

MODUL IV-FUNGSI PADA PHP

4.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Memahami kegunaan elemen-elemen dalam form
2. Mengolah data yang dimasukkan melalui form
3. Memahami metode pengiriman data melalui form
4. Memahami cara membuat validasi melalui form

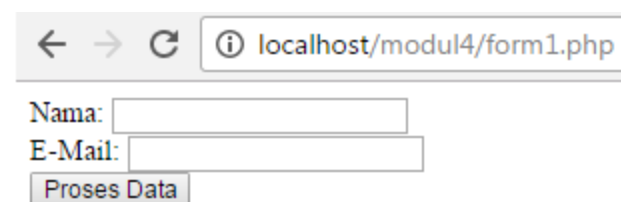
4.2 Membuat dan Memproses Form

Form adalah fitur yang sangat penting dalam sebuah website. Hampir seluruh situs modern membutuhkan form sebagai fitur utama, seperti *form pendaftaran*, *form login*, *form register peserta*, *form pembayaran* dan lain-lain.

Untuk dapat memproses data dari form, kita membutuhkan perpaduan antara kode **HTML** dengan kode **PHP**. **HTML** digunakan untuk menampilkan form, sedangkan **PHP** digunakan untuk memproses form. **CSS** dan **JavaScript** biasanya juga digunakan untuk mempercantik halaman dan memudahkan interaksi.

Berikut adalah struktur dasar form sederhana dalam HTML:

```
1 <form action="proses.php" method="get">
2     Nama: <input type="text" name="nama" />
3     <br />
4     E-Mail: <input type="text" name="email" />
5     <br />
6     <input type="submit" value="Proses Data" />
7 </form>
```



← → ↻ ⓘ localhost/modul4/form1.php

Nama:

E-Mail:

Jika anda menjalankan form HTML tersebut, akan ditampilkan form sederhana dengan 2 buah kotak inputan dan sebuah tombol “Proses Data” yang berfungsi untuk **submit** form. Dari struktur dasar tersebut, di dalam tag **<form>** terdapat 2 buah atribut. Yakni atribut **action** dan atribut **method**. Kita akan membahas kedua atribut ini secara lebih rinci.

Atribut pertama adalah **action**. Atribut **action** ini diisi dengan nilai berupa alamat halaman PHP dimana kita akan memproses isi form tersebut. Dalam contoh di atas, kita membuat nilai **action="proses.php"**, yang berarti kita harus menyediakan sebuah file dengan nama: **proses.php** untuk memproses form tersebut.

Isi atribut action sebenarnya adalah alamat dari halaman PHP. Karena atribut *action* pada contoh diatas ditulis **action="proses.php"**, maka file **proses.php** harus berada di dalam 1 folder dengan halaman HTML yang berisi form ini. Namun anda bisa dengan bebas mengubah alamat **proses.php** ini tergantung dimana file tersebut berada, misalnya menjadi alamat relatif seperti **action="file_php/proses.php"**, ataupun alamat absolut seperti: **action="http://www.domainku.com/proses.php"**.

Atribut kedua yang berkaitan dengan pemrosesan form HTML adalah atribut **method**. Atribut inilah yang akan menentukan bagaimana cara form '*dikirim*' ke dalam halaman **proses.php**. Nilai dari atribut **method** hanya bisa diisi dengan 1 dari 2 pilihan, yakni **get** atau **post**.

Jika seperti contoh diatas kita membuat nilai **method="get"**, maka nilai dari form akan dikirim melalui alamat *URL website*. Namun jika nilai method diubah menjadi **method="post"**, maka nilai form tidak akan terlihat di dalam alamat URL.

Setelah membuat tag pembuka form dengan atribut **action** dan **method**, isi form selanjutnya adalah 2 buah tag **<input type="text">** yang akan menampilkan kotak isian form. Hal yang paling penting diperhatikan adalah atribut **name** dari masing-masing tag **<input>**. Nilai dari **name** inilah yang menjadi penanda masing-masing objek form agar dapat diproses dengan PHP.

Setelah 2 buah *text input*, objek form terakhir adalah tombol **submit** yang apabila di klik akan mengirimkan data dari form ke halaman **proses.php** untuk diproses. Atribut penting disini adalah atribut **type="submit"**, yang akan otomatis mengirim isian form ketika tombol ini di klik.

