



POLITEKNIK ELEKTRONIKA NEGERI SURABAYA



DIKTISAINTEK
BERDAMPAK



PHP FORM & Koneksi Database

Workshop Pemrograman Web

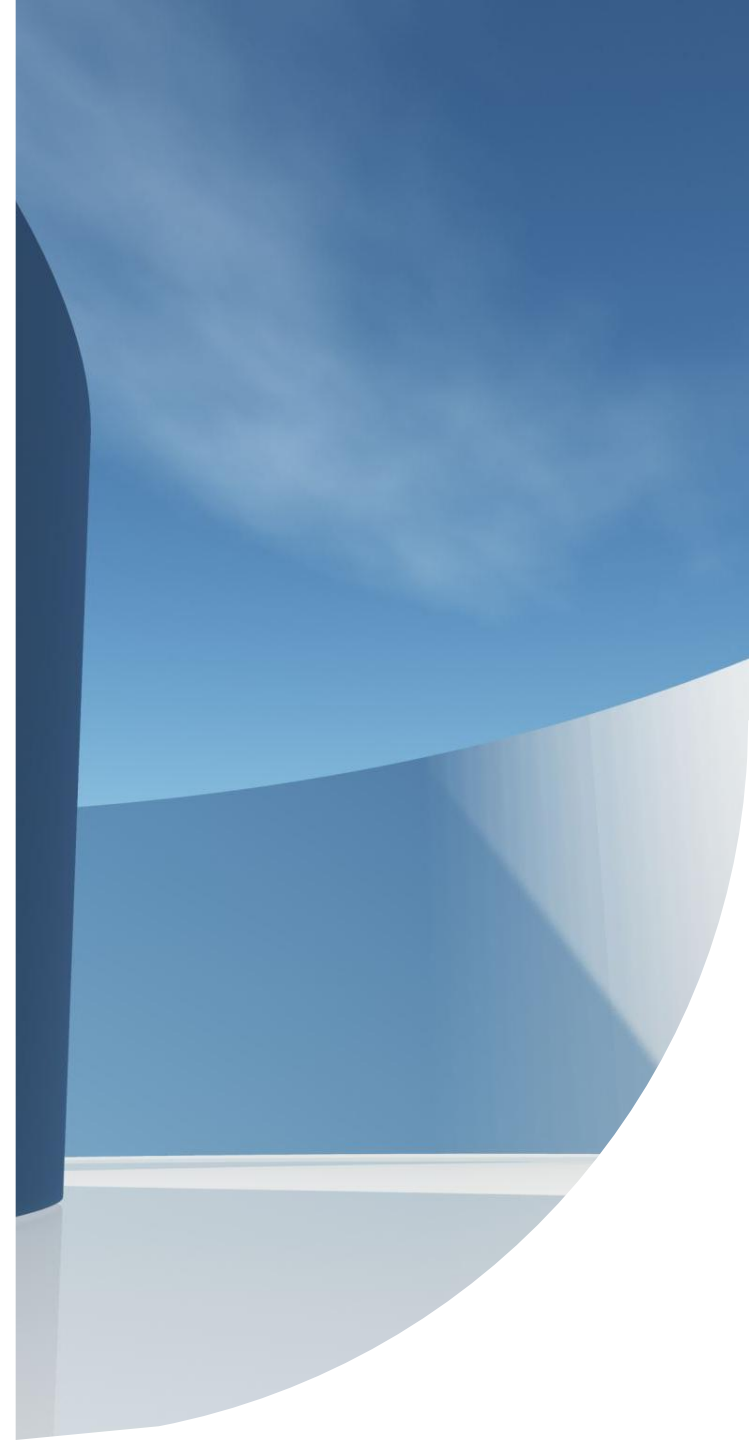
Yunia Ikawati

Teknik Informatika-PENS





PHP FORM





FORM

- ❖ Elemen interaktif yang memungkinkan pengguna untuk memasukkan data atau informasi, yang kemudian dapat diproses oleh server atau aplikasi web.
- ❖ Formulir ini biasanya digunakan untuk berbagai keperluan, seperti **mendaftar akun, login, mengirim pesan, memberikan feedback, atau bahkan melakukan pembayaran.**



KOMPONEN UTAMA FORM

1. Tag **<form>**: Digunakan untuk mendefinisikan form.
2. Input Fields: Area tempat pengguna memasukkan data, seperti:
 - ✓ Teks (`<input type="text">`)
 - ✓ Password (`<input type="password">`)
 - ✓ Email (`<input type="email">`)
 - ✓ Textarea untuk teks yang lebih panjang.

