Unit-5 Form Handling

- **What is Form?**
- Forms are used to get input from the user and submit it to the web server for processing.
- A form is an HTML tag that contains graphical user interface items such as input box, check boxes radio buttons etc.
- The form is defined using the <form>...</form> tags and GUI items are defined using form elements such as input.

When and why we are using forms?

- Forms come in handy when developing flexible and dynamic applications that accept user input.
- Forms can be used to edit already existing data from the database
- Form submission type POST or GET.
- Submission URL that will process the submitted data
- Input fields such as input boxes, text areas, buttons,checkboxes etc.

```
<html>
<head>
 <title>Registration Form</title>
<meta http-equiv="Content-Type" content="text/html;
    charset=UTF-8">
 </head>
 <body>
 <h2>Registration Form</h2>
 <form action="registration_form.php" method="POST">
 First name: <input type="text" name="firstname"> <br>
 Last name: <input type="text" name="lastname">
<input type="hidden" name="form_submitted" value="1"
    />
<input type="submit" value="Submit"> </form> </body>
    </html>
```

- <form…>…</form> are the opening and closing form tags
- action="registration_form.php" method="POST"> specifies the destination URL and the submission type.
- First/Last name: are labels for the input boxes
- <input type=”text”…> are input box tags
- <br> is the new line tag
- <input type="hidden" name="form_submitted" value="1"/> is a hidden value that is used to check whether the form has been submitted or not
- <input type="submit" value="Submit"> is the button that when clicked submits the form to the server for processing

- **<u>Submitting the form data to the server</u>**

  The action attribute of the form specifies the submission URL that processes the data. The method attribute specifies the submission type.

- **<u>PHP POST method</u>**

- This is the built in PHP super global array variable that is used to get values submitted via HTTP POST method.
- The array variable can be accessed from any script in the program; it has a global scope.
- This method is ideal when you do not want to display the form post values in the URL.
- A good example of using post method is when submitting login details to the server.

- **It has the following syntax.**

  <?php $_POST['variable_name']; ?>
- "$_POST[…]" is the PHP array
- "'variable_name'" is the URL variable name.

- **PHP GET method**

- This is the built in PHP super global array variable that is used to get values submitted via HTTP GET method.
- The array variable can be accessed from any script in the program; it has a global scope.
- This method displays the form values in the URL.
- It’s ideal for search engine forms as it allows the users to book mark the results.

- **It has the following syntax.**
- <?php $_GET['variable_name']; ?>
- “$_GET[…]” is the PHP array
- “'variable_name'” is the URL variable name.

GET vs POST Methods

POST	GET
Values not visible in the URL	Values visible in the URL
Has not limitation of the length of the values since they are submitted via the body of HTTP	Has limitation on the length of the values usually 255 characters. This is because the values are displayed in the URL. Note the upper limit of the characters is dependent on the browser.
Has lower performance compared to Php_GET method due to time spent encapsulation the Php_POST values in the HTTP body	Has high performance compared to POST method dues to the simple nature of appending the values in the URL.
Supports many different data types such as string, numeric, binary etc.	Supports only string data types because the values are displayed in the URL
Results cannot be book marked	Results can be book marked due to the visibility of the values in the URL

- PHP Get Form
- Get request is the default form request. The data passed through get request is visible on the URL browser so it is not secured. You can send limited amount of data through get request.
- *Example : form1.html*

```
<form action="welcome.php" method="get">
Name: <input type="text" name="name"/>
<input type="submit" value="visit"/>
</form>
```

File: welcome.php

```
<?php
$name=$_GET["name"];//receiving name field value in $
    name variable
echo "Welcome, $name";
?>
```

- PHP Post Form
- Post request is widely used to submit form that have large amount of data such as file upload, image upload, login form, registration form etc.
- The data passed through post request is not visible on the URL browser so it is secured. You can send large amount of data through post request.

- *Example: form1.html*

```
<form action="login.php" method="post">
<table>
<tr><td>Name:</td><td> <input type="text" name="name"/></td></tr>
<tr><td>Password:</td><td> <input type="password" name="password"/></td><
    /tr>
<tr><td colspan="2"><input type="submit" value="login"/>  </td></tr>
</table>
</form>
```

login.php

```
<?php
$name=$_POST["name"];//receiving name field value in $name variable
$password=$_POST["password"];//receiving password field value in $password
    variable
echo "Welcome: $name, your password is: $password";
?>
```

- When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the HTTP POST method.

