

LAPORAN HASIL PRAKTIKUM
PEMROGRAMAN WEB DAN MOBILE I



Nama : Andy Saputra
NIM : 193030503052
Kelas : A
Modul : II (Form Handling)

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PALANGKA RAYA

2021

BAB I

TUJUAN DAN LANDASAN TEORI

1.1. Tujuan

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

1.2. 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>
```

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"];
```

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>
```

dengan file “welcome_get.php” sebagai berikut:

```
<html>

    <body>

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

1.2.1. 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 sensitif lainnya!

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 *

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="na
me">E-
```

mail:

```
<input
type="text"
"
```

```
name="e
mail">
```

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_SE  
LF"]);? >">
```

Ketika form disubmit, data pada form dikirim dengan method “post”.

- 1.2.2. **\$_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/script%3E](http://localhost/<nama_folder>/test_form.php/%22%3E%3Cscript%3Ealert('hacked')%3C/script%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

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;script&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;

    }

```

?>

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 =  
    }  
?>
```

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;?></s
```

```
pan> <br><br>E-
```

```
mail:
```

```
<input type="text" name="email">
```

```
<span class="error">* <?php echo $emailErr;?></span>
```

```
<
```

```
b
```

```
r
```

```
>
```

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

```
<span class="error">* <?php echo $genderErr;?></span>
```

```
<br><br>
```

```
<input type="submit" name="submit" value="Submit">
```

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

```
$name = test_input($_POST["name"]);
```

```
if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
```

```
    $nameErr = "Only letters and white space allowed";
```

\$nameErr:

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";
```

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`:

Biasanya, jika user salah memasukkan 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 fieldinput 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:

BAB II

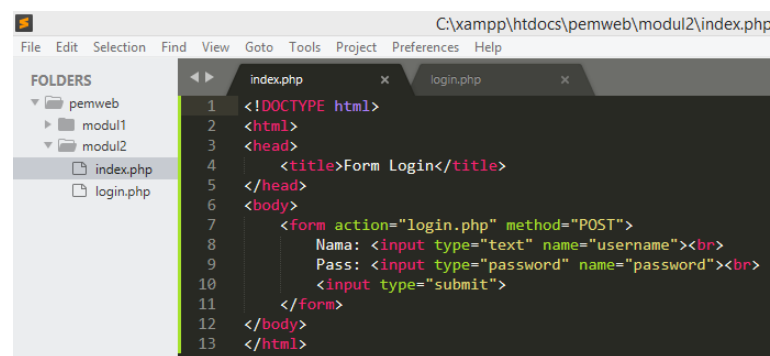
PEMBAHASAN

2.1. 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, angkadan karakter khusus.
3. Jumlah karakter password tidak boleh kurang dari sepuluh karakter.

Dari tugas dan kriteria diatas berikut source code dan pembahasannya :

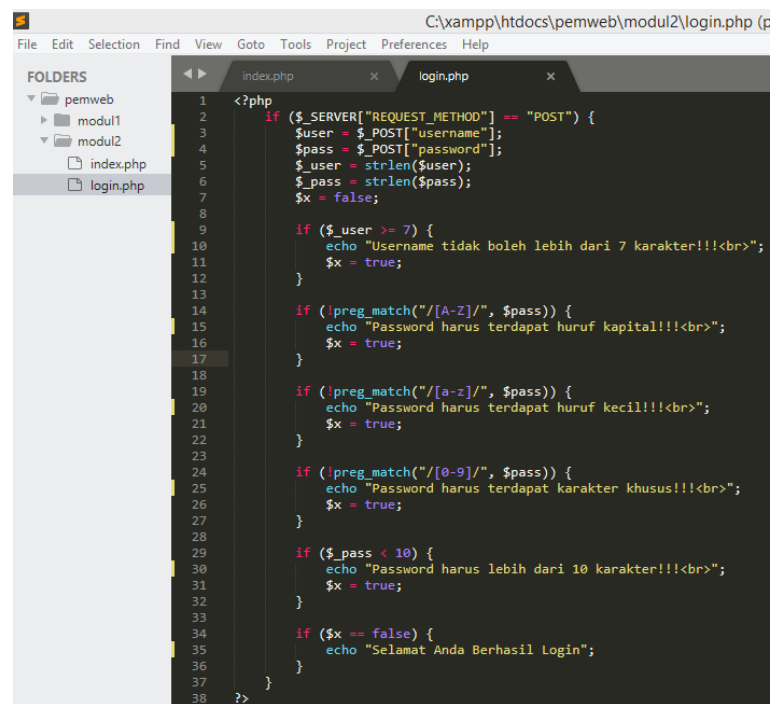


```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Form Login</title>
5 </head>
6 <body>
7 <form action="login.php" method="POST">
8   Nama: <input type="text" name="username"><br>
9   Pass: <input type="password" name="password"><br>
10  <input type="submit">
11 </form>
12 </body>
13 </html>
```

