UNIT – IV

USER DATA INPUT THROUGH FORMS

PHP Form Handling

When you are designing the web page, that accepts the input from the user and process the information entered by the user, at that time you need to design a from in your application.

In PHP, you can design the form which contains various input controls that allows user to input information using the concept of form element.

Syntax:
<form>

//code here

</form >

Two important attributes of FROM element are:

1.Method : This attribute specify which method will be used by the form to transfer the contents of the various controls to the specified file.

–It can have one of the two values : GET or POST

2.Action : This attribute specify the name of the file to which the contents of the various controls are sent for the processing.

Thus the general syntax of Form element should be like:

<form method = "GET/POST" action = "filename">
</form >

INPUT Elements

Once you define the Form in your PHP page, your next task is to place various controls that accepts input from the user in different ways.

INPUT element allows you to place various controls in the Form to accept input from the user.

INPUT element must be contained in the FORM element as shown below:

<FORM method = "GET/POST" action = "filename">
<INPUT type = "text" name = "username">
<INPUT type = "password" name = "password">

</FORM>
Following are the important attributes that are used with INPUT element.

1.Name : This attribute specify the name of the input control. The name is used to retrieve the value of the input control in the destination file.

2.Type : This attribute specify the type of the input control. You can specify different values for this attribute as:

Type	**Control**
Text	Text Box
Password	Password Field
Checkbox	Check Box
Radio	Radio Button
Hidden	Hidden Field
Submit	Submit Button

3.Value : This attribute specify the value associated with the input control.

TextBox

Textbox can be used to enter single line of text.

Password

Password field used to enter password. When user enter any text within this control, it is displayed in the form of asterisk (*).

Checkbox

checkbox allows a user to select more than one option from given option.

Radiobutton

Radiobutton allows a user to select any one option from given option.

HiddenField

Hidden Field contains a value but it is not visible to the user.

Submit button

Every form requires a submit button. This is the element that causes the browser to send the names and values of the INPUT elements to the file specified by the ACTION attribute of the FORM element.

TextArea

TextArea can be used to enter multiline text. It is useful form entering address, Feedback etc.

List Box

<SELECT> element is used to display list of the options, from which anyone option can be selected.

Example:

Designing File: reg.php

```
<form name="frm1"  method="post" action="process_reg.php">
<table>
        <tr>
         <td>Name:</td>
           <td><input type="text" name="ntxt" placeholder="Enter Name"/></td>
        </tr>
        <tr>
         <td>Gender:</td>
           <td> <input type="radio" name="g" value="male" />Male
                  <input type="radio" name="g" value="female" />Female</td>
        </tr>
        </tr>
         <tr>
         <td>Department:</td>
           <td> <select name="dtxt">
                          <option>Select department</option>
                 <option>Computer Engineering</option>
                 <option>Civil Engineering</option>
           </select></td>
        </tr>
         <tr>
```

```
 <td>Phone no:</td>
 <td><input type="tel" name="phtxt" placeholder="Enter Phone Number"  /></td>
</tr>
 <tr>
 <td>Password:</td>
   <td> <input type="password" name="ptxt" placeholder="Enter Your Password"
 /></td>
</tr>
<tr>
 <td>Confirm password:</td>
   <td> <input type="password" name="cptxt" placeholder="Re-Enter Password" /></td>
</tr>
<tr>
 <td>E-mail id:</td>
   <td> <input type="email" name="emtxt" placeholder="Enter Your E-mail id" /></td>
</tr>
<tr>
<td></td>
 <td><input type="submit" name="submit" value="Register"></td>
</tr>
</table>
<input type==”hidden” name=”hiddenVal” value=”111”/>
</form>
```

Coding File: process_reg.php

```
<?php

 $name=$_POST["ntxt"];
 echo $name.”<br>”;
 $gen=$_POST["g"];
 echo $gen.”<br>”;
 $dept=$_POST["dtxt"];
 echo $dept.”<br>”;
 $phnum=$_POST["phtxt"];
 echo $phnum.”<br>”;
 $pwd=$_POST["ptxt"];
 echo $pwd.”<br>”;
 $cpwd=$_POST["cptxt"];
```

```
        $eid=$_POST["emtxt"];
        echo $eid.”<br>”;
?>
```

Get Vs. Post Vs. Request

Using GET Method:

When The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the ? character.

http://www.test.com/index.htm?name1=value1&name2=value2

• The GET method produces a long string that appears in your server logs, in the browser's Location: box.
• The GET method is restricted to send upto 1024 characters only.
• Never use GET method if you have password or other sensitive information to be sent to the server.
• GET can't be used to send binary data, like images or word documents, to the server.
• The data sent by GET method can be accessed using QUERY_STRING environment variable.
• The PHP provides $_GET associative array to access all the sent information using GET method.

Using POST Method:

The POST method transfers information via HTTP headers. The information is encoded as described in case of GET method and put into a header called QUERY_STRING.

• The POST method does not have any restriction on data size to be sent.

• The POST method can be used to send ASCII as well as binary data.

• The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP (https) you can make sure that your information is secure.

• The PHP provides $_POST associative array to access all the sent information using GET method.

