Pemrograman Web

15. User Authentication, Form Validation, Paging.

M. Udin Harun Al Rasyid, S.Kom, Ph.D
http://lecturer.eepis-its.edu/~udinharun
udinharun@eepis-its.edu

User Authentication

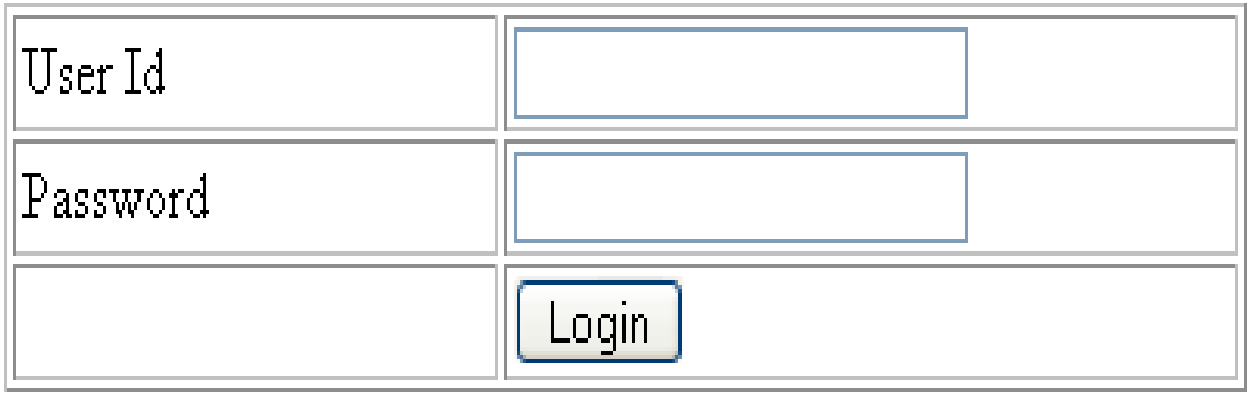
Authentication method

- Basic authentication
- User & password stored in database

Basic authentication

- We hard code the username and password combination in the login script itself. Suitable for simple application.
- With this basic authentication method we store the user information (user id and password) directly in the script.
- This is only good if the application only have one user since adding more user means we must also add the new user id and password in the script.

Example



Code login.php

- See login.pdf

- Make sure that the form method is set to **post** since we certainly don't want to show up the user id and password in the address bar.
- We simply check if the user id and password exist in $_POST and check if these two match the hardcoded user id and password.

- If the submitted user id and password match these two then we set the value of $_SESSION['basic_is_logged_in'] to true. After that we move the application's main page. In this case it's called main.php

- If the user id and password don't match we set the error message. This message will be shown on top of the login form.

Checking if the user is logged in or not

- Since the application main page, main.php, can only be accessed by those who already authenticated themselves we must check that before displaying the page.
- $_SESSION['basic_is_logged_in'] is set or not.

- If $_SESSION['basic_is_logged_in'] is set and it's value is true then we can continue showing the rest of the page.

Code main.php

```
<?php
// like i said, we must never forget to start the session
session_start();

// is the one accessing this page logged in or not?
if (!isset($_SESSION['basic_is_logged_in']) || $_SESSION['basic_is_logged_in'] !== true) {
    // not logged in, move to login page
    header('Location: login.php');
    exit;
}

?>
<html>
<head>
<title>Main User Page</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body>
<p>This is the main application page. You are free to play around here since you
 are an autenthicated user :-) </p>
<p> </p>
<p><a href="logout.php">Logout</a> </p>
</body>
</html>
```

The Logout Script

- check if $_SESSION['basic_is_logged_in'] is already set or not and check whether it's value is true. Using this information we can build the logout script to simply unset this session **or** set the session value to false.