Gambar 2.1 Source code HTML.

Pada gambar 2.1 diatas merupakan source code html. “<!DOCTYPE html>” berfungsi untuk mendeklarasikan kepada komputer bahwa Anda akan menuliskan perintah dalam kode HTML. “<html></html>” merupakan tag yang menandakan bahwa Anda memulai dan mengakhiri dokumen dalam kode HTML “<head><head>” diisi dengan metadata dari dokumen HTML. Seperti judul tab dengan kita menuliskan “<title>Form Login</title>”. “<body></body>” diisi dengan konten halaman website. “form action="login.php" method="POST">” source code tersebut berguna untuk

menjelaskan kemana nanti data yang ada pada form dikirimkan dan dari atribut ini data akan dikirimkan ke halaman login.php serta menggunakan metode POST. “<input type="text" name="username">
” source code tersebut berguna untuk membuat form dengan tipe masukannya teks dan nama nya username serta untuk form password menggunakan tipe masukannya password. “<input type="submit">” source code tersebut untuk membuat tombol submit.



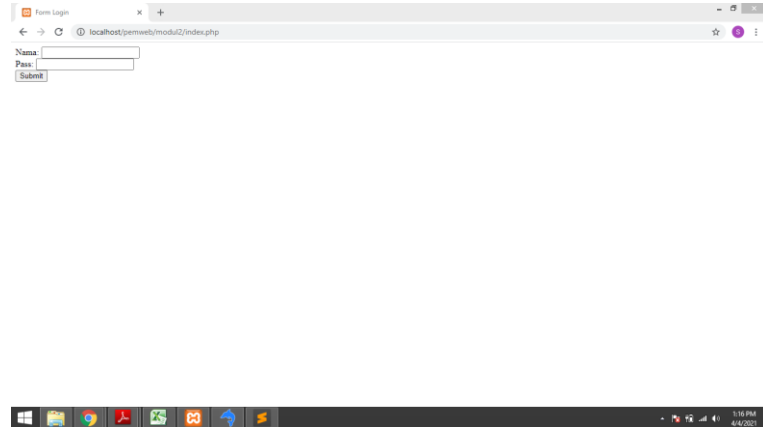
```
1 <?php
2 if ($_SERVER["REQUEST_METHOD"] == "POST") {
3     $user = $_POST["username"];
4     $pass = $_POST["password"];
5     $user = strlen($user);
6     $pass = strlen($pass);
7     $x = false;
8
9     if ($user >= 7) {
10         echo "Username tidak boleh lebih dari 7 karakter!!!<br>";
11         $x = true;
12     }
13
14     if (!preg_match("/[A-Z]/", $pass)) {
15         echo "Password harus terdapat huruf kapital!!!<br>";
16         $x = true;
17     }
18
19     if (!preg_match("/[a-z]/", $pass)) {
20         echo "Password harus terdapat huruf kecil!!!<br>";
21         $x = true;
22     }
23
24     if (!preg_match("/[0-9]/", $pass)) {
25         echo "Password harus terdapat karakter khusus!!!<br>";
26         $x = true;
27     }
28
29     if ($pass < 10) {
30         echo "Password harus lebih dari 10 karakter!!!<br>";
31         $x = true;
32     }
33
34     if ($x == false) {
35         echo "Selamat Anda Berhasil Login";
36     }
37 }
38 ?>
```

Gambar 2.2 Source code PHP.

