

LAPORAN PRAKTIKUM
PEMROGRAMAN WEB & MOBILE I



NAMA : ETRILIAN TONGALU
NIM : E1E118003
KELAS : C
MODUL : II

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PALANGKA RAYA
2021

BAB I

TUJUAN DAN LANDASAN TEORI

1. Tujuan Praktikum

- 1.1. Mahasiswa mampu membuat handling yang mampu mengolah data dari form HTML.
- 1.2. Mahasiswa mampu membuat batasan-batasan untuk menangani inputan dari form HTML.

2. Landasan Teori

Variabel superglobal PHP `$_GET` dan `$_POST` digunakan untuk mengumpulkan data-form. Contoh berikut menunjukkan form HTML sederhana dengan dua field input dan tombol submit:

```
<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>
```

Gambar 1.1 PHP

Ketika user mengisi form, dan menekan tombol click, data form dikirim untuk memproses file PHP dengan nama “welcome.php”. Data form dikirimkan dengan method HTTP POST. Untuk menampilkan data yang sudah disubmit bisa dilakukan dengan mencetak data tersebut menggunakan perintah echo. File “welcome.php” adalah sebagai berikut:

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

Gambar 1.2 PHP

Jika field nama diinputkan dengan Tono dan email diinputkan dengan tono@mail.com maka output yang akan tampil adalah sebagai berikut:

Welcome Budi

Your email address is tono@mail.com

Hasil yang sama juga akan tampil dengan menggunakan method get sebagai berikut:

```
<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>
```

Gambar 1.3 PHP

dengan file “welcome_get.php” sebagai berikut:

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

Gambar 1.4 PHP

GET vs. POST

GET dan POST membuat sebuah array (contoh array(kunci => nilai, kunci2 => nilai2, kunci3 => nilai3, ...)). Array ini menyimpan pasangan kunci/nilai, dimana kunci- kunci adalah nama-nama dari form control dan nilai- nilai adalah data input dari user. Method GET diakses menggunakan \$_GET dan method POST diakses menggunakan \$_POST. Kedua variabel ini adalah variabel superglobal, yang selalu bisa diakses, tanpa memperhatikan lingkup dan bisa diakses dari fungsi, class atau file yang berbeda tanpa harus melakukan teknik khusus. \$_GET adalah sebuah array dari variabel yang dikirimkan ke skrip melalui parameter URL. \$_POST adalah sebuah array dari

variabel yang dikirimkan ke skrip melalui method HTTP POST.

Kapan sebaiknya menggunakan GET?

Informasi dikirim dari sebuah form dengan method GET bisa dilihat oleh semua orang (semua nama dan nilai variabel ditampilkan di URL). GET juga memiliki batas pada jumlah informasi yang dikirim. Batasannya adalah sekitar 2000 karakter. Namun, karena variabel ditunjukkan di URL, ia memungkinkan untuk dilakukan bookmark halaman. Dalam beberapa kasus, hal ini sangat bermanfaat. GET bisa digunakan untuk mengirimkan data yang tidak sensitif.

Ingat! GET tidak boleh digunakan untuk mengirimkan password atau informasi

Kapan menggunakan POST?

Informasi yang dikirim dari sebuah form dengan method POST tidak bisa dilihat oleh siapapun (semua nama-nama atau nilai-nilai tertanam didalam body request HTTP) dan tidak memiliki batasan jumlah informasi yang akan dikirim. POST juga mendukung fungsionalitas lanjutan seperti dukungan untuk input biner multi-part ketika sedang melakukan upload file ke server. Namun, karena variabel tidak ditampilkan di URL, tidak mungkin untuk dilakukan bookmark halaman (data tidak ter-bookmark). Developer lebih baik menggunakan POST untuk mengirimkan data form.

Validasi Form PHP

Pertimbangkan keamanan ketika memproses form PHP!

PHP Form Validation Example

* required field.

