



Super Global Variabel

PHP : Form

Untuk membangun sebuah web dinamis, kita membutuhkan inputan data. Di antara inputan data yang paling dasar dalam halaman web adalah: form.

Apa itu Form?

Form merupakan sintaks HTML yang berisi kumpulan kolom isian data, misal:

- form login yang berisi isian nama pengguna dan kata sandi.
- form pendaftaran yang berisi isian nama, jenis kelamin, tanggal lahir, alamat, surel, dan lain-lain.

Dalam pembuatan web dinamis, kita bisa melakukan pengiriman data dari form HTML untuk kemudian data tersebut akan diproses lebih lanjut oleh bahasa pemrograman PHP.

Membuat Form Sederhana

Bahasa yang kita gunakan untuk membuat form untuk web dinamis adalah HTML.

1. Silakan anda membuat file dengan nama `form.php`
2. Lalu isi dengan kode program di bawah ini:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Membuat Form Sederhana</title>
7  </head>
8  <body>
9      <form>
10         <div>
11             <label>Nama</label> <br>
12             <input name="nama" type="text" placeholder="Masukkan nama">
13         </div>
14         <div>
15             <label>Alamat</label> <br>
16             <input name="alamat" type="text" placeholder="Masukkan alamat">
17         </div>
18         <div>
19             <button>Submit</button>
20         </div>
21     </form>
22 </body>
23 </html>
```



Kode program di atas akan membuat 2 buah input teks:

1. Input bertipe teks dengan *name* `nama`
2. Input bertipe teks dengan *name* `alamat`

Jika kita eksekusi file `form.php` di atas, output yang kita dapat adalah seperti berikut:



The screenshot shows a web browser window with the title "Membuat Form Sederhan". The address bar displays "localhost:1235/01-membuat-...". The page content includes two text input fields. The first field is labeled "Nama" and contains the placeholder text "Masukkan nama". The second field is labeled "Alamat" and contains the placeholder text "Masukkan alamat". Below these fields is a "Submit" button.

Menampilkan Data yang Dikirim melalui Form

Kita telah berhasil membuat sebuah form dengan 2 buah isian data. Sekarang:

1. Kita coba isi inputan nama dan inputan alamat.
2. Lalu kita klik tombol submit.
3. Setelah itu kita perhatikan url browser kita.

Misal saya isi “Hacktiv8” untuk nama, dan “Jakarta” untuk alamat.

Maka saya mendapatkan url seperti berikut:

```
localhost/form.php?nama=Hacktiv8&alamat=Jakarta
```

Bagian url setelah tanda `?` dinamakan *query string*. Dan pada pembahasan tentang *variabel bawaan PHP*, kita telah mempelajari bagaimana cara mengakses *query string* dengan variabel `$_GET`.

Untuk menampilkan data hasil inputan, ikuti langkah-langkah berikut:

1. Cari tag `</form>` dari kode di atas
2. Buat baris baru.
3. Lalu tambahkan kode program berikut:

```
21      <?php # membuka tag PHP
22          $nama = @$_GET['nama'];
23          $alamat = @$_GET['alamat'];
24          # di sini nanti kita akan tampilkan variabel $nama dan $alamat
25          # jangan lupa tutup tag PHP
26      ?>
27  </form>
```

Selanjutnya, tambahkan kode program berikut setelah komentar # di sini nanti kita akan tampilkan variabel \$nama dan \$alamat.

```
24 # di sini nanti kita akan tampilkan variabel $nama dan $alamat
25 if ($nama) {
26     echo "<strong>Nama:</strong> {$nama} <br>";
27 }
28
29 if ($alamat) {
30     echo "<strong>Alamat:</strong> {$alamat} <br>";
31 }
```

Kode program di atas akan memeriksa apakah variabel `$nama` dan `$alamat` tidak kosong. Jika memang tidak kosong alias ada isinya, maka variabel-variabel tersebut akan ditampilkan.

Perbedaan Metode GET dan POST

Terdapat beberapa metode pengiriman data dalam protokol HTTP/HTTPS. Akan tetapi yang didukung oleh HTML hanya dua saja: yaitu metode GET dan metode POST [1].

Form pada HTML secara *default* akan menggunakan metode GET untuk mengirimkan data. Seperti yang telah kita lakukan di atas.

Akan tetapi, kita bisa mengatur metode apa yang harus digunakan oleh Form untuk mengirim data dengan menambahkan atribut `method` pada tag `<form>`.

Di situ tag `<form>` akan mengirimkan data ke server dengan menggunakan metode POST.

Lalu, apa bedanya POST dan GET?

Bedanya adalah:

- Metode GET akan menampilkan semua data dalam url (yang kemudian disebut sebagai *query string*).
- Sedangkan POST, ia akan menyimpan data di dalam *body request* tanpa menampilkannya secara langsung di dalam URL.

Bayangkan jika sebuah form login yang berisi kata sandi dikirim melalui metode GET? Tentu saja kata sandi tersebut akan terekspos di dalam URL dan ini akan memudahkan peretas untuk mencuri data.



