



Bharathidasan University

Centre for Differently Abled Persons
Tiruchirappalli - 620024.

- Programme Name: Bachelor of Computer Applications
- Course Code : 20UCA6CC9
- Course Title : PHP
- Unit : Unit III
- Compiled by : Dr. M. Prabavathy
Associate Professor

Ms. M. Hemalatha
Guest Faculty

FORM HANDLING

- Form are used to get input from the user and submit it to the web server for processing
- `$_GET` and `$_POST` are used to collect form-data
- When the user fills out the form and click the **SUBMIT** button, the form data is sent for processing

Simple Form Code:

```
<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>
```

OUTPUT:

N	<input type="text" value="viji"/>
E	<input type="text" value="cdap@gmail.com"/>
<input type="submit" value="Submit"/>	

- After the form data is submitted, data is sent for processing to a PHP file named “Welcome.php”
- The form data is sent with the HTTP POST method

welcome.php

```
<html>
```

```
<body>
```

```
Welcome <?php echo $_POST["name"]; ?> <br>
```

```
Your email address is: <?php echo $_POST["email"]; ?>
```

```
</body>
```

```
</html>
```

Output:

Welcome viji

Your email address is: cdap@gmail.com

FORM VALIDATION

Validation means check the input submitted by user

There are two types of validation

1. Client-Side Validation

Validation is performed on client machine i.e. web browser

2. Server-Side Validation

After submitting the data, data is sent to the server and perform validation checks in server machine

Some Validation rules:

Field	Validation Rules
Name	Required letters and white space
Email	Required @ and .
Website	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 Element

```
<form method="post" action="<?php echo  
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

- When form is submitted, the form data is sent with method = “post”
- The `$_SERVER["PHP_SELF"]` → returns the filename of currently executing script
- The `htmlspecialchars()` → function converts special characters to HTML entites

```
<form method="post" action="test_form.php"> → action is  
done to this page
```


Sample code

```
<?php
// define variables and set to empty values
$nameErr = "";
$name = "";
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (empty($_POST["name"])) {
        $nameErr = "Name is required"; → Validation
    } else {
        $name = test_input($_POST["name"]);
    }
}
?>
```

\$_GET Variable

- \$_GET is a global variable that collects form data using method= "get"
- \$_GET can also collect data sent in the URL.
- The GET method produce a long string that appear in browser

EXAMPLE:

<http://www.test.com/index.htm?name1=value1&name2=va>

```
FORM SUBMISSION GET METHOD
<form action="registration_form.php" method="GET">
  First name: <input type="text" name="firstname"><br>
  Last name: <input type="text" name="lastname">
  <br>
  <input type="hidden" name="form_submitted" value="1"/>
  <input type="submit" value="Submit">
</form>
```

SUBMISSION URL SHOWS FORM VALUES

localhost/tuttis/registration_form.php?firstname=Smith&lastname=Jones&form_submitted=1



Sample Code:

```
<html>
<body>
  <ahref="test_get.php?dress=CHUDI&website=amazon.com">Test $GET</a>
</body>
</html>
```

- When a user clicks on the link "Test \$GET", the parameters "subject" and "web" are sent to "test_get.php", and can access their values in "test_get.php" with \$_GET.

test_get.php

```
<html>
<body>
<?php
echo "Purchase " . $_GET['dress'] . " at " . $_GET['website'];
?>
</body>
</html>
```

OUTPUT

Purchase CHUDI at amazon.com

\$_POST Variable

\$_POST is a PHP super global variable which is used to collect form data after submitting an HTML form with method="post".

\$_POST is also widely used to pass variables.

FORM SUBMISSION POST METHOD

```
<form action="registration_form.php" method="POST">  
  First name: <input type="text" name="firstname"><br>  
  Last name: <input type="text" name="lastname">  
  <br>  
  <input type="hidden" name="form_submitted" value="1"/>  
  <input type="submit" value="Submit">  
</form>
```

Submission URL does not show form values



Sample code

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<form method="post" action="<?php echo  
$_SERVER['PHP_SELF'];?>">
```

```
  Name: <input type="text" name="fname">
```

```
  <input type="submit">
```

```
</form>
```

```
<?php
```

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
```

```
    // collect value of input field
```

```
    $name = $_POST['fname'];
```

```
    if (empty($name)) {
```

```
        echo "Name is empty";
```

```
    } else {
```

```
        echo $name;
```

```
    }
```

```
}
```

```
?> </body>
```

```
</html>
```

- In this code, PHP file points itself for processing data
- Name should have valid data
- If data is not entered it return a message “Name is empty”
- POST method is used to post either data or invalid message

\$_REQUEST Variable

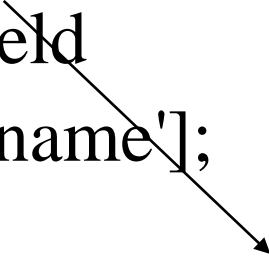
It collect data from HTML page after clicking SUBMIT button

Sample code:

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

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

collect



CREATING FORMS



CREATING A FORM

```
<html>
```

```
<head>
```

```
</head>
```

```
<body>
```

```
// "mail.php" is a php file name and its method is POST
```

```
<form action="mail.php" method="POST">
```

```
<p>Name</p> <input type="text" name="name"> → Name field
```

```
<p>Email</p> <input type="text" name="email">
```

```
<p>Phone</p> <input type="text" name="phone">
```

// CHECK BOX

<p>Request Phone Call:</p>

Yes:<input type="checkbox" value="Yes" name="call">

No:<input type="checkbox" value="No" name="call">

<p>Website</p> <input type="text" name="website">

// DROP DOWN

<p>Priority</p>

<select name="priority" size="1">

<option value="Low">Low</option>

<option value="Normal">Normal</option>

<option value="High">High</option>

```
<option value="Emergency">Emergency</option>
```

```
</select>
```

```
<br />
```

```
<p>Type</p>
```

```
<select name="type" size="1">
```

```
<option value="update">Website Update</option>
```

```
<option value="change">Information Change</option>
```

```
<option value="addition">Information Addition</option>
```

```
<option value="new">New Products</option>
```

```
</select>
```

```
<br />
```

// TEXTAREA

```
<p>Message</p><textarea name="message" rows="6"  
cols="25"></textarea><br />
```

//BUTTON

```
<input type="submit" value="Send"><input type="reset"  
value="Clear">  
</form>
```

mail.php

```
<?php
```

```
$name = $_POST['name'];
```

```
$email = $_POST['email'];
```

```
$phone = $_POST['phone'];
```

```
$call = $_POST['call'];
```

```
$website = $_POST['website'];
```

```
$priority = $_POST['priority'];
```

```
$type = $_POST['type'];
```

```
$message = $_POST['message'];
```



```
$formcontent=" From: $name \n Phone: $phone \n Call Back: $call
\n Website: $website \n Priority: $priority \n Type: $type \n Message:
$message";
$recipient = "youremail@here.com";
$subject = "Contact Form";
$mailheader = "From: $email \r\n";
mail($recipient, $subject, $formcontent, $mailheader) or
die("