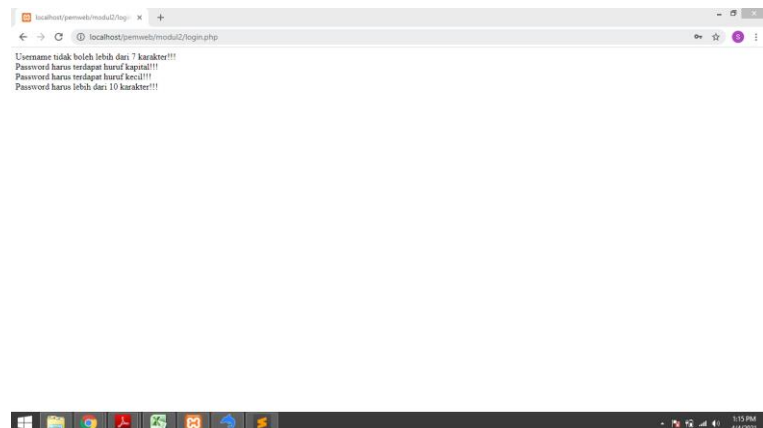
Pada gambar 2.2 diatas merupakan bagian php, untuk mendeklarasikan php kita cukup menggunakan <?php?>. “if (\$_SERVER["REQUEST_METHOD"] == "POST") {” source code tersebut berfungsi untuk mengecek bahwa terdapat form yang disumbit. “\$user = \$_POST["username"];” source code tersebut merupakan data metode POST yang disimpan didalam variable yang berbentuk array yang kemudian disimpan kedalam variable user. “\$user = strlen(\$user);” source code tersebut berfungsi untuk menghitung panjang string dan disimpan pada variable user. “\$x = false;” source code tersebut berfungsi sebagai nilai pada

percabangan. “if (\$_user >= 7) {” source code tersebut merupakan kondisi dimana jika nilai yang tersimpan pada variable user lebih sama dengan 7 maka akan melakukan pernyataan yang ada pada kondisi tersebut.

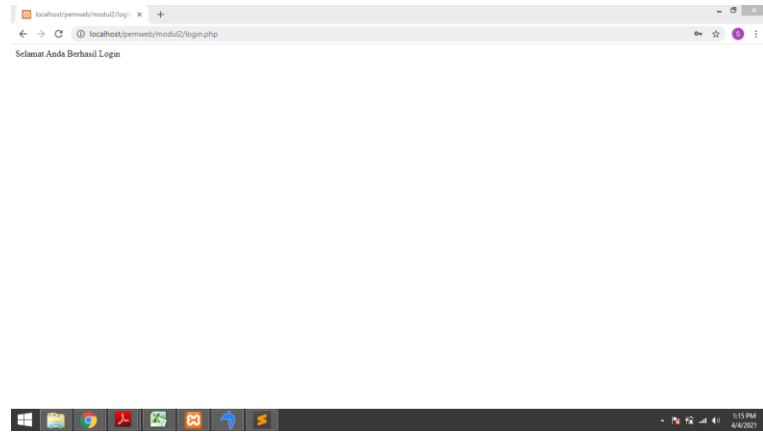
Dari pembahasan diatas berikut adalah hasil website form handling :



Gambar 2.3 Form Login.



Gambar 2.4. Handling.



Gambar 2.5 Handling.

