UNIT III- WORKING WITH FORMS, FILE UPLOADS AND EXCEPTION HANDLING

1. EXPLAIN ABOUT FORMS IN PHP?

★ Forms are a mechanism that allow you to type information into fields on a browser screen and submit the information into a web server. You can fill out a form submit it and then the information is uploaded to the designated server to handle a form.

★ In HTML, forms are used to collect user input/information. But, as far as HTML goes, it just provides the graphical interface and the ability to write in the input fields.

★ That usually means saving the inputs to a database so that they can be accessed later for several purposes. In this section, we will not send information to the database, as that requires an active database that is already set up and includes knowledge form SQL language, but we can retrieve information that the user has given us.

★ Forms have may functions. They can be used for gathering information about a user , survey, order online etc

- ★ FORM TAGS
  Forms provide an interface for collecting, displaying, delivering information and are used as a key component of HTML.
- ★ From is created by using <from>tag , it is also paired tag. <from></form>
- ★ Forms have different types of fields such as text input fields, radio buttons, check boxes, submit and reset button etc.
- ★ Syntax :
  <form action = "FILE NAME" method="post/get">
  Form fields……
  Form fields……
  </form>
  Eg :
  <form action = "ai.php" method="get">
  Name: <input type="text" name= "pname"/>
  Address : <input type="text" name="paddress"/>
  </form>

FORM INPUT TAGS

It is an independent tag which means it does not have closing tags. It supports 13 form controls with different structure.

- ➢ TEXT BOX: A rectangular shaped field in which a user can enter text is considered as text box. A text box is the type of input tag with name, size max length and values.
  Eg:
  NAME<input type="text" name="name" size=45>

  FNAME <input type="text" name="name" size=45>

- ➢ PASSWORD : The password input type creates a single line empty text box where the user can enter text into it. It is similar to text fields. It display the text in the fields like stars (*) and dots(.) that means it creates masked fields in passwords
  Eg:
  Username: <input type="text" name= "user">
  Password : <input type="password" name="password">
- ➢ CHECK BOXES: A check box is represented by square icon . This is used for multiple selections. The user can select or deselect by clicking on it. A selected check box is usually show in dark grey and unselected check box is shown in light grey
  Eg:
  <input type= "checkbox" name="vehicle" value="bike">BSC
  < input type= "checkbox" name="vehicle" value="car">BCOM

- RADIO BUTTONS: these are group of buttons from which only one can be selected at a time
  Eg:
  <input type=”radio” name=”gender” value=”male”>Male
  <input type=”radio” name=”gender” value=”female”>Female
- ACTION BUTTONS: Two types of action buttons they are submit and reset.
  The user clicks the submit button the values that have been entered into the form are sent the program that process the form.
  Reset button is to allow to clear all of the input entered in the from and starts from the first.
  Eg:
  Submit :<input type=”submit” value=”submit”>
  reset :<input type=”reset ” value=”clear”>
- SELECT : The select tag allow choosing any subset of items from a group , the items given in select tag are usually rendered in the style of a pop-up menu by using <option></option>tags but ending tag is optional.
  Eg: <select>
  <option value=”bcom”>bcom</option>
  <option value=”bsc”>bsc</option>
  </select>
- FILE : The file type of input tag provides a file upload from box for html forms.
  Eg: <form action=”demo_form.asp” method=”get”>
  <input type=”file” name=”textbox”>

- TEXT AREA : The<text area> tag defines a multi line text input control.
  A text area can hold an unlimited number of characters, fixed width font and the area can be specified by the cols and rows attributes.
  Eg:
  <text area rows=”10”cols=”30”>
  // text
  </text area>

EXAMPLE PROGRAM1:

```
<html>
<head>
<title> form using html </title>
</head>
<body>
<form action= ‘file.php’ method= ‘get>
Name: <input type= “text” name= ‘Name’>
<input type= “submit” value= ‘submit’>
</form>
</body>
</html>
```

Create a php program.

```
<html>
<head>
<title> php program<title>
</head>
<body>
<?php
$name=$_post[‘name’];
```

```
Echo “welcome .$_post[name’];
?>
</body> </html>
```