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

Output:

```
Welcome John
Your email address is john.doe@example.com
```

- The same result could also be achieved using the HTTP GET method:

```
<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>
```

- "welcome_get.php”

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

- Both GET and POST create an array (e.g. array(key1 => value1, key2 => value2, key3 => value3, ...)). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.
- Both GET and POST are treated as $_GET and $_POST. These are superglobals, which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.
- $_GET is an array of variables passed to the current script via the URL parameters.
- $_POST is an array of variables passed to the current script via the HTTP POST method.

When to use GET?

- Information sent from a form with the GET method is **visible to everyone** (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.
- GET may be used for sending non-sensitive data.

Note: GET should NEVER be used for sending passwords or other sensitive information!

When to use POST?

- Information sent from a form with the POST method is **invisible to others** (all names/values are embedded within the body of the HTTP request) and has **no limits** on the amount of information to send.
- Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.
- However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

Developers prefer POST for sending form data.

The $_REQUEST variable

- The PHP $_REQUEST variable contains the contents of both $_GET, $_POST, and $_COOKIE.
- The PHP $_REQUEST variable can be used to get the result from form data sent with both the GET and POST methods.
- The **$_REQUEST** variable is used to read the data from the submitted HTML form.
- **Sample code:**
- Here, the **$_REQUEST** variable is used to read the submitted form field with the name ‘**username**’. If the form is submitted without any value, then it will print as “**Name is empty**”, otherwise it will print the submitted value.

```
<!DOCTYPE html>
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_S
ELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // collect value of input field
    $name = htmlspecialchars($_REQUEST['fname']);
    if (empty($name)) {
        echo "Name is empty";
    } else {
        echo $name;
    }
}
?>

</body>
</html>
```

PHP- Required field

- Required field will check whether the field is filled or not in the proper way. Most of cases we will use the * symbol for required field.

What is Validation ?

- Validation means check the input submitted by the user. There are two types of validation are available in PHP.

- **Client-Side Validation** – Validation is performed on the client machine web browsers.

- **Server Side Validation** – After submitted by data, The data has sent to a server and perform validation checks in server machine.

Some of Validation rules for field

Field	Validation Rules
Name	Should required letters and white-spaces
Email	Should required **@** and **.**
Website	Should required a valid URL
Radio	Must be selectable at least once
Check Box	Must be checkable at least once
Drop Down menu	Must be selectable at least once

Form Validation in PHP

- An HTML form contains various input fields such as text box, checkbox, radio buttons, submit button, and checklist, etc. These input fields need to be validated, which ensures that the user has entered information in all the required fields and also validates that the information provided by the user is valid and correct.

- There is no guarantee that the information provided by the user is always correct. You need to validate a few things:

- Empty String
- Validate String
- Validate Numbers
- Validate Email
- Validate URL
- Input length

- Empty String
- The code below checks that the field is not empty. If the user leaves the required field empty, it will show an error message. Put these lines of code to validate the required field.

```
if (empty ($_POST["name"])) {
   $errMsg = "Error! You didn't enter the Name.";
          echo $errMsg;
} else {
  $name = $_POST["name"];
}
```

- Validate String
- The code below checks that the field will contain only alphabets and whitespace, for example - name. If the name field does not receive valid input from the user, then it will show an error message:

```
$name = $_POST ["Name"];
if (!preg_match ("/^[a-zA-z]*$/", $name) )
 {
   $ErrMsg = "Only alphabets and whitespace are allow
   ed.";
             echo $ErrMsg;
} else {
   echo $name;
}
```

- Validate Number
- The below code validates that the field will only contain a numeric value. **For example -** Mobile no. If the *Mobile no* field does not receive numeric data from the user, the code will display an error message:

```
$mobileno = $_POST ["Mobile_no"];
if (!preg_match ("/^[0-9]*$/", $mobileno) ){
    $ErrMsg = "Only numeric value is allowed.";
    echo $ErrMsg;
} else {
    echo $mobileno;
    }
```