Name: *

E-mail: *

Website:

Comment:

Gender: ☐ Female ☐ Male *

Gambar 1.5 PHP Form

Form HTML yang akan kita gunakan pada modul ini, mengandung bermacam- macam field input, misalnya text field yang harus diisi dan text field yang opsional, tombol pilihan (radio button), dan tombol submit. Rule atau aturan validasi untuk form diatas adalah sebagai berikut:

Field	Rule Validasi
Name	Dibutuhkan. + Harus hanya mengandung huruf dan spasi
E-mail	Dibutuhkan. + Harus mengandung sebuah alamat email yang valid dengan @ dan .
Website	Opsional. Jika ada, harus mengandung URL yang valid.
Comment	Opsional. Field input multi-line (text area).
Gender	Dibutuhkan. Harus memilih salah satu

Kode HTML untuk membentuk Form tersebut adalah sebagai berikut:

Text Field

Field nama, email dan website adalah elemen-elemen text input, dan field komentar adalah textarea yaitu sebagai berikut:

Name: `<input type="text" name="name">`

E-mail: `<input type="text" name="email">`

Website: `<input type="text" name="website">`

Comment: `<textarea name="comment" rows="5" cols="40"></textarea>`

Radio Button

Field jenis kelamin adalah radio button yaitu sebagai berikut:

Gender:

```
<input type="radio" name="gender" value="female">Female
```

```
<input type="radio" name="gender" value="male">Male
```

Form Element

Kode HTML untuk membentuk form pada gambar diatas adalah sebagai berikut:

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

Ketika form disubmit, data pada form dikirim dengan method “post”. **\$_SERVER[“PHP_SELF”]** adalah variabel super global yang mengembalikan nama file dari skrip yang sedang dieksekusi. Sehingga kode form diatas mengirim data pada form ke halaman itu sendiri. Sedangkan fungsi **htmlspecialchars()** adalah fungsi yang mengkonversikan karakter-karakter spesial ke entitas HTML. Sebagai contoh, fungsi tersebut akan mengkonversikan karakter < dan > menjadi < dan >. Fungsi ini mencegah injeksi yang bisa dilakukan dengan HTML atau javascript (Cross-site Scripting Attack) pada form tersebut.

Catatan Penting pada Keamanan Form PHP

Variabel **\$_SERVER[“PHP_SELF”]** bisa digunakan oleh hacker! Jika **PHP_SELF** digunakan pada halaman web, user bisa memasukkan skrip dengan terlebih dahulu memasukkan garis miring (/) kemudian beberapa perintah Cross Site Scripting (XSS) untuk dieksekusi. XSS adalah tipe kelemahan keamanan komputer yang secara tipikal ditemukan dalam aplikasi web.

Asumsikan kita memiliki halaman web dengan nama “test_form.php”, dan form hanya kita deklarasikan sebagai berikut:

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

Kemudian user memasukkan URL pada address bar dengan alamat sebagai berikut:

[http://localhost/<nama_folder>/test_form.php/%22%3E%3Cscript%3Ealert\('hacked'\)%3C/scr ipt %3E](http://localhost/<nama_folder>/test_form.php/%22%3E%3Cscript%3Ealert('hacked')%3C/scr ipt %3E)

yang jika ditranslasikan akan menjadi:

```
<form method="post" action="test_form.php/"><script>alert('hacked')
</script>
```

Kode ini menambah tag script dan perintah alert atau peringatan, ketika halaman dibuka, kode javascript tersebut akan dieksekusi, maka user akan melihat kotak peringatan dengan tulisan “hacked”.

Berhati-hatilah dengan kemungkinan penambahan kode javascript pada tag <script>!

Hacker bisa mengarahkan user ke file pada server yang lain, dan file itu bisa mengandung kode yang bisa merubah variabel global atau melakukan submit form pada alamat web yang berbeda untuk mencuri data user.

Bagaimana menghindari penyalahgunaan \$_SERVER[“PHP_SELF”]?

Caranya adalah dengan menggunakan fungsi htmlspecialchars(). Fungsi tersebut akan mengkonversikan karakter khusus ke entitas HTML. Ketika user memasukkan URL dengan tag script seperti contoh sebelumnya, maka akan ditranslasikan sebagai berikut:

```
<form          method="post"          action="test_form.php/&quot;&gt;&lt;sc
ript&gt;alert('hacked')&lt;/script&gt;">
```

dengan cara ini, percobaan penyalahgunaan akan gagal.

Memvalidasi Data Form dengan PHP

Hal pertama yang akan kita lakukan adalah memasukkan semua variabel melalui fungsi htmlspecialchars(). Kemudian ada juga dua hal ketika user melakukan submit form:

1. Membuang karakter-karakter yang tidak dibutuhkan (seperti spasi extra, tab extra, dan baris baru yang ekstra) dari data input user (dengan fungsi trim()).