4.3 Cara Mengirimkan Nilai Form ke dalam PHP

Untuk memahami cara mengirimkan nilai form ke dalam PHP, kita akan langsung praktek dengan membuat 2 buah file, yakni halaman HTML yang berisi form dengan nama file **form.html**, dan halaman PHP yang akan berisi kode untuk menampilkan hasil form dengan nama file: **proses.php**. Karena kita akan mengeksekusi kode **PHP**, kedua file ini harus dijalankan dengan **XAMPP** dan berada di dalam folder **htdocs**.

Sebagai langkah pertama, kita akan membuat file **form.php** yang berisi kode HTML sebagai berikut:

```
1 <!DOCTYPE html>
2 <head>
3   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
4   <title>Belajar Form PHP</title>
5 </head>
6
7 <body>
8   <h2>Tutorial Belajar Form HTML - PHP </h2>
9   <form action="proses.php" method="get">
10     Nama: <input type="text" name="nama" />
11     <br />
12     E-Mail: <input type="text" name="email" />
13     <br />
14     <input type="submit" value="Proses Data" />
15   </form>
16 </body>
17 </html>
```

← → ↻ ⓘ localhost/modul4/form2.php

Tutorial Belajar Form HTML - PHP

Nama:

E-Mail:

Kode HTML diatas hanya berisi struktur kode HTML sederhana dengan 1 buah form yang berisi 2 text inputan untuk **nama** dan **e-mail**.

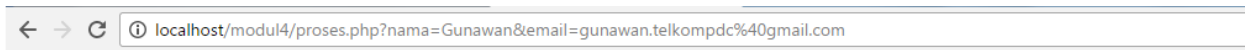
Sebelum membuat halaman **proses.php**, kita akan mencoba melakukan sedikit percobaan dengan form HTML ini. Silahkan coba input kedua kotak isian form ini dengan nilai apapun dan klik tombol **submit**. Ketika anda men-klik tombol submit, anda akan mendapati halaman error seperti berikut ini:

← → ↻ ⓘ localhost/modul4/form2.php

Tutorial Belajar Form HTML - PHP

Nama:

E-Mail:



Object not found!

The requested URL was not found on this server. The link on the [referring page](#) seems to be wrong or outdated. Please inform the author of [that page](#) about the error.

If you think this is a server error, please contact the [webmaster](#).

Error 404

[localhost](#)
Apache/2.4.17 (Win32) OpenSSL/1.0.2d PHP/5.6.23

Halaman error tersebut memberitahu kita bahwa halaman **proses.php** tidak ditemukan.

Kita bisa melihat data yang dikirim karena pada saat pembuatan form, kita menggunakan **method=get**. Namun jika anda merubah form HTML kita dengan menggunakan **method=post**, maka anda tidak akan melihat karakter-karakter ini di dalam URL. Selanjutnya kita akan mencoba menampilkan nilai ini dengan PHP pada halaman **proses.php**.

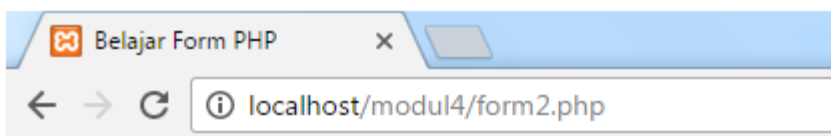
4.4 Cara Menampilkan nilai Form dengan PHP (\$_GET dan \$_POST)

Setelah membuat halaman **form.html** yang berisi form HTML, kita akan membuat halaman **proses.php** yang berisi kode PHP untuk menangani nilai dari form ini.

Silahkan buat file **proses.php** dengan kode program sebagai berikut, dan savelah di dalam folder yang sama dengan **form.php** berada:

```
1 <?php
2     echo $_GET['nama'];
3     echo "<br />";
4     echo $_GET['email'];
5 ?>
```

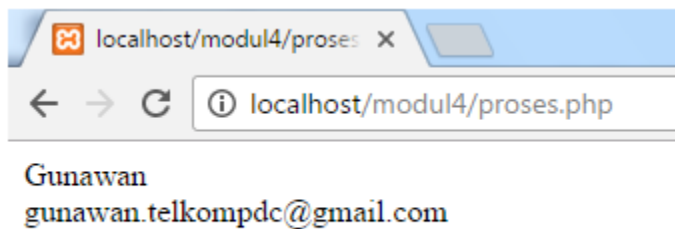
Sebelum kita membahas kode program PHP tersebut, silahkan buka kembali halaman **form.php**, isi kotak input *nama* dan *email*, lalu klik tombol *Proses Data*. Apabila tidak ada error, maka akan tampil hasil berikut ini:



Tutorial Belajar Form HTML - PHP

Nama:

E-Mail:



Tampilan di atas adalah hasil dari 3 baris kode program PHP yang kita buat di dalam halaman **proses.php**.