KOMPONEN UTAMA FORM

3. Elemen Pilihan:

- ✓ **Dropdown** (<select>) untuk memilih opsi.
- ✓ **Checkbox** untuk memilih lebih dari satu opsi.
- ✓ **Radio Button** untuk memilih salah satu dari beberapa opsi.

4. Tombol Pengiriman:

- ✓ **Submit** (<input type="submit">) untuk mengirim data.
- ✓ **Reset** (<input type="reset">) untuk menghapus data yang sudah diisi.

5. Attribute action dan method:

- ✓ **action**: Menentukan URL tempat data akan dikirim.
- ✓ **method**: Menentukan metode pengiriman data (POST atau GET).



CONTOH FORM

```
<html>
<body>

<form action="welcome.php" method="POST">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

Name:

E-mail:

welcome.php

```
<html>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>

</body>
</html>
```

Welcome yunia
Your email address is: yunia@gmail.com

METHOD FORM PHP

- Dalam pembuatan form di web, terdapat dua metode utama yang digunakan untuk mengirimkan data dari form ke server, yaitu GET dan POST.

Metode	Keunggulan	Kekurangan
GET	Mudah digunakan, bookmarkable URL	Tidak cocok untuk data sensitif
POST	Aman untuk data sensitif	Tidak bookmarkable, butuh SSL/TLS

Metode GET

- Data yang dikirim akan terlihat di URL (ditambahkan sebagai query string setelah tanda ?).
- Cocok digunakan untuk permintaan yang tidak bersifat sensitif atau membutuhkan penyimpanan dalam URL, seperti pencarian.
- Batasan ukuran data: tergantung pada browser, biasanya sekitar 2.048 karakter.
- Keamanan: Kurang aman, karena data terlihat oleh pengguna dan dapat disalin dengan mudah.

CONTOH FORM DENGAN METHOD “GET”

```
<html>
<body>

<form action="welcome_get.php" method="GET">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

Name:

E-mail:

welcome_get.php

```
<html>
<body>

Welcome <?php echo $_GET["name"]; ?><br>
Your email address is: <?php echo $_GET["email"]; ?>

</body>
</html>
```

Welcome IKA
Your email address is: ika@gmail.com

Metode POST

- Data dikirim melalui body permintaan HTTP, sehingga tidak terlihat di URL.
- Cocok untuk mengirim data yang lebih besar atau bersifat sensitif, seperti login atau pendaftaran.
- Tidak memiliki batasan ukuran data yang signifikan.
- Keamanan: Lebih aman dibandingkan GET, tetapi sebaiknya dipadukan dengan SSL/TLS (HTTPS) untuk meningkatkan keamanan.

CONTOH FORM DENGAN METHOD "POST"

```
<html>
<body>

<form action="welcome.php" method="POST">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

Name:

E-mail:

welcome.php

```
<html>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>

</body>
</html>
```

Welcome yunia
Your email address is: yunia@gmail.com

Form Tanpa Validasi

- Tidak Ada Pemeriksaan Data
- Keamanan Rendah
- Pengguna Kurang Dipandu
- Kode Lebih Sederhana

```
<!DOCTYPE html>
<html>
<body>
    <form method="post">
        Nama: <input type="text" name="nama"><br>
        Email: <input type="text" name="email"><br>
        <input type="submit" value="Kirim">
    </form>