BAB III

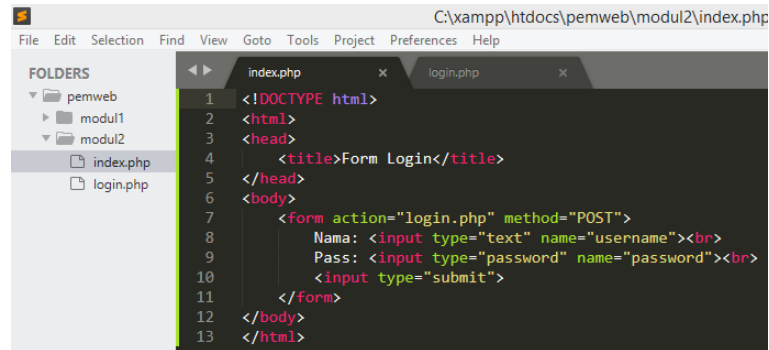
KESIMPULAN

Dari hasil praktikum yang sudah dilaksanakan dapat saya tarik kesimpulan bahwa Form Handling adalah mekanisme untuk menangani suatu masukan dari sebuah form dan terdapat array untuk menyimpan data dari form. Terdapat 3 array yaitu GET, POST dan REQUEST.

DAFTAR PUSTAKA

Modul Praktikum Pemrograman Web & Mobile I. Jurusan Teknik Informatika.
Fakultas Teknik. Universitas Palangka Raya. 2021.

LAMPIRAN

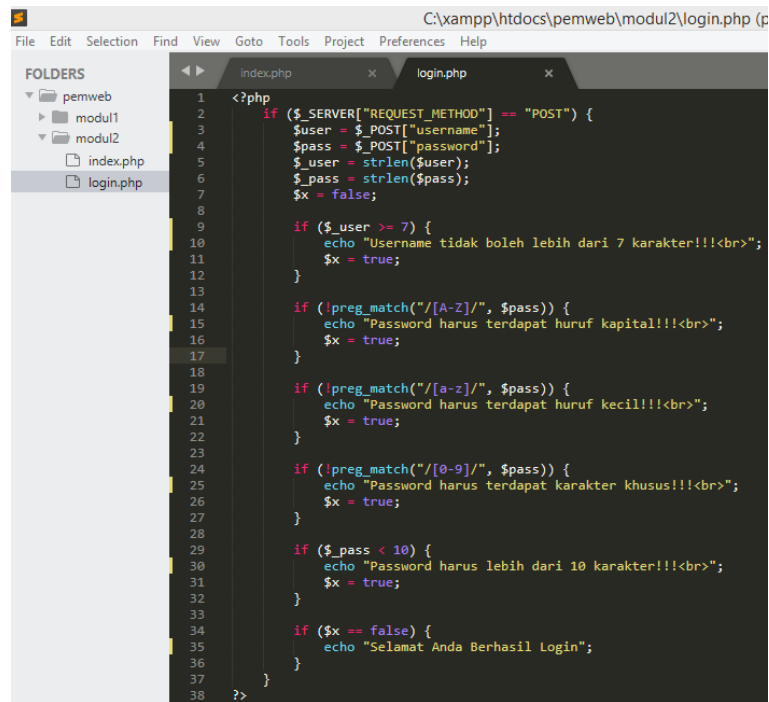


```
C:\xampp\htdocs\pemweb\modul2\index.php
File Edit Selection Find View Goto Tools Project Preferences Help

FOLDERS
  pemweb
    modul1
    modul2
      index.php
      login.php

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Form Login</title>
5 </head>
6 <body>
7   <form action="login.php" method="POST">
8     Nama: <input type="text" name="username"><br>
9     Pass: <input type="password" name="password"><br>
10    <input type="submit">
11  </form>
12 </body>
13 </html>
```

Gambar 2.1 Source code HTML.