Example program 2:

```
<html>
<head>
<title>student form</title>
</head>
<body>
<form  action="mon.php" method="get">
Name:<input type="text"  name="name" placeholder="Name"> <br>
Email:<input type="text"  name="email" placeholder="E-Mail"> <br>
Password: <input type= “password” name= ‘pwd’> <br>
Age:<input type="number" name="age" placeholder="Age"> <br>
Gender:<input type="radio" name="gender" value="Male">Male
<input type="radio" name="gender" value="Female">Female <br>
Places: <select>
<option value= ‘kkd’>kkd<br>
<option value= ‘vzg’>vizag<br>
<option value= ‘hyd’>hyderabad<br>
<option value= ‘rjy’>rajamundry<br>
<option value= ‘vzm’>vizianagaram
</select><br>
Subjects: <br>
<input type="checkbox" id="sub1" name="sub1" value="artificial intelligence">
  <label for="sub1">Artificial intelligence</label><br>
<input type="checkbox" id="sub2" name="sub2" value="computer">
  <label for="sub2">computer science</label><br>
<input type="checkbox" id="sub3" name="sub3" value="chemistry">
  <label for="sub3">chemistry </label><br>
<input type="checkbox" id="sub4" name="sub4" value="php">
  <label for="sub4">php</label><br>
<input type="submit" name="submit" value="Submit">
</form>
</body>
</html>
```

This form is just a regular one, and includes inputs for name, e-mail, age and gender. This information will be subject of a print when the Submit button is clicked. That just proves that we got the information from the user. For now, let’s see what we got and fill the form.

Example:

```
<html>
<head>
<title>form</title>
</head>
<body>
<div class="cs">
<h2class="text">student data</h2>
<?php
```

```
$name=$_POST['name'];
$email=$_POST['email'];
$age=$_POST['age'];
$gender=$_POST['gender'];
if(empty($name)) {
Echo "enter student name";
}
else if(empty($email)) {
echo "enter email id";
}
elseif(empty($age)){
 echo "enter age";
}
elseif(empty($gender)){
echo "select gender";
}
Else{
 echo"Name:".$name."<br>email:".$email."<br>age:".$age."<br>gender:".$gender;
}
?>
</div>
</body>
</html>
```

Next, we check each input to make sure the user has written/chosen something, and the input is not empty. We do this using two well-known functions in PHP, the empty(). After making sure we have this right, we can also validate fields. In this case, only the name input may need some sort of validation to make sure one simply doesn't write numbers in there, while other inputs are more or less validated from HTML5, in the case of email and age.

2. WRITE ABOUT ACCESSING FORM INPUT VALUES IN PHP?

Now we know that how to gather information from HTML elements that submit a single value per elementname,suchastextfields,textareas,andradiobuttons.Butthereisaproblemwhenworking with elements such as checkboxes because it is possible for the user to choose one or more items.

EG:

```
<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>
```

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. To display the submitted data you could simply echo all the variables.

The "welcome.php" looks like this:

<html>
<body>
Welcome **<?php** echo $_POST["name"]; **?>**
Your email address is: **<?php** echo $_POST["email"]; **?>**
</body>
</html>
O/P:
Welcome John
Your email address is john.doe@example.com

3. **DIFFERNCE S BETWEEN GET AND POST METHODS?**
   Php provides 3 methods. They are get,post and request. These are denoted by $ and _ symbols with subscripts.
   To access the values from user by using $_get[ ] and $_post[ ] methods based on program.
   $_request[ ] method access the values from $_get[ ], $_post[ ] and cookies.

GET	**POST**
It is a method.	**I**t is a method.
It is used to transfers data to server.	**I**t is used to transfers data to server.
It supports only text format of data.	**It** supports all different kinds of data.
It displays the information in the address page.	It does not displays the information in the address page.
$_GET[ ]	**$_POST[ ]**
In the above method works like an array and used for accessing the values.	In the above method works like an array and used for accessing the values.
Less secure.	More secure.
It displays information in address page up to 1024 characters.	It represents file name only.
Eg: http://www.test.com/index.html?name=krishna&email=k@gmail.com	Eg:http://www.test.com/index.html?form.php

4. **HOW TO COMBINE HTML AND PHP CODE ON A SINGLE PAGE?**

★ Write PHP code with HTML code on a single page as on hard copy.
★ More flexibility to write the entire page dynamically.
★ For this use a PHP_SELF variable is in the action field of the <form> tag.
★ The action field of the FORM instructs where to submit the form data when the user presses the "submit" button.
★ The same PHP page as the handler for the form as well.
★ The action field of form use to switch to control to other page but Using PHP_SELF variable donot need to edit the action field.

