
    <div class="row">
        <div class="col-8">
            <h2 class="my-3">Form Tambah Data Mahasiswa</h2>

            <form action="/mahasiswa/save" method="post">
                <?= csrf_field(); ?>
                <div class="form-group row">
                    <label for="nim" class="col-sm-2 col-form-label">NIM</label>
                    <div class="col-sm-10">
                        <input type="text" class="form-control" <?= ($validation->hasError('nim')) ?>
                        <div class="invalid-feedback">
                            <?= $validation->getError('nim'); ?>
                        </div>
                    </div>
                </div>
                <div class="form-group row">
                    <label for="nama" class="col-sm-2 col-form-label">Nama</label>
                    <div class="col-sm-10">
                        <input type="text" class="form-control" <?= ($validation->hasError('nama')) ?>
                        <div class="invalid-feedback">
                            <?= $validation->getError('nama'); ?>
                        </div>
                    </div>
                </div>
                <div class="form-group row">
                    <label for="jurusan" class="col-sm-2 col-form-label">Jurusan</label>
                    <div class="col-sm-10">
                        <input type="text" class="form-control" id="jurusan" name="jurusan" value="<?= $validation->hasError('jurusan') ?><?= $validation->get('jurusan') ?>">
                    </div>
                </div>
                <div class="form-group row">
                    <label for="email" class="col-sm-2 col-form-label">Email</label>
                    <div class="col-sm-10">
                        <input type="text" class="form-control" <?= ($validation->hasError('email')) ?>
                        <div class="invalid-feedback">
                            <?= $validation->getError('email'); ?>
                        </div>
                    </div>
                </div>
            </form>
        </div>
    </div>
</div>
```

```
        <div class="invalid-feedback">
            <?= $validation->getError('email'); ?>
        </div>
    </div>
    <button type="submit" class="btn btn-primary">Tambah Data</button>
</form>
</div>
</div>
<?= $this->endSection(); ?>
```

File: app/Views/mahasiswa/edit.php



```
<?= $this->extend('layout/template'); ?>

<?= $this->section('content'); ?>
<div class="container">
    <div class="row">
        <div class="col-8">
            <h2 class="my-3">Form Ubah Data Mahasiswa</h2>

            <form action="/mahasiswa/update/<?= $mahasiswa['id']; ?>" method="post">
                <?= csrf_field(); ?>
                <input type="hidden" name="id" value="<?= $mahasiswa['id']; ?>">
                <div class="form-group row">
                    <label for="nim" class="col-sm-2 col-form-label">NIM</label>
                    <div class="col-sm-10">
                        <input type="text" class="form-control <?= ($validation->hasError('nim')) ? 'is-invalid' : ''>" name="nim" value="<?= $mahasiswa['nim']; ?>">
                        <div class="invalid-feedback">
                            <?= $validation->getError('nim'); ?>
                        </div>
                    </div>
                </div>
                <div class="form-group row">
                    <label for="nama" class="col-sm-2 col-form-label">Nama</label>
                    <div class="col-sm-10">
                        <input type="text" class="form-control <?= ($validation->hasError('nama')) ? 'is-invalid' : ''>" name="nama" value="<?= $mahasiswa['nama']; ?>">
                        <div class="invalid-feedback">
                            <?= $validation->getError('nama'); ?>
                        </div>
                    </div>
                </div>
            </form>
        </div>
    </div>
</div>
```

```
        </div>
    </div>
    <div class="form-group row">
        <label for="jurusan" class="col-sm-2 col-form-label">Jurusan</label>
        <div class="col-sm-10">
            <input type="text" class="form-control" id="jurusan" name="jurusan" value="Teknik Informatika" required="required" />
        </div>
    </div>
    <div class="form-group row">
        <label for="email" class="col-sm-2 col-form-label">Email</label>
        <div class="col-sm-10">
            <input type="text" class="form-control" id="email" name="email" value="yamin@unimed.ac.id" required="required" />
            <div class="invalid-feedback">
                <?= $validation->getError('email'); ?>
            </div>
        </div>
    </div>
    <button type="submit" class="btn btn-primary">Ubah Data</button>
</form>
</div>
</div>
<?= $this->endSection(); ?>
```

File: app/Views/layout/template.php (Layout Dasar - Opsional, tapi direkomendasikan)

```
<!doctype html>
<html lang="en">
    <head>
        <!-- Required meta tags -->
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" />
        <!-- Bootstrap CSS -->
        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css" integrity="sha384-9�
```

```

<div class="container">
    <a class="navbar-brand" href="#">CI4 App</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
        <div class="navbar-nav">
            <a class="nav-item nav-link active" href="/">Home <span class="sr-only">Active</span>
            <a class="nav-item nav-link" href="/mahasiswa">Mahasiswa</a>
        </div>
    </div>
</div>
</nav>