```
36 <form method="POST">
37   <div>
38     <label>Email</label> <br>
39     <input name="email" type="email" placeholder="Masukkan email">
40   </div>
41   <div>
42     <label>Kata Sandi</label> <br>
43     <input name="password" type="password" placeholder="Masukkan kata sandi">
44   </div>
45   <div>
46     <button>Login</button>
47   </div>
48 </form>
49 <?php # membuka tag PHP
50     $email = @$_POST['email'];
51     $password = @$_POST['password'];
52 # jangan lupa tutup tag PHP
53     if ($email) {
54         echo "<strong>Email:</strong> {$email} <br>";
55     }
56
57     if ($password) {
58         echo "<strong>Kata Sandi:</strong> {$password} <br>";
59     }
60     ?>
```



Penjelasan:

- Kita menambahkan atribut `method` dengan nilai `"POST"` pada tag `<form>`
- Kita menggunakan variabel `$_POST` sebagai ganti dari variabel `$_GET`



Menggunakan Variable `$_REQUEST`

Seperti yang pernah kita pelajari pada pembahasan variabel bawaan PHP. Kita bisa menggunakan variabel `$_REQUEST` untuk memanggil data yang dikirim melalui form, baik data tersebut dikirim dengan metode GET mau pun menggunakan metode POST.

Sehingga variabel `$_REQUEST` bisa menjadi pengganti dari variabel `$_GET` mau pun variabel `$_POST`

Mengirim Data ke File Yang Berbeda

Pada 2 contoh form yang telah kita buat, kita menggabungkan antara halaman input dan halaman proses. Sehingga data yang kita tampilkan pun masih berada dalam file yang sama.

Sebenarnya, kita bisa memisahkan antara halaman input dan halaman presentasi data.

Kita bisa melakukannya dengan menambahkan atribut `action` pada tag `<form>`.

Mengirim Data ke File Yang Berbeda

Pada 2 contoh form yang telah kita buat, kita menggabungkan antara halaman input dan halaman proses. Sehingga data yang kita tampilkan pun masih berada dalam file yang sama.

Sebenarnya, kita bisa memisahkan antara halaman input dan halaman presentasi data.

Kita bisa melakukannya dengan menambahkan atribut `action` pada tag `<form>`.