Example:-
A file called self.php and want to load the same page after the form is submitted.
<form method="post" action="self.php">
We can use the PHP_SELF variable instead of "self.php".
The code becomes:
<form method="post" action="<?*Php* echo $_SERVER['PHP_SELF']; ?>

```
<?php
if(isset($_POST['submit']))
{
    $name=$_POST['name'];
    Echo "Glad to meet you<b>$name</b>";
    }
?>
```

```
<form method="post" action="<?Php echo $_SERVER['PHP_SELF'];?>">
<input type="text" name="name"><br>
<input type="submit" name="submit" value="Submit">

</form>
```

5. WRITE ABOUT HIDDEN FIELD AND REDIRECTING THE USER IN PHP?

- ★ The <input type="hidden"> defines a hidden input field. A hidden field lets web developers include data that cannot be seen or modified by users when a form is submitted.
- ★ A hidden field often stores what database record that needs to be updated when the form is submitted.
- ★ While the value is not displayed to the user in the page's content, it is visible (and can be edited) using any browser's developer tools or "View Source" functionality. Do not use hidden inputs as a form of security!

Eg:

```
<!DOCTYPE html>
<html>
<body>
<h1>A Hidden Field (look in source code)</h1>
<form action="/action_page.php">
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <input type="hidden" id="custId" name="custId" value="3487">
  <input type="submit" value="Submit">
</form>
<p><strong>Note:</strong> The hidden field is not shown to the user, but the data is sent when the form is submitted.</p>
</body>
</html>
```

REDIRECTION:

Redirection from one page to another in PHP is commonly achieved using the following two ways:

Using Header Function in PHP:

The header() function is an inbuilt function in PHP which is used to send the raw HTTP (Hyper Text Transfer Protocol) header to the client.

Syntax:

header($header, $replace, $http_response_code)

Parameters: This function accepts three parameters as mentioned above and described below:

- **$header:** This parameter is used to hold the header string.
- **$replace:** This parameter is used to hold the replace parameter which indicates the header should replace a previous similar header, or add a second header of the same type. It is optional parameter.
- **$http_response_code:** This parameter hold the HTTP response code.

```
<?php

// Redirect browser
header("Location: https://www.google.co.in/");

exit;
?>
```

AND REFER TO PAGE NO 85 PROGRAM

6. HOW TO SENDING MAIL ON FORM SUBMISSION?

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 BCC 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 news letter 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 Use the mail PHP

Sending mail using PHP

The PHP mail function has the following basic syntax

```
<?php
mail($to_email_address,$subject,$message,[$headers],[$parameters]);
?>
```

HERE,

- ✓ "$to_email_address" is the email address of the mail recipient
- ✓ "$subject" is the email subject
- ✓ "$message" is the message to be sent.
- ✓ "[$headers]" is optional, it can be used to include information such as CC, BCC
  - ★ CC is the acronym for carbon copy. It's used when you want to send a copy to an interested person i.e. a complaint email sent to a company can also be sent as CC to the complaints board.
  - ★ BCC is the acronym for blind carbon copy. It is similar to CC. The email addresses included in the BCC section will not be shown to the other recipients

```
<?php
$receiver="aditya16@gmail.com";
$subject="Email Test via PHP using Local host";
$body="Hi, there...This is a test email send from Local host.";
$sender="From:awdckkd@gmail.com";
if(mail($receiver, $subject, $body, $sender)){
echo "Email sent successfully to $receiver";
}else{
Echo "Sorry, failed while sending mail!";
}
```

7. EXPLAIN ABOUT FILES AND FILE UPLODING IN PHP?

Create The HTML Form

Next, create an HTML form that allows users to choose the image file they want to upload:

```
<!DOCTYPE html>
<html>
<body>
<form action="upload.php" method="post" enctype="multipart/form-data">
  Select image to upload:
  <input type="file" name="fileToUpload" id="fileToUpload">
  <input type="submit" value="Upload Image" name="submit">
</form>
</body>
</html>
```

Some rules to follow for the HTML form above:

- Make sure that the form uses method="post"
- The form also needs the following attribute: enctype="multipart/form-data".
- It specifies which content-type to use when submitting the form Without the requirements above, the file upload will not work.