Untuk mengambil nilai form, PHP menyediakan 2 buah variabel global yaitu variabel `$_GET` dan `$_POST`. Kita menggunakan variabel `$_GET` jika pada saat pembuatan form menggunakan atribut **method=get**, dan menggunakan variabel `$_POST` jika form dibuat dengan **method=post**.

Kedua variabel ini sebenarnya adalah array, sehingga cara mengakses nilai dari form adalah dengan cara: `$_GET['nama_objek_form']`.

'*nama_objek_form*' adalah nilai dari atribut **name** di dalam **form**. Jika kita memiliki tag dengan kode HTML `<input type="text" name="nama" />`, maka untuk mengakses nilainya adalah dengan `$_GET['nama']`, dan untuk tag `<input type="text" name="email" />` diakses dengan nilai `$_GET['email']`.

Sebagai latihan, silahkan anda mengganti atribut *method* dalam file **form.php** menjadi:

`<form action="proses.php" method="post">`

Lalu ubah juga file **proses.php** menjadi:

```
1 <?php
2     echo $_POST['nama'];
3     echo "<br />";
4     echo $_POST['email'];
5 ?>
```

Dan PHP akan menampilkan hasil yang sama, namun kali ini form dikirim menggunakan **method=post**.

4.5 Cara Membuat Validasi Form PHP (fungsi `isset` dan `empty`)

Setelah berhasil mengambil dan menampilkan nilai dari form, hal berikutnya yang harus kita lakukan terhadap data tersebut adalah melakukan proses **validasi**. Proses validasi dilakukan terhadap nilai yang dimasukkan melalui **form** menggunakan fungsi `isset()` dan fungsi `empty()`.

Pentingnya Melakukan Validasi Nilai Form

Nilai yang telah diinput oleh user atau pengunjung web, tidak bisa begitu saja di simpan langsung ke dalam database. Karena kita tidak tahu apakah nilai tersebut telah sesuai dengan nilai yang kita

kehendaki. Misalkan apakah nilai tersebut harus berupa angka, atau hanya bisa berupa huruf, atau apakah hanya bisa diinput dalam range tertentu saja.

Dalam kasus yang *ekstrem*, seorang user bisa saja memasukkan *kode script* atau tag HTML yang bisa merusak situs kita, hal ini dikenal dengan **Cross-site Scripting**. Sebuah proses **validasi** nilai merupakan hal yang sangat penting dalam merancang form.

Dalam pembahasan validasi form ini, kita masih menggunakan contoh halaman **form.php** yang pernah kita buat pada bagian sebelumnya, berikut adalah kode HTML untuk halaman form.html:

```
<!DOCTYPE html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <title>Belajar Form PHP</title>
</head>
<body>
  <h2>Tutorial Belajar Form HTML - PHP </h2>
  <form action="proses.php" method="get">
    Nama: <input type="text" name="nama" />
    <br />
    E-Mail: <input type="text" name="email" />
    <br />
    <input type="submit" value="Proses Data" />
  </form>
</body>
</html>
```

The image shows a web browser window displaying a form. The title of the page is "Tutorial Belajar Form HTML - PHP". The form itself is simple, with a label "Nama:" followed by a text input field, and "E-Mail:" followed by another text input field. Below these fields is a button labeled "Proses Data". The form is styled with a light background and a simple border.

4.6 Memeriksa Ketersediaan Variabel Form dengan Fungsi isset()

Validasi pertama yang paling sederhana dan '*hampir*' selalu ada dalam tiap proses validasi form dalam PHP adalah memeriksa apakah objek form tersebut sudah tersedia atau tidak. Sebagai contoh sederhananya: apakah variabel `$_GET['nama']` tersedia untuk diproses atau tidak.

Proses memeriksa '*ketersediaan*' variabel ini menjadi penting karena **PHP** akan mengeluarkan pesan peringatan jika kita mengakses nilai sebuah variabel yang belum didefinisikan terlebih dahulu.