Using REQUEST Method:

$_REQUEST is a super global variable which is widely used to collect data after submitting html forms.

Combining HTML & PHP in Single Page

In order to develop an application that accepts input from user, process that input and display the output, we generally need to design two separate files.

•One file that contains code to design FORM that accepts input from user and another file that contains code to process the input.

•It is also possible to combine FORM designing code and FORM processing code on single file.

Example:

```
<HTML>
<HEAD>
<TITLE>Form Example</TITLE>
</HEAD>
<BODY>
<FORM method = "POST" action="#" >
Enter Your Full Name: <INPUT TYPE="TEXT" NAME="fname" >
<BR>
<INPUT TYPE="SUBMIT" NAME="submit" VALUE="SUBMIT">
</FORM>
<?php
      if(!isset($_POST['fname']))
      {
            $msg= "";
      }
      else
      {
            $name = $_POST['fname'];
            $msg= "Welcome: " . $name;
      }

      echo $msg;
?>
</BODY>
</HTML>
```

Note: here we cab also write **<?php echo $_SERVER["PHP_SELF"]; ?>** instead of # in form's action attribute.

Cookies

A cookie is a small piece of information that is stored on the client computer, either in the browser's memory or in a file on the hard disk.

A cookie having name and value.

A cookie is useful for storing user specific information such as username, password, last visit etc.

Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

Create Cookies With PHP

A cookie is created with the `setcookie()` function.

Syntax

```
setcookie(name, value, expire);
```

Only the *name* parameter is required. All other parameters are optional.

Create/Retrieve a Cookie

The following example creates a cookie named "user" with the value "User 1". The cookie will expire after 30 days (86400 * 30).

We then retrieve the value of the cookie "user" (using the global variable $_COOKIE). We also use the `isset()`function to find out if the cookie is set:

```
<?php
$cookie_name = "user";
$cookie_value = "User 1";
setcookie($cookie_name, $cookie_value, time() + (60*2)); // 2 minutes
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
```

```
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

Modify a Cookie Value

To modify a cookie, just set (again) the cookie using the `setcookie()` function.

Delete a Cookie

To delete a cookie, use the `setcookie()` function with an expiration date in the past or empty value for the name of cookie to be deleted:

```
<?php
// set the expiration date to one hour ago
setcookie("user", "", time() - 3600);

//or set the value empty
setcookie("user", "");

?>
<html>
<body>

<?php
echo "Cookie 'user' is deleted.";
?>

</body>
</html>
```

Check if Cookies are Enabled

The following example creates a small script that checks whether cookies are enabled.

```
<?php
setcookie("test_cookie", "test", time() + 120);
?>
```

```php
<html>
<body>

<?php
if(count($_COOKIE) > 0) {
    echo "Cookies are enabled.";
} else {
    echo "Cookies are disabled.";
}
?>

</body>
</html>
```

Sessions

A session is a way to store information (in variables) to be used across multiple pages.

Unlike a cookie, the information is not stored on the users computer.

When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.

So; Session variables hold information about one single user, and are available to all pages in one application.

Start a PHP Session

A session is started with the `session_start()` function.

Session variables are set with the PHP global variable: $_SESSION.

Example

create a new page called "**demo_session1.php**". In this page, we start a new PHP session and set some session variables:

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
```

```
echo "Session variables are set.";
?>

</body>
</html>
```

Get PHP Session Variable Values

Next, we create another page called "demo_session2.php". From this page, we will access the session information we set on the first page ("demo_session1.php").

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>
</body>
</html>
```

Another way to show all the session variable values for a user session is to run the following code:

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>

</body>
</html>
```

Modify a PHP Session Variable

To change a session variable, just overwrite it:

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// to change a session variable, just overwrite it
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>

</body>
</html>
```

Destroy a PHP Session

To remove all global session variables and destroy the session, use `session_unset()` and `session_destroy()`

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// remove all session variables
session_unset();

// destroy the session
session_destroy();
?>
</body>
</html>
```

Managing User Preferences with session

you can use sessions to manage your users' preferences when they visit your site. In this three-step example, you'll start a session, ask a user for her font family and base font size preferences, display those preferences on subsequent pages, and allow the user to change her mind and reset the values.

Starting a Session and Registering Defaults

In this script, you'll start a session and register the font_family and font_size variables. The displayed HTML will be a form that allows you to change your preferences.

1. Open a new file in your text editor, start a PHP block, and call the session_start() function:

```
<?php
  session_start();
```

2. Start an [if...else] block to check for any previous values for font_family and font_size. If values are not present in the current session, assign default values and add them:

```
if ((!$_SESSION[font_family]) || (!$_SESSION[font_size])) {
    $font_family = "sans-serif";
    $font_size = "10";
    $_SESSION[font_family] = $font_family;
    $_SESSION[font_size] = $font_size;
```

3. If previous values do exist, extract the values from the $_SESSION superglobal:

```
} else {
    $font_family = $_SESSION[font_family];
    $font_size = $_SESSION[font_size];
}

?>
```

The user will come back to this script to reset her display preferences, you have to take into account the fact that the values of the variables must always be extracted from the session itself. If you simply added the variables to a session without checking for previous values, each time the page was loaded, the value of these variables would be overwritten as an empty string.