2. Membuang backslash (\) satu garis miring dari data input user (dengan fungsi stripslashes()).

Langkah berikutnya adalah membuat fungsi yang akan melakukan pemeriksaan kebenaran data yang diinputkan oleh user. Contohnya adalah sebagai berikut:

```
<?php
// define variables and set to empty values
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $name = test_input($_POST["name"]);
    $email = test_input($_POST["email"]);
    $website = test_input($_POST["website"]);
    $comment = test_input($_POST["comment"]);
    $gender = test_input($_POST["gender"]);
}

function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}

?>
```

Gambar 1.6 Validasi Data

Ingat bahwa pada permulaan skrip, adalah pemeriksaan apakah form sudah disubmit menggunakan `$_SERVER["REQUEST_METHOD"]`. Jika `REQUEST_METHOD` adalah `POST`, maka form telah disubmit dan seharusnya tervalidasi. Jika belum tersubmit, lewati langkah validasi dan tampilkan form kosong. Namun pada contoh diatas semua field input adalah opsional. Skrip bekerja baik bahkan jika user tidak melakukan entri data.

Field yang Dibutuhkan

Kode program berikut terdapat tambahan variabel baru yaitu: `$nameErr`, `$emailErr`, `$genderErr`. Variabel-variabel error ini akan menangani pesan error untuk field yang dibutuhkan. Percabangan dengan `if else` juga akan

ditambahkan untuk setiap variabel \$_POST. Fungsinya untuk memeriksa apakah variabel \$_POST kosong, hal ini dilakukan dengan menggunakan fungsi empty(). Jika kosong, maka pesan error disimpan dalam variabel error yang berbeda, dan jika tidak kosong, ia akan mengirim data input user melalui fungsi test_input():

```
<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST")
{
    if (empty($_POST["name"])) {
        $nameErr = "Name is required";
    } else {
        $name = test_input($_POST["name"]);
    }

    if (empty($_POST["email"])) {
        $emailErr = "Email is required";
    } else {
        $email = test_input($_POST["email"]);
    }

    if (empty($_POST["website"])) {
        $website = "";
    } else {
        $website = test_input($_POST["website"]);
    }

    if (empty($_POST["comment"])) {
        $comment = "";
    } else {
        $comment = test_input($_POST["comment"]);
    }

    if (empty($_POST["gender"])) {
        $genderErr = "Gender is required";
    } else {
        $gender = test_input($_POST["gender"]);
    }
}
?>
```

Gambar 1.7 Field

Setelah kode diatas ditambahkan, beberapa skrip ditambahkan pada setiap field yang dibutuhkan pada form, fungsinya untuk menampilkan pesan

error jika field yang dibutuhkan tidak diisi. Form HTMLnya adalah sebagai berikut:

```
<form method="post" action="<?php echo  
htmlspecialchars($_SERVER["PHP_SELF"]);?>">  
  
    Name: <input type="text" name="name">  
    <span class="error">* <?php echo  
    $nameErr;?></span> <br><br>  
    E-mail:  
    <input type="text" name="email">  
    <span class="error">* <?php echo $emailErr;?></span>  
    <br><br>  
    Website:  
    <input type="text" name="website">  
    <span class="error"><?php echo $websiteErr;?></span>  
    <br><br>  
    Comment: <textarea name="comment" rows="5" cols="40"></textarea>  
    <br><br>  
    Gender:  
    <input type="radio" name="gender" value="female">Female  
    <input type="radio" name="gender" value="male">Male  
    <span class="error">* <?php echo $genderErr;?></span>  
    <br><br>  
    <input type="submit" name="submit" value="Submit">  
  
</form>
```

Gambar 1.8 Field

Validasi Nama

Kode berikut menunjukkan cara sederhana untuk memeriksa apakah field nama hanya mengandung huruf dan spasi. Jika nilai dari nama tidak valid, maka pesan error akan disimpan didalam variabel \$nameErr:

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

Gambar 1.9 Validasi Nama

Fungsi preg_match() mencari string berdasarkan pola, mengembalikan nilai true jika polanya ada, false jika polanya tidak ada.