<?= $this->renderSection('content'); ?>

<!-- Optional JavaScript -->
<!-- jQuery first, then Popper.js, then Bootstrap JS -->
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-pJjzG7L7tHdI76DQ3oLcD6yR7EjZqV5CLfWp3YU+Xz5q7kYXQ" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-Q2hwPVSxV/SOvJFz2J5hKXHn5yJmVZQW6NpOZTgjVXgkZC&lt;script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js" integrity="sha384-OgVRvuATPZKwMlJ+D0Lj60Jl96P9i4jXeDjWZGOF0uJ3Ji6B" crossorigin="anonymous"></script>
</body>
</html>

```

5. Konfigurasi Routes (app/Config/Routes.php):

Tambahkan atau modifikasi rute berikut:

```
$routes->get('/', 'Home::index');

$routes->get('/mahasiswa', 'Mahasiswa::index');
$routes->get('/mahasiswa/create', 'Mahasiswa::create');
$routes->post('/mahasiswa/save', 'Mahasiswa::save');
$routes->get('/mahasiswa/edit/(:num)', 'Mahasiswa::edit/$1');
$routes->post('/mahasiswa/update/(:num)', 'Mahasiswa::update/$1');
$routes->delete('/mahasiswa/(:num)', 'Mahasiswa::delete/$1');
```

5. Langkah-langkah Praktikum

1. Persiapan Proyek CodeIgniter:

- Pastikan Anda telah menginstal CodeIgniter 4 (misalnya di folder `ci4_app`) dan telah mengkonfigurasi database `db_akademik` serta tabel `mahasiswa` seperti yang dijelaskan di Praktikum 9.

2. Buat Model Mahasiswa:

- Buka terminal/CMD di root folder `ci4_app`.
- Jalankan perintah: `php spark make:model MahasiswaModel`.
- Buka file `app/Models/MahasiswaModel.php` dan salin/tempel kode dari bagian "Contoh Kode" di atas, termasuk `validationRules` dan `validationMessages`.

3. Buat Controller Mahasiswa:

- Buat file `app/Controllers/Mahasiswa.php`.
- Salin dan tempel kode `Mahasiswa.php` dari bagian "Contoh Kode".

4. Buat Struktur Folder Views:

- Di dalam `app/Views`, buat folder baru bernama `mahasiswa`.
- Di dalam `app/Views`, buat folder baru bernama `layout`.

5. Buat File Views:

- Buat `app/Views/mahasiswa/index.php`, `app/Views/mahasiswa/create.php`, dan `app/Views/mahasiswa/edit.php`.
- Salin dan tempel kode masing-masing file dari bagian "Contoh Kode".
- Buat `app/Views/layout/template.php` dan salin/tempel kodennya. (Ini adalah layout dasar yang menggunakan Bootstrap untuk tampilan, Anda bisa menyesuaikannya).

6. Konfigurasi Routes:

- Buka `app/Config/Routes.php`.
- Tambahkan atau modifikasi rute seperti yang ditunjukkan di bagian "Contoh Kode" untuk mengarahkan permintaan ke Controller `Mahasiswa`.

7. Uji Aplikasi CRUD:

- Akses `http://localhost/ci4_app/public/mahasiswa` di browser Anda.
- **Read:** Anda seharusnya melihat daftar mahasiswa (jika ada data di tabel `mahasiswa`).
- **Create:** Klik "Tambah Data Mahasiswa", isi form, dan klik "Tambah Data". Perhatikan validasi jika ada input yang salah. Data baru seharusnya muncul di daftar.
- **Update:** Klik "Edit" pada salah satu baris data, ubah informasi, dan klik "Ubah Data". Data seharusnya diperbarui di daftar.
- **Delete:** Klik "Delete" pada salah satu baris data dan konfirmasi. Data seharusnya terhapus dari daftar.

8. Eksplorasi Lanjutan:

- Pelajari lebih lanjut tentang fitur validasi di CodeIgniter, termasuk aturan kustom.
 - Eksplorasi fitur pagination untuk daftar data yang besar.
 - Coba implementasikan fitur pencarian data mahasiswa.
 - Integrasikan dengan library pihak ketiga lainnya menggunakan Composer.
-

Referensi

- [1] CodeIgniter. (n.d.). *Models*. Retrieved from <https://codeigniter4.github.io/userguide/models/model.html>
- [2] CodeIgniter. (n.d.). *Query Builder*. Retrieved from
https://codeigniter4.github.io/userguide/database/query_builder.html
- [3] CodeIgniter. (n.d.). *Form Validation*. Retrieved from
<https://codeigniter4.github.io/userguide/libraries/validation.html>