Code logout.php

```
?php
// i will keep yelling this
// DON'T FORGET TO START THE SESSION !!!
session_start();

// if the user is logged in, unset the session
if (isset($_SESSION['basic_is_logged_in'])) {
    unset($_SESSION['basic_is_logged_in']);
}

// now that the user is logged out,
// go to login page
header('Location: login.php');
?>
```

Login Using Database

- Buat database user → tbl_auth_user

```
CREATE TABLE `tbl_auth_user` (
  `user_id` varchar(10) NOT NULL default '',
  `user_password` varchar(32) NOT NULL default '',
  PRIMARY KEY  (`user_id`)
);
```

Contoh script di file PDF

- Session-login.pdf

Form Validation

- Whenever you make a form you should not leave it alone without any form validation. Why?
- Because there is no guarantee that the input is correct and processing incorrect input values can make your application give unpredictable result.

- You can validate the form input on two places, **client side** and **server side**.

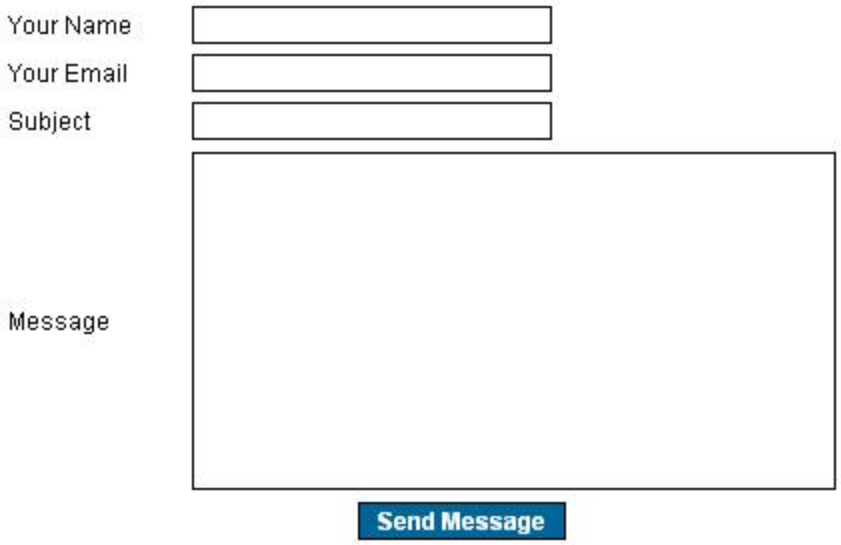
- Client side form validation usually done with javascript.
- Client side validation makes your web application respond **'faster'** while server side form validation with PHP can act as a **backup** just in case the user switch off javascript support on her browser.

- And since different browsers can behave differently there is always a possibility that the browser didn't execute the javascript code as you intended.

Some things you need to check :

- empty values
- numbers only
- input length
- email address

Example : contact form



Source Code Contact Form Validation

- See formvalidation.doc

This contact form requires four input :

- sender name
- sender email
- message subject
- message body

- First let's focus on the client side validation. On the "Send Message" button I put this javascript code : onClick="return checkForm();", which is triggered when you click on it.

- Clicking the button will run the function checkForm().

- Every input is checked to see whether they are valid input.
- When an invalid input is found the function returns false so the form is not submitted
- When you insert valid input the function will return true and the form is submitted.

- To access the value of the name input box we use cname.value.
- The name values is trimmed to remove extra spaces from the beginning and end of the name.
- If you do not enter your name or only entering spaces then an alert box will pop up.

- Using cname.focus() the cursor will be placed to the name input box and then checkForm() return false which cancel the form submit.

- The code above uses trim() function.
- This is not a built in javascript function.
- Anyway it's not a big deal because we can just make our own trim() function.
- The solution here uses **regular expression** to replace any spaces in the beginning and end of a string with blank string.

```
function trim(str)
{
   return str.replace(/^\s+|\s+$/g,'');
}
```

So in english the search replace function above can be read as :

- "Replace one or more whitespace character from the beginning or ending of a string with blank character"

As for the email input, we need to double check it.