Validasi Email

Cara paling mudah dan paling aman untuk memeriksa apakah sebuah alamat email memiliki pola yang sesuai adalah dengan menggunakan fungsi `filter_var()`. Kode dibawah memeriksa apakah alamat email yang dimasukkan menggunakan pola yang sesuai atau tidak, jika tidak, maka pesan error akan disimpan kedalam variabel `$emailErr`:

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

Gambar 1.10 Validasi Email

Validasi URL

Kode program berikut menunjukkan cara untuk memeriksa apakah sintaks alamat URL valid atau tidak. Ekspresi reguler ini mengizinkan keberadaan tanda pisah pada URL. Jika sintaks alamat URL tidak valid, maka pesan error akan disimpan kedalam variabel `$websiteErr`:

```
$website = test_input($_POST["website"]);  
if (!preg_match("/^b(?:(:?https?|ftp):\\//|www\\.)[-a-z09+&@#/%?~_!:,;]*[-a-z0-9+&@#/%?~_]/i",$website)) {  
    $websiteErr = "Invalid URL";  
}
```

Biasanya, jika user salah menginputkan nilai, maka halaman yang tampil adalah halaman yang sama dengan field yang sudah terisi dengan nilai field yang sudah diinput sebelumnya. Untuk menunjukkan nilai dalam field input setelah user menekan tombol submit, ada beberapa skrip PHP yang perlu ditambahkan didalam atribut value pada field input name, email, dan website. Khusus untuk field textarea, akan skrip tersebut akan ditambahkan antara tag `<textarea>` dan tag `</textarea>`. Skrip yang singkat akan mengeluarkan nilai dari variabel `$name`, `$email`, `$website` dan `$comment`. Untuk radio button atau tombol radio, akan ditambahkan kode yang membuat salah satu pilihan terpilih.

```
Name: <input type="text" name="name" value="<?php echo $name;?>">

E-mail: <input type="text" name="email" value="<?php echo $email;?>">

Website: <input type="text" name="website" value="<?php echo $website;?>">

Comment: <textarea name="comment" rows="5" cols="40"><?php echo
$comment;? ></textarea>

Gender:
<input type="radio" name="gender"
<?php if (isset($gender) && $gender=="female") echo
"checked";?> value="female">Female
<input type="radio" name="gender"
<?php if (isset($gender) && $gender=="male") echo
"checked";?> value="male">Male
```

Gambar 1.11 Validasi URL

BAB II

PEMBAHASAN

Buatlah program web untuk menginputkan username dan password menggunakan form dan penanganan input data dengan kriteria sebagai berikut:

1. Username yang diinputkan tidak boleh lebih dari tujuh karakter.
2. Password yang diinputkan harus terdiri dari huruf kapital, huruf kecil, angka dan karakter khusus.
3. Jumlah karakter password tidak boleh kurang dari sepuluh karakter.

Dari kriteria tugas diatas berikut adalah kode program untuk tugas diatas :

```
<?php
$username = "";
$password = "";
$usernameErr = "";
$passwordErr = "";
$valid_panjang_uname = true;
$valid_panjang_uname_msg = "";
$valid_panjang_pass = true;
$valid_panjang_pass_msg = "";
$pass_valid = true;
$pass_valid_msg = "";
$success = "";

if (isset($_POST['submit'])) {
    $username = trim($_POST['username']);
    $password = trim($_POST['password']);

    if (empty($username)) {
        $usernameErr = "Username masih kosong<br>";
    }
    if (strlen($username) > 7) {
        $valid_panjang_uname = false;
        $valid_panjang_uname_msg = "Username maksimal 7 karakter<br>";
    }

    if (empty($password)) {
        $passwordErr = "Password masih kosong<br>";
    }
}
```

```

    } else {
        $kapital = preg_match("@[A-Z]@", $pass);
        $kecil = preg_match("@[a-z]@", $pass);
        $angka = preg_match("@[0-9]@", $pass);
        $karakter = preg_match("@[^\w]@", $pass);

        if (strlen($pass) < 10) {
            $valid_panjang_pass = false;
            $valid_panjang_pass_msg = "Password minimal 10 karakter.
            <br>";
        }
        if (!$kapital || !$kecil || !$angka || !$karakter) {
            $pass_valid = false;
            $pass_valid_msg = "Password terdiri dari
            huruf KAPITAL, huruf kecil, angka dan karakter khusus.<b
            r>";
        }
    }

    if (!empty($username) and !empty($pass) and $valid_panjang_uname
    and $valid_panjang_pass and $pass_valid) {
        $username = "";
        echo "<script>alert('Selamat, Anda Berhasil Mengisi Form den
        gan Benar')</script>";
    }
}
?>

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
    <title>Tugas Modul 2</title>
</head>

<body>
    <div style="margin: 0 auto; width: 300px;">
        <h3>Form</h3>
        <form action="index2.php" method="post">
            Username : <input type="text" name="username" value="<?=
            $username ?>"><br>