Sebagai contoh, jika kita mengakses langsung halaman **proses.php** (tanpa melalui halaman **form.php**) dan tanpa menambahkan URL, PHP akan menampilkan pesan peringatan seperti berikut ini:

```
Notice: Undefined index: nama in
D:\xampp\htdocs\belajar_form\proses.php on line 2

Notice: Undefined index: email in
D:\xampp\htdocs\belajar_form\proses.php on line 4
```

Notice: Undefined index adalah pesan error yang terjadi karena kita langsung menampilkan variabel `$_GET['nama']` dan `$_GET['email']` yang memang belum diset sebelumnya.

Untuk memeriksa apakah sebuah objek form telah didefinisikan atau telah di-set sebelumnya, kita bisa menggunakan fungsi bawaan PHP: **isset()**. Fungsi **isset()** akan menghasilkan nilai **true** jika sebuah variabel telah didefinisikan, dan **false** jika variabel tersebut belum dibuat.

Sebagai langkah antisipasi, kita akan membuat proses **validasi** untuk menangani variabel `$_GET` yang belum di-set, berikut adalah modifikasi file **proses.php**:

```
<?php
if (isset($_GET['nama']))
{
    echo $_GET['nama'];
}
```

```
echo "<br />";
```

```
if (isset($_GET['email']))
{
    echo $_GET['email'];
}
?>
```

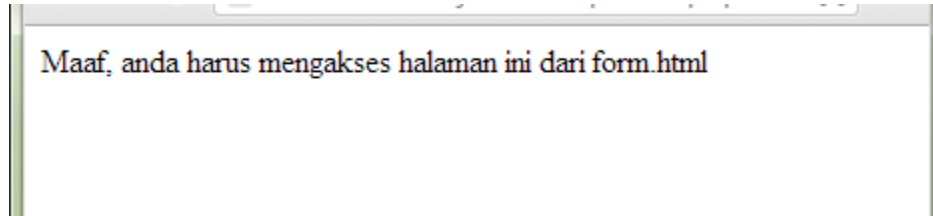
Sekarang, file **proses.php** tidak akan menghasilkan error apabila diakses tanpa melalui **form.html**. Namun perubahan kode tersebut tidak terlalu berguna karena tidak memberikan pesan error yang jelas. Berikut adalah modifikasi file **proses.php** agar lebih informatif:

```
<?php
if (isset($_GET['nama']) AND isset($_GET['email']))
{
    echo $_GET['nama'];
    echo $_GET['email'];
}
```

```

}
else
{
    echo "Maaf, anda harus mengakses halaman ini dari form.php";
}
?>

```



Pada kode PHP diatas kita mengharuskan nilai `$_GET['nama']` dan `$_GET['email']` tersedia, baru nilai ditampilkan, namun jika tidak ada, akan ditampilkan pesan bahwa halaman ini hanya bisa diakses dari **form.php**.

4.7 Memeriksa Apakah Variabel Form Telah Diisi

Fungsi `isset()` yang kita bahas sebelumnya hanya memeriksa apakah sebuah objek form ada atau tidak. Fungsi `isset()` tetap bernilai **true** meskipun user tidak mengisi form sama sekali (variabel form bernilai kosong, namun variabel tersebut dianggap telah di-set).

Untuk memeriksa apakah sebuah objek form telah diisi atau tidak, kita bisa menggunakan fungsi: `empty()`.

Fungsi `empty()` akan menghasilkan nilai **false** jika sebuah variabel telah diisi, dan bernilai **true** jika variabel tersebut belum diisi. Dengan menggunakan stuktur **IF** dan fungsi `empty()`, kita bisa membuat logika validasi objek form mana saja yang dianggap perlu (harus diisi) dan mana yang boleh dikosongkan. Dan kemudian menampilkan pesan error yang sesuai.

Sebagai contoh, kita akan memodifikasi file **proses.php** agar menampilkan pesan error jika kotak input **nama** tidak diisi. Berikut adalah kode PHP pada halaman proses.php:

```

<?php
if (isset($_GET['nama']) AND isset($_GET['email']))
{
    $nama=$_GET['nama'];
    $email=$_GET['email'];
}
else
{
    die("Maaf, anda harus mengakses halaman ini dari form.html");
}

if (!empty($nama))
{
    echo "Nama: $nama <br /> Email: $email";
}

```



```

}
else
{
    die("Maaf, anda harus mengisi nama");
}
?>

```

Dalam kode PHP diatas, kita memodifikasi beberapa bagian kode program.