    <?php
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        echo "Nama: " . $_POST["nama"] . "<br>";
        echo "Email: " . $_POST["email"];
    }
    ?>
</body>
</html>
```

Validasi Formulir PHP

PHP Form Validation adalah proses untuk memastikan data yang dimasukkan pengguna ke dalam form sesuai dengan aturan tertentu sebelum data tersebut diproses lebih lanjut.

1. **Validasi di sisi klien (client-side validation):** Dilakukan menggunakan JavaScript sebelum data dikirim ke server. Namun, ini tidak sepenuhnya aman karena pengguna bisa memanipulasi kode di browser.
2. **Validasi di sisi server (server-side validation):** Dilakukan menggunakan PHP di server untuk memastikan data yang diterima valid. Hal ini lebih aman dan merupakan langkah yang sangat penting.

Validasi Formulir PHP

Kode HTML untuk Form Validation

```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

- ✓ **\$_SERVER["PHP_SELF"]** adalah variabel super global yang mengembalikan nama file skrip yang sedang dijalankan.
- ✓ **\$_SERVER["PHP_SELF"]** mengirimkan data formulir yang dikirimkan ke halaman itu sendiri. Pengguna akan mendapatkan pesan kesalahan pada halaman yang sama dengan formulir.
- ✓ Fungsi **htmlspecialchars()** mengubah karakter khusus HTML seperti < dan > dengan < dan > menjadi entitas HTML.
- ✓ Fungsi **htmlspecialchars()** mencegah penyerang mengeksploitasi kode HTML atau Javascript (serangan Cross-site Scripting) dalam formulir.



Form dengan Validasi

- **Ada Pemeriksaan Data:** Data diperiksa untuk memastikan validitas (contohnya: format email, panjang kata sandi, dll.).
- **Keamanan Lebih Tinggi:** Menghindari serangan seperti SQL injection atau data tidak valid
- **Pengalaman Pengguna Lebih Baik:** Pengguna diberi tahu jika ada kesalahan dalam input
- **Kode Lebih Kompleks:** Membutuhkan logika tambahan untuk memvalidasi input.

CONTOH FORM VALIDASI

```
<html>
<body>
  <form method="post">
    Nama: <input type="text" name="nama"><br>
    Email: <input type="email" name="email"><br>
    <input type="submit" value="Kirim">
  </form>

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

    if (empty($nama)) {
      $errors[] = "Nama tidak boleh kosong.";
    }
    if (empty($email) || !filter_var($email, FILTER_VALIDATE_EMAIL)) {
      $errors[] = "Email tidak valid.";
    }

    if (!empty($errors)) {
      foreach ($errors as $error) {
        echo "<p style='color:red;'>$error</p>";
      }
    } else {
      echo "Nama: " . htmlspecialchars($nama) . "<br>";
      echo "Email: " . htmlspecialchars($email);
    }
  }
  ?>
</body>
</html>
```

Nama: yunia

Email: lala



Please include an '@' in the email address. 'lala' is missing an '@'.

PHP Form Required

- **Required** adalah atribut HTML yang digunakan untuk memastikan bahwa input dari pengguna tidak boleh kosong sebelum formulir dikirim

```
<!DOCTYPE html>
<html lang="id">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Formulir Required</title>
</head>
<body>
  <form method="POST" action="">
    <label for="nama">Nama: </label>
    <input type="text" id="nama" name="nama" required>
    <br><br>

    <label for="email">Email: </label>
    <input type="email" id="email" name="email" required>
    <br><br>

    <button type="submit">Kirim</button>
  </form>
```

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  // Mendapatkan data dari input
  $nama = trim($_POST["nama"]);
  $email = trim($_POST["email"]);

  // Validasi required di sisi server
  if (empty($nama)) {
    echo "<p>Nama wajib diisi.</p>";
  }

  if (empty($email)) {
    echo "<p>Email wajib diisi.</p>";
  }

  // Jika semua diisi
  if (!empty($nama) && !empty($email)) {
    echo "<p>Data berhasil dikirim! Nama: $nama, Email: $email</p>";
  }
}
?>
```

Nama:

Email:

Kirim  Please fill out this field.

Validasi Nama, Email dan Url

- PHP - Validasi Nama'