- First, check if the email is entered and second check if the input is in a valid email format.
- For the second check well use isEmail() function.
- This function also uses regular expression.

A valid email format can be described as :

- [a string consisting of alphanumeric characters, underscores, dots or dash] @ ([a valid domain name] DOT [a valid TLD]) OR [a valid IP adress]
- In case you're wondering TLD means Top Level Domain such as com, net, org, biz, etc.

- Finally, if all input are considered valid checkForm() returns true and the form will be submitted.
- This will set the $_POST['send'] variable and now we start validating the input on the server side using PHP.

Validating using PHP

```php
<?php

$errmsg  = ''; // error message
$sname   = ''; // sender's name
$email   = ''; // sender's email addres
$subject = ''; // message subject
$message = ''; // the message itself

if(isset($_POST['send']))
{
   $sname   = $_POST['sname'];
   $email   = $_POST['email'];
   $subject = $_POST['subject'];
   $message = $_POST['message'];

   if(trim($sname) == '')
   {
      $errmsg = 'Please enter your name';
   }
   else if(trim($email) == '')
   {
      $errmsg = 'Please enter your email address';
   }
   else if(!isEmail($email))
   {
      $errmsg = 'Your email address is not valid';
   }
   else if(trim($subject) == '')
   {
      $errmsg = 'Please enter message subject';
   }
   else if(trim($message) == '')
   {
      $errmsg = 'Please enter your message';
   }

// ... more code here
?>
```

- The PHP validation is doing the same thing as the javascript validation.
- It check each value to see if it's empty and if it is we consider that as an error. We also recheck the validity of the email address.

- When we find an error we set the value of $errmsg.
- We will print this value so the user can fix the error.
- If everything is okay the value of $errmsg will be blank. So we continue processing the input.

```php
<?php

// ... previous validation code

if($errmsg == '')
{
   if(get_magic_quotes_gpc())
   {
      $subject = stripslashes($subject);
      $message = stripslashes($message);
   }

   $to      = "email@yourdomain.com";
   $subject = '[Contact] : ' . $subject;
   $msg     = "From : $sname \r\n " . $message;
   mail($to,
        $subject,
        $msg,
        "From: $email\r\nReturn-Path: $email\r\n");

// ... more code here
?>
```

- After finishing all that boring job of validating the input we finally come to the last, and the most important step, sending the message using the mail() function.

- The first parameter we pass to the mail() function is the receiver's email address.
- The second is the email subject.
- The third is the message itself and the fourth is an additional headers.

Paging

- Paging means showing your query result in **multiple pages** instead of just put them all in one long page
- Imagine waiting for five minutes just to load a search page that shows 1000 result.
- By splitting the result in multiple pages you can save download time plus you don't have much scrolling to do.

- To show the result of a query in several pages first you need to know **how many rows you have** and **how many rows per page** you want to show. For example if I have 295 rows and I show 30 rows per page that mean I'll have ten pages (rounded up).

Example

- a table named randoms that store 295 random numbers. Each page shows 20 numbers.

- When paging.php is called for the first time the value of $_GET['page'] is not set. This caused $pageNum value to remain 1 and the query is :
  SELECT val FROM randoms LIMIT 0, 20

- which returns the first 20 values from the table. But when paging.php is called like this
- http://localhost/paging/paging.php?page=4 the value of $pageNum becomes 4 and the query will be :
  SELECT val FROM randoms LIMIT 60, 20

- After showing the values we need to print the links to show any pages we like. But first we have to count the number of pages. This is achieved by dividing the number of total rows by the number of rows to show per page : $maxPage = ceil($numrows/$rowsPerPage);

- The mathematical function ceil() is used to round up the value of $numrows/$rowsPerPage.
- In this case the value of total rows $numrows is 295 and $rowsPerPage is 20 so the result of the division is 14.75 and by using ceil() we get $maxPage = 15

- Now that we know how many pages we have we make a loop to print the link.
- Each link will look something like this:
5

FINISH