```

```

        <small style="color: red;"><?php echo $usernameErr .
        $valid_panjang_uname_msg; ?></small>
        <br>
        Password : <input type="password" name="pass"><br>
        <small style="color: red;"><?php echo $passErr . $valid_
        panjang_pass_msg . $pass_valid_msg; ?></small>
        <br><input type="submit" name="submit" value="Submit">
    </form>
</div>
</body>

</html>

```

Sebelum membahas code-code yang akan digunakan berdasarkan kriteria tugas, kita membuat terlebih dahulu variable-variabel yang akan dibutuhkan untuk code selanjutnya. Berikut adalah variable-variabel tersebut

```

$username = "";
$pass = "";
$usernameErr = "";
$passErr = "";
$valid_panjang_uname = true;
$valid_panjang_uname_msg = "";
$valid_panjang_pass = true;
$valid_panjang_pass_msg = "";
$pass_valid = true;
$pass_valid_msg = "";
$success = "";

```

Langkah selanjutnya yaitu membuat form yang berisi input username dan password terlebih dahulu. Code yang dibuat yaitu:

```

<form action="index.php" method="post">
    Username : <input type="text" name="username" value="<?= $user
    name ?>"><br>
    <small style="color: red;"><?php echo $usernameErr . $valid_pa
    njang_uname_msg; ?></small>
    <br>
    Password : <input type="password" name="pass"><br>
    <small style="color: red;"><?php echo $passErr . $valid_panjan
    g_pass_msg . $pass_valid_msg; ?></small>
    <br><input type="submit" name="submit" value="Submit">
</form>

```

Berdasarkan code di atas, tag <small> digunakan untuk memperlihatkan pesan error jika ada error dalam penginputan data. Sedangkan dalam tag <form> berisi action =index.php, berarti setelah klik submit maka kita tetap berada di halaman yang sama, yaitu index.php. Sedangkan method yang digunakan yaitu POST.

Untuk kriteria yang pertama, kita akan membuat aturan ketika menginput username, yaitu karakter yang dimasukkan maksimal 7 karakter. Code yang digunakan yaitu:

```
if (strlen($username) > 7) {  
    $valid_panjang_uname = false;  
    $valid_panjang_uname_msg = "Username maksimal 7 karakter<br>";  
}
```

Pertama-tama, jumlah huruf dari string variable \$username dihitung terlebih dahulu menggunakan *strlen*. Jika username lebih dari 7 karakter maka akan menjalankan code di dalamnya, yaitu variabel \$valid_panjang_uname yang sebelumnya bernilai true menjadi false dan \$valid_panjang_uname_msg yang sebelumnya kosong menjadi berisi pesan. Pesan tersebut berisi petunjuk berdasarkan kesalahan yang dilakukan dalam penginputan data.

Kemudian untuk tugas yang kedua, kita diminta untuk menambahkan aturan dalam penginputan password. Aturan tersebut yaitu password yang diinputkan harus terdiri dari huruf kapital, huruf kecil, angka dan karakter khusus . Berikut adalah code untuk membuat program tersebut.

```
$kapital = preg_match("@[A-Z]@", $pass);  
$kecil = preg_match("@[a-z]@", $pass);  
$angka = preg_match("@[0-9]@", $pass);  
$karakter = preg_match("@[^\w]@", $pass);  
  
if (!$kapital || !$kecil || !$angka || !$karakter) {  
    $pass_valid = false;  
    $pass_valid_msg = "Password terdiri dari minimal satu  
    huruf KAPITAL, huruf kecil, angka dan karakter khusus.  
    <br>";  
}
```