```
$name = test_input($_POST["name"]);  
if (!preg_match("/^[a-zA-Z-' ]*$/",$name)) {  
    $nameErr = "Only letters and white space allowed";  
}
```

❖ **Fungsi preg_match()** digunakan untuk melakukan pencocokan pola menggunakan ekspresi reguler (regular expressions) pada sebuah string.

- PHP - Validasi E-mail

```
$email = test_input($_POST["email"]);  
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {  
    $emailErr = "Invalid email format";  
}
```

❖ **Fungsi filter_var()** digunakan untuk memfilter data/input berdasarkan filter yang telah ditentukan, seperti validasi email, URL, angka, dll.

- PHP - Validasi URL

```
$website = test_input($_POST["website"]);  
if (!preg_match("/\b(?:(:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?~_]|!:,.;]*[-a-z0-9+&@#\/%?~_]/i",$website)) {  
    $websiteErr = "Invalid URL";  
}
```



DIKTISAINTEK
BERDAMPAK

KAMPUS
INOVASI

Koneksi PHP_MySQL





Koneksi ke Database

- PHP menyediakan fungsi untuk melakukan koneksi ke database dengan sejumlah fungsi untuk pengaturan baik menghubungkan maupun memutuskan koneksi dengan server database MySQL.
- Selain MySQL , PHP juga menyediakan beragam fungsi untuk melakukan hubungan dengan server database.

Koneksi Database PHP ke MySQL

Ada dua cara untuk membuat koneksi ke database MySQL dengan PHP, yaitu

1. MySQLi

MySQLi merupakan singkatan dari **MySQL Improved**. Ekstensi MySQL ini menambahkan fitur baru di interface database MySQL. MySQLi bersifat prosedural dan juga berorientasi objek.

2. PDO

PDO adalah singkatan dari **PHP Data Object**. PDO justru berorientasi hanya pada objek dan mendukung sejumlah tipe database yang menggunakan PHP, seperti MySQL, MSSQL, Informix, dan PostgreSQL.

Koneksi Menggunakan MYSQLI

```
<?php
// Konfigurasi database
$host = "localhost"; // Host database
$username = "root"; // Username database
$password = ""; // Password database
$database = "nama_database"; // Nama database

// Membuat koneksi
$conn = new mysqli($host, $username, $password, $database);

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

Koneksi Menggunakan PDO

```
<?php
try {
    $dsn = "mysql:host=localhost;dbname=nama_database";
    $username = "root";
    $password = "";

    $conn = new PDO($dsn, $username, $password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Koneksi berhasil!";
} catch (PDOException $e) {
    echo "Koneksi gagal: " . $e->getMessage();
}
?>
```

Membuat DSN (Data Source Name): DSN adalah string yang berisi informasi koneksi ke database seperti jenis database, hostname, dan nama database

Membuat Objek PDO: Objek PDO digunakan untuk mengelola koneksi ke database. Anda memerlukan nama pengguna dan kata sandi database sebagai parameter.

Menangani Error dengan Exception: Anda dapat mengatur mode error dengan menggunakan setAttribute untuk menangani kesalahan

Menguji Koneksi: Anda dapat menggunakan **try-catch** untuk memeriksa apakah koneksi berhasil atau gagal:

PDO Mendukung berbagai jenis database (MySQL, PostgreSQL, SQLite, dll.).



Koneksi Database di XAMPP

1. Jalankan XAMPP Control Panel:

- Buka **XAMPP Control** Panel di komputer Anda.
- Klik tombol Start pada modul **Apache** dan **MySQL**.

2. Akses phpMyAdmin:

- Buka browser Anda dan ketikkan URL berikut:
<http://localhost/phpmyadmin>.
- Tekan Enter, dan Anda akan diarahkan ke halaman phpMyAdmin.

3. Kelola Database:

- Di phpMyAdmin, Anda dapat membuat, mengedit, atau menghapus database sesuai kebutuhan.