```
36 <form method="proses.php">
```

Isi dari atribut `action` bisa berupa nama file, dan bisa juga berupa url lengkap.



PHP : Variabel SuperGlobal

Variabel **\$_GET** dan **\$_POST** (dan juga **\$_REQUEST**) di dalam PHP termasuk ke dalam kelompok variabel yang dikenal dengan 'Variabel SuperGlobal'.

Variabel SuperGlobals adalah variabel khusus di dalam PHP yang bisa diakses dari halaman PHP manapun tanpa perlu mendefinisikannya terlebih dahulu, dan untuk mengakses variabel ini kita juga tidak perlu menggunakan keyword **global** (sebagaimana variabel global pada umumnya)

Selain variabel **\$_GET**, **\$_POST** dan **\$_REQUEST**, PHP masih memiliki beberapa variabel *superglobal* lainnya seperti **\$_COOKIE**, **\$_SESSION**, dan **\$_SERVER**. Ciri khusus untuk variabel global di dalam PHP, diawali dengan tanda **\$_**. Namun pada sesi ini kita hanya fokus kepada variabel **\$_GET**, **\$_POST** dan **\$_REQUEST**.

Variabel **\$_GET**, **\$_POST** dan **\$_REQUEST** merupakan **tipe data array**, sehingga untuk mengakses nilainya, kita menggunakan cara akses array yakni dengan menggunakan kurung siku seperti: **\$_GET['nama']** dimana **nama** adalah nilai dari atribut **name** pada objek form yang akan diakses.

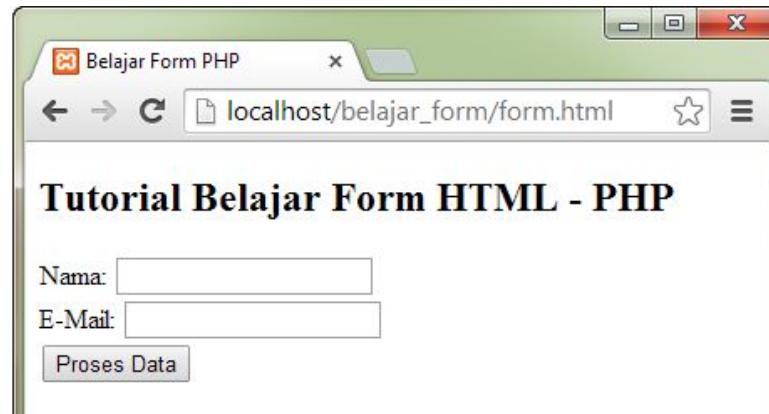
Perbedaan variabel global \$_GET, \$_POST dan \$_REQUEST

Seperti yang telah kita praktekan dalam tutorial Cara Menampilkan Hasil Form HTML dengan PHP, kita telah mengetahui bahwa jika form dikirim menggunakan **method=get** maka di dalam PHP kita mengaksesnya dengan variabel **\$_GET**, namun jika form dibuat menggunakan **method=post**, kita mengaksesnya dengan variabel **\$_POST**.

Bagaimana jika pada saat memproses form kita tidak mengetahui dengan pasti apakah form dikirim dengan **GET** atau **POST**? **PHP** menyediakan variabel **\$_REQUEST** sebagai salah satu solusinya. Variabel **\$_REQUEST** menampung nilai form yang dikirim dengan **method=get**, maupun **method=post** secara bersamaan.

Untuk mencobanya, silahkan jalankan file **form.html** dengan isi kode HTML sebagai berikut:

```
62 <!DOCTYPE html>
63 <head>
64   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
65   <title>Belajar Form PHP</title>
66 </head>
67 <body>
68   <h2>Tutorial Belajar Form HTML - PHP </h2>
69   <form action="proses.php" method="get">
70     Nama: <input type="text" name="nama" />
71     <br />
72     E-Mail: <input type="text" name="email" />
73     <br />
74     <input type="submit" value="Proses Data" />
75   </form>
76 </body>
77 </html>
```



The screenshot shows a web browser window with the title "Belajar Form PHP". The address bar displays "localhost/belajar_form/form.html". The page content includes the heading "Tutorial Belajar Form HTML - PHP" and a form with two text input fields labeled "Nama:" and "E-Mail:", followed by a "Proses Data" submit button.

Halaman **form.html** diatas persis sama dengan yang kita gunakan pada tutorial sebelumnya, namun untuk halaman **proses.php**, kita akan modifikasi dengan menggunakan variabel **\$_REQUEST**:

```
61  <?php
62      echo $_REQUEST['nama'];
63      echo "<br />";
64      echo $_REQUEST['email'];
65  ?>
```

Jika anda menjalankan **form.html** dan men-klik tombol '**Proses Data**', maka hasil form akan ditampilkan sebagaimana mestinya. Anda juga bisa mengubah method form menjadi post, dan variabel **\$_REQUEST** akan tetap menampilkan hasil form.

Jadi, variabel apa yang sebaiknya digunakan? apakah **\$_GET**, **\$_POST** atau **\$_REQUEST**? Jawabannya tergantung kepada desain kode program yang dirancang. Jika anda dapat memastikan bahwa form akan dikirim dengan **method=get**, maka gunakan variabel **\$_GET**, jika from anda menggunakan **method=post**, maka gunakan **\$_POST**, namun jika metodanya tidak dapat dipastikan, variabel superglobal **\$_REQUEST** bisa menjadi solusi.



SESSIONS

Secara sederhana session merupakan data yang disimpan di sebuah server dan dapat digunakan / diakses secara global di

Lalu bagaimana contoh penggunaan session ?

Contoh penggunaan session yang sering digunakan adalah digunakan untuk pembuatan fitur login, session digunakan untuk menyimpan data user yang sedang login, sehingga jika ada halaman pada aplikasi yang mengharuskan pengguna login, anda hanya perlu login sekali, dan data login tersebut akan disimpan di session, data session ini yang akan diperiksa oleh setiap halaman yang memerlukan autentikasi login user, contohnya pada aplikasi social media atau email anda.

Memulai Session di PHP

Untuk memulai Session di PHP anda dapat menggunakan function `session_start()`.

Baik dalam contoh kita akan membuat sebuah file dengan nama `set_session.php`, lalu tuliskan skrip seperti dibawah ini :