- Validate Email
- A valid email must contain @ and . symbols. PHP provides various methods to validate the email address. Here, we will use regular expressions to validate the email address.

```
$email = $_POST ["Email"];
$pattern = "^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(
   \.[a-z0-9-]+)*(\.[a-z]{2,3})$^";
if (!preg_match ($pattern, $email) ){
   $ErrMsg = "Email is not valid.";
         echo $ErrMsg;
} else {
   echo "Your valid email address is: " .$email;
}
```

- Input Length Validation
- The input length validation restricts the user to provide the value between the specified range, for Example - Mobile Number. A valid mobile number must have 10 digits.

```
$mobileno = strlen ($_POST ["Mobile"]);
$length = strlen ($mobileno);

if ( $length < 10 && $length > 10) {
    $ErrMsg = "Mobile must have 10 digits.";
            echo $ErrMsg;
} else {
    echo "Your Mobile number is: " .$mobileno;
}
```

- Validate URL
- The below code validates the URL of website provided by the user via HTML form. If the field does not contain a valid URL, the code will display an error message, i.e., "URL is not valid".

```
 $websiteURL = $_POST["website"];

  if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-
  9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$web
  site)) {
  $websiteErr = "URL is not valid";
 echo $websiteErr;
} else {
  echo "Website URL is: " .$websiteURL;
}
```

- Button Click Validate
- The below code validates that the user click on submit button and send the form data to the server one of the following method - get or post.

```
if (isset ($_POST['submit'])) {
    echo "Submit button is clicked.";
    if ($_SERVER["REQUEST_METHOD"] == "POST") {

        echo "Data is sent using POST method ";
    }
} else {
    echo "Data is not submitted";
}
```

- **Note: Remember that validation and verification both are different from each other.**

PHP File Upload

- PHP allows you to upload single and multiple files through few lines of code only.
- PHP file upload features allows you to upload binary and text files both. Moreover, you can have the full control over the file to be uploaded through PHP authentication and file operation functions.

- <u>PHP $_FILES</u>
- The PHP global $_FILES contains all the information of file. By the help of $_FILES global, we can get file name, file type, file size, temp file name and errors associated with file.

- Here, we are assuming that file name is *filename*.

$_FILES['filename']['name']

- returns file name.

$_FILES['filename']['type']

- returns MIME type of the file.

$_FILES['filename']['size']

- returns size of the file (in bytes).

$_FILES['filename']['tmp_name']

- returns temporary file name of the file which was stored on the server.

$_FILES['filename']['error']

- returns error code associated with this file.

- move_uploaded_file() function
- The move_uploaded_file() function moves the uploaded file to a new location. The move_uploaded_file() function checks internally if the file is uploaded thorough the POST request. It moves the file if it is uploaded through the POST request.
- **Syntax**
- bool move_uploaded_file (string $filename , string $destination)

- *uploadform.html*

```
<form action="uploader.php" method="post" enctype="multipart/form-data">

    Select File:
    <input type="file" name="fileToUpload"/>
    <input type="submit" value="Upload Image" name="submit"/>
</form>
```

- *File: uploader.php*

```
<?php
$target_path = "e:/";
$target_path = $target_path.basename( $_FILES['fileToUpload']['name']);

if(move_uploaded_file($_FILES['fileToUpload']['tmp_name'], $target_path) {

    echo "File uploaded successfully!";
} else{
    echo "Sorry, file not uploaded, please try again!";
}

?>
```

What is PHP mail?

- PHP mail is the built in PHP function that is used to send emails from PHP scripts.
- The mail function accepts the following parameters;

Email address

Subject

Message

CC or BC email addresses

  - It's a cost effective way of notifying users on important events.
  - Let users contact you via email by providing a contact us form on the website that emails the provided content.
  - Developers can use it to receive system errors by email
  - You can use it to email your newsletter subscribers.
  - You can use it to send password reset links to users who forget their passwords
  - You can use it to email activation/confirmation links. This is useful when registering users and verifying their email addresses

mail() Function

- Example:

```
<?php
 $to_email = 'name @ company . com';
 $subject = 'Testing PHP Mail';
 $message = 'This mail is sent using the PHP
   mail function';
 $headers = 'From: noreply @ company . com';
  mail($to_email,$subject,$message,$headers);
?>
```