Other things to notice:

- The type="file" attribute of the <input> tag shows the input field as a file-select control, with a "Browse" button next to the input control

The form above sends data to a file called "upload.php", which we will create next.

Create The Upload File PHP Script

The "upload.php" file contains the code for uploading a file:

```
<?php
$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));
// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
  $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
  if($check !== false) {
    echo "File is an image - " . $check["mime"] . ".";
```

```
  $uploadOk = 1;
 } else {
  echo "File is not an image.";
  $uploadOk = 0;
 }
}
?>
```
OR REFER TO PHP TEXT BOOK PG NO 87 TO 91

PHP script explained:

- $target_dir = "uploads/" - specifies the directory where the file is going to be placed
- $target_file specifies the path of the file to be uploaded
- $uploadOk=1 is not used yet (will be used later)
- $imageFileType holds the file extension of the file (in lower case)
- Next, check if the image file is an actual image or a fake image

Check if File Already Exists

Now we can add some restrictions.
First, we will check if the file already exists in the "uploads" folder. If it does, an error message is displayed, and $uploadOk is set to 0:

```
// Check if file already exists
if (file_exists($target_file)) {
 echo "Sorry, file already exists.";
 $uploadOk = 0;
}
```

Limit File Size

The file input field in our HTML form above is named "fileToUpload".
Now, we want to check the size of the file. If the file is larger than 500KB, an error message is displayed, and $uploadOk is set to 0:

```
// Check file size
if ($_FILES["fileToUpload"]["size"] > 500000) {
 echo "Sorry, your file is too large.";
 $uploadOk = 0;
}
```

8. WRITE ABOUT EXCEPTION HANDLING?

An exception is an unexpected program result that can be handled by the program itself. Exception Handling in PHP is almost similar to exception handling in all programming languages. Run time errors are php exceptions.

PHP provides the following specialized keywords for this purpose.

- ★ **try:** It represents a block of code in which exceptions can arise.
- ★ **catch:** It represents a block of code that will be executed when a particular exception has been thrown. It is used for solve the raised exception.
- ★ **throw:** It is used to throw an exception. It is also used to list the exceptions that a function throws, but doesn't handle itself.
- ★ **finally:** It is used in place of a catch block or after a catch block basically it is put for cleanup activity in PHP code.

Php exceptions are classified into 3 types. They are

i) Notices - Normal errors /optional
ii) warnings - file doesn't exist /serious
iii) Fatal errors-terminate the script / critical divert

set-error-handler( ) function is used to all php errors to a custom function that are defined like.

i)error type

ii) Descriptive message

iii) file name

iv) line number

- ★ Notices are php errors, these are default errors, not visible to any user.
- ★ Warning:

```
<?php

echo <script> alert('check your email id')

</script>;

?>
```

- ★ **fatal error:**

```
<?Pup

class Hello(){

echo 'Hello world';
}
callbyvalue();  -calling function.
?>
```

Fatal error: Call to undefined function call by value() in Ap.php on line 5.

EXAMPLE:

```
<?php
//create function with an exception
function checkNum($number) {
  if($number>1) {
    throw new Exception("Value must be 1 or below");
  }
  return true;
}
```

```
//trigger exception in a "try" block
try {
  checkNum(2);
  //If the exception is thrown, this text will not be shown
  echo 'If you see this, the number is 1 or below';
}

//catch exception
catch(Exception $e) {
  echo 'Message: ' .$e->getMessage();
}
?>
```

The code above throws an exception and catches it:

1. The checkNum() function is created. It checks if a number is greater than 1. If it is, an exception is thrown
2. The checkNum() function is called in a "try" block
3. The exception within the checkNum() function is thrown
4. The "catch" block retrieves the exception and creates an object ($e) containing the exception information
5. The error message from the exception is echoed by calling $e->getMessage() from the exception object