```
73  <?php
74  session_start();
75  $_SESSION["username"] = "administrator";
76  $_SESSION["password"] = "12345678";
77  ?>
```



HACKTIV8

Keterangan :

- Pada Line 2 kita menuliskan function `session_start()`; function ini digunakan untuk melakukan start pada session
- Variabel session telah diset sebagai variabel global di PHP yaitu `$_SESSION`, sehingga untuk membuat session kita perlu menyimpannya di variabel `$_SESSION`.
- Pada Line 3 kita menuliskan `$_SESSION["username"] = "administrator"`; yang artinya kita akan membuat session dengan nama "username" dengan nilai "administrator"
- Pada Line 4 kita menuliskan `$_SESSION["password"] = "12345678"`; yang artinya kita akan membuat session dengan nama "password" dengan nilai "12345678"
- sehingga saat file `set_session.php` dijalankan akan maka akan membuat session dengan nama username dengan nilai 'administrator' dan session dengan nama password dengan nilai '12345678', session tersebut disimpan di server.



COOKIE

Cookie merupakan sebuah file text yang berisi data tertentu yang disimpan didalam browser.

Secara sederhana peran cookie ini hampir mirip dengan session, hanya saja kalau session itu disimpan di server, tapi kalau cookie ini disimpan di browser

Contoh dari penggunaan Cookie

Coba perhatikan login pada facebook, atau web – web lain, biasanya terdapat checklist untuk mengingat password (Remember Me), nah jika ada menchecklist bagian itu, informasi akun anda akan disimpan di cookie browser, sehingga ketika anda membuka web tersebut akan otomatis melakukan login, meskipun browser sudah diclose sebelumnya.

Berbeda halnya jika anda tidak menchecklist bagian remember me, informasi akun anda hanya disimpan session, sehingga ketika browser di close dan anda masuk ke browser ulang untuk mengakses web, maka anda diharuskan untuk login ulang, karena data yang disimpan di session otomatis di hapus ketika browser pengguna (client) di tutup.

Bagaimana sudah paham tentang peran dari cookie ?

Jika sudah paham kita lanjut ke pembahasan dari cookie.



HACKTIV8

Baik kita akan coba membuat perintah cookie, silahkan buat file dengan nama **set_cookie.php**, lalu isinya adalah seperti berikut ini :

```
80  <?php
81  setcookie('username', 'administrator', time() + (60 * 60 * 24 * 5), '/');
82  setcookie('nama', 'Hacktiv8', time() + (60 * 60 * 24 * 15), '/');
83  ?>
```

Keterangan :

- Coba Jalankan skrip di file set_cookie.php
- Baris 2 digunakan untuk membuat cookie dengan nama username, dengan nilai 'administrator' dengan waktu expire cookie selama 5 hari, dan dapat diakses diseluruh path dari direktori tempat menyimpan file php disimpan.
- Baris 3 digunakan untuk membuat cookie dengan nama nama, dengan nilai 'Budi Nurcahya' dengan waktu expire cookie selama 15 hari, dan dapat diakses diseluruh path dari direktori tempat menyimpan file php disimpan.

Perbedaan Cookie dan Session

Sebelum kita membahas lebih detail mengenai cookie, kita akan bahas dulu apa perbedaan dari cookie dan session, perbedaannya antara lain :

Cookie

1. Cookie disimpan di sisi klien, lebih tepatnya di browser dari pengguna aplikasi
2. Penggunaan Cookie tidak aman bagian klien, karena cookie disimpan di sisi klien (browser) sehingga memungkinkan klien (pengguna aplikasi) dapat menghapus mengedit serta melakukan disabled pada cookie.
3. Data yang disimpan di dalam cookie dapat disimpan lebih lama, karena waktu expired dari cookie dapat diatur.

Session

Session disimpan di sisi server

Penyimpanan data melalui session lebih aman karena data disimpan di sisi server, tidak seperti cookie yang datanya dimasukkan di sisi klien (browser)

Data Session otomatis terhapus ketika web browser klien dimatikan (diclose), selain itu data session tidak dapat diatur waktu expired.