Pada **logika IF** pertama, kita melakukan pengecekan apakah variabel `$_GET['nama']` dan `$_GET['email']` tersedia atau tidak. Jika tersedia maka pindahkan nilainya ke variabel `$nama` dan `$email` agar lebih mudah untuk diproses. Namun jika tidak, fungsi `die()` akan menghentikan proses dan menampilkan pesan kesalahan.

Pada logika IF kedua, kita memeriksa apakah variabel `$nama` kosong atau tidak dengan fungsi `!empty()`. Fungsi `!empty($nama)` akan menghasilkan nilai **true** hanya jika variabel `$nama` tidak kosong (perhatikan tanda **!** sebagai pembalik logika `empty()`). Namun jika `$nama` ternyata kosong (tidak diisi), maka tampilkan pesan kesalahan.



4.8 Menyeleksi Tipe Data Objek Form

Setelah objek form dipastikan tersedia, tidak kosong, validasi berikutnya yang biasanya dilakukan adalah memastikan tipe data dan range data yang diinput oleh user.

Untuk mengecek tipe data sebuah variabel, PHP menyediakan beberapa fungsi tergantung tipe datanya, yakni fungsi `is_string()`, `is_int()`, `is_float()`, `is_numeric()`, `is_bool()`, `is_array()`, dan `is_object()`. Sesuai dengan namanya, masing-masing fungsi tersebut akan mengecek tipe data dari variabel yang dites.

Diantara fungsi-fungsi diatas, fungsi `is_numeric()` mungkin butuh sedikit penjelasan. Fungsi `is_numeric()` akan mengecek apakah sebuah tipe data merupakan angka baik itu **float** atau **integer**.

Khusus objek form variabel angka seperti umur, biasanya selain menyeleksi apakah nilainya berupa angka **integer**, kita mungkin juga menambahkan aturan bahwa nilai umur harus diatas 17 tahun. Untuk menambahkan fungsi ini, fungsi `is_int()` dapat dikombinasikan dengan **struktur IF**.

Sebagai contoh kita akan menambahkan validasi untuk **tag input nama** bahwa nama tidak boleh diisi dengan angka. Untuk keperluan ini kita akan menggunakan fungsi `is_numeric()`. Berikut adalah modifikasi file proses.php:

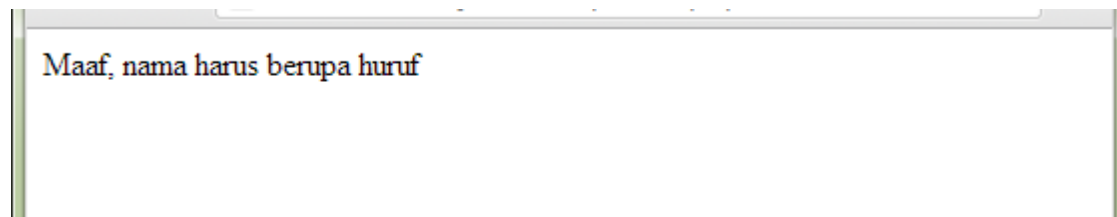
```

<?php
if (isset($_GET['nama']) AND isset($_GET['email']))
{
    $nama=$_GET['nama'];
    $email=$_GET['email'];
}
else
{
    die("Maaf, anda harus mengakses halaman ini dari form.html");
}

if(empty($nama))
{
    die("Maaf, anda harus mengisi nama");
}
else
{
    if (is_numeric($nama))
    {
        die("Maaf, nama harus berupa huruf");
    }
    else
    {
        echo "Nama: $nama <br /> Email: $email";
    }
}
?>

```

Dalam kode diatas, kita menambahkan 1 lagi **logika IF** untuk menyeleksi apakah variabel **\$nama** berisi angka numerik (*integer* atau *float*). Jika **\$nama** bertipe numerik, maka tampilkan pesan error.



Di dalam materi form PHP kali ini kita telah mempelajari cara menvalidasi nilai inputan form. Namun apa yang telah kita pelajari disini hanya sebagian kecil dari proses validasi yang sebenarnya harus dilakukan. Misalnya, untuk menfilter variabel **\$nama** diatas, akan lebih cocok menggunakan *regular expression* daripada fungsi **is_numeric()**, namun yang penting kita telah bisa '*menangkap*' cara pembuatan validasi form.