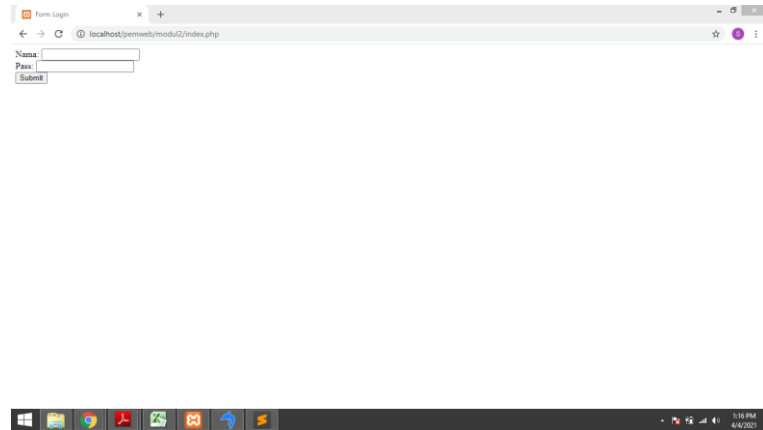


```
C:\xampp\htdocs\pemweb\modul2\login.php (p
File Edit Selection Find View Goto Tools Project Preferences Help

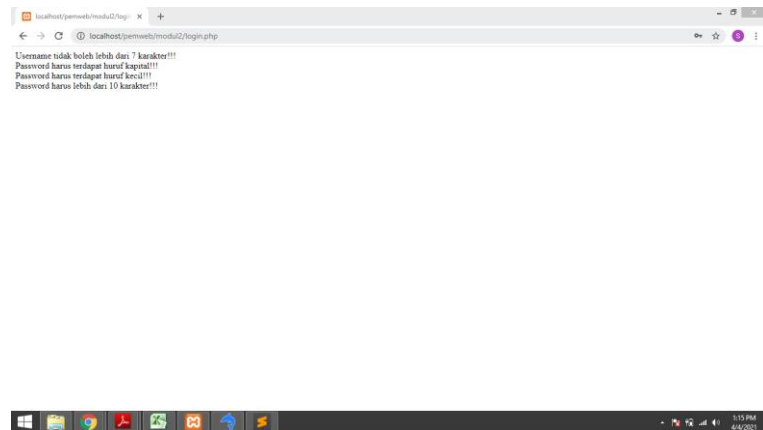
FOLDERS
  pemweb
    modul1
    modul2
      index.php
      login.php

1 <?php
2 if ($_SERVER["REQUEST_METHOD"] == "POST") {
3   $user = $_POST["username"];
4   $pass = $_POST["password"];
5   $user = strlen($user);
6   $pass = strlen($pass);
7   $x = false;
8
9   if ($user >= 7) {
10    echo "Username tidak boleh lebih dari 7 karakter!!!<br>";
11    $x = true;
12   }
13
14   if (!preg_match("/[A-Z]/", $pass)) {
15    echo "Password harus terdapat huruf kapital!!!<br>";
16    $x = true;
17   }
18
19   if (!preg_match("/[a-z]/", $pass)) {
20    echo "Password harus terdapat huruf kecil!!!<br>";
21    $x = true;
22   }
23
24   if (!preg_match("/[0-9]/", $pass)) {
25    echo "Password harus terdapat karakter khusus!!!<br>";
26    $x = true;
27   }
28
29   if ($pass < 10) {
30    echo "Password harus lebih dari 10 karakter!!!<br>";
31    $x = true;
32   }
33
34   if ($x == false) {
35    echo "Selamat Anda Berhasil Login";
36   }
37 }
38 ?>
```

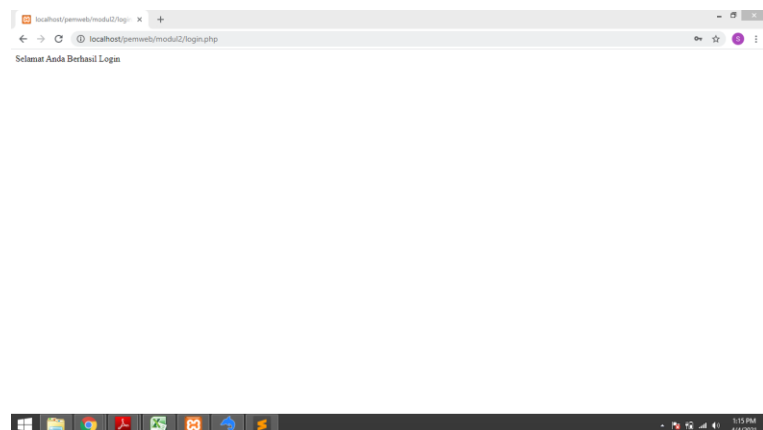
Gambar 2.2 Source code PHP.



Gambar 2.3 Form Login.



Gambar 2.4. Handling.



Gambar 2.5 Handling.