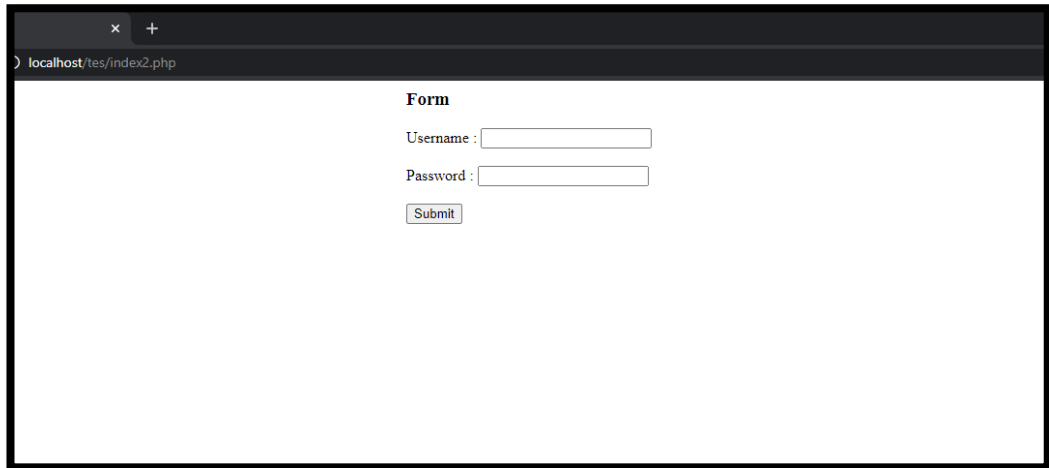

Berdasarkan code di atas, digunakan `preg_match` untuk mencari pola tertentu dalam sebuah string. Pola yang dimasukkan ke dalam fungsi `preg_match()` disesuaikan berdasarkan masing-masing variable yang dibuat, yaitu `$kapital` untuk mencari huruf kapital (A-Z), `$kecil` untuk mencari huruf kecil (a-z), `$angka` untuk mencari angka (0-9), dan `$karakter` untuk mencari karakter khusus (`^\w`). Lalu menggunakan kondisi `if else` untuk mengetahui apakah string yang dimaksud (password) tidak mengandung huruf kapital, huruf kecil, angka, dan karakter. Jika bernilai `TRUE` maka `$pass_valid_msg` akan berisi pesan petunjuk. Jika bernilai `False` maka tidak terjadi apapun.

Selanjutnya, pada kriteria yang terakhir diperintahkan untuk menambahkan aturan dalam penginputan password. Aturan tersebut yaitu jumlah karakter password yang dimasukkan minimal sepuluh karakter. Code yang dibuat yaitu sebagai berikut:

```
if (strlen($pass) < 10) {  
    $valid_panjang_pass = false;  
    $valid_panjang_pass_msg = "Password minimal 10 karakter.  
    <br>";  
}
```

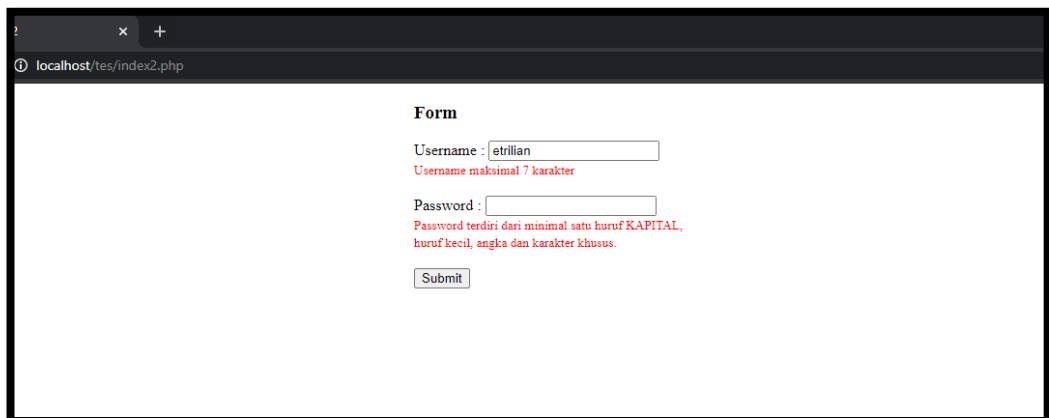
Untuk penjelasannya hampir sama seperti code pada kriteria pertama, yaitu diawali dengan menghitung jumlah huruf dari string variable `$pass` menggunakan `strlen`. Jika password kurang dari 10 karakter maka akan menjalankan code di dalamnya, yaitu variabel `$valid_panjang_pass` yang sebelumnya bernilai `true` menjadi `false` dan `$valid_panjang_pass_msg` yang sebelumnya kosong menjadi berisi pesan. Pesan tersebut berisi petunjuk berdasarkan kesalahan yang dilakukan dalam penginputan data, yaitu "Password minimal 10 karakter".

Output dari program ini jika dijalankan adalah sebagai berikut.



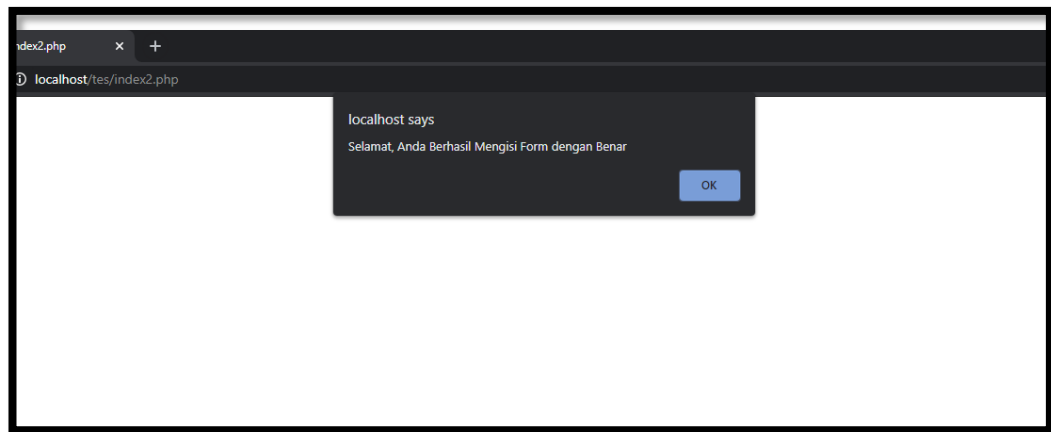
A screenshot of a web browser window. The address bar shows 'localhost/tes/index2.php'. The page content is titled 'Form' and contains two input fields: 'Username : ' and 'Password : '. Below the password field is a 'Submit' button.

Gambar 2.1 Output tampilan di localhost



A screenshot of the same web browser window. The 'Username' field now contains the text 'ettrilian'. Below it, a red error message reads 'Username maksimal 7 karakter'. The 'Password' field is empty, and below it, a red error message reads 'Password terdiri dari minimal satu huruf KAPITAL, huruf kecil, angka dan karakter khusus.' The 'Submit' button remains at the bottom.

Gambar 2.2 Output jika username dan password yang dimasukkan salah



Gambar 2.3 Output jika berhasil mengisi form dengan benar

KESIMPULAN

POST adalah proses mengirim data (submit) melalui form yang berasal dari (X)HTML. Karena PHP adalah bahasa program atau *scripting*, maka PHP bisa digunakan untuk menerima data hasil submit tersebut. Data yang diterima nantinya bisa diproses dalam script PHP.

Untuk menerima data dari proses submit form dalam PHP biasanya gunakan *statement assignment* `$namaVariabel = $_POST['nama komponen dalam form'];`. Berbeda dengan GET, yaitu proses mengirim data (submit) melalui URL. Karena data yang memakai method GET dikirim melalui URL, maka informasi yang bersifat penting atau pribadi lebih baik dikirim menggunakan method POST.

Sedangkan validasi data merupakan proses yang sangat penting dalam sebuah program yang mengharuskan user untuk mengisi form. Hal tersebut penting karena kesalahan atau error dalam program sering disebabkan karena input data yang salah. Maka dari itu, validasi data tetap diperlukan dalam sebuah form agar si developer mampu meminimalisir error dari data yang dimasukkan dan user mengetahui kesalahan dari data yang dimasukkan.

DAFTAR PUSTAKA

Praktikum, K. (n.d.). *MODUL PRAKTIKUM PEMROGRAMAN WEB I Jurusan Teknik Informatika Fakultas Teknik Universitas Palangka Raya*.

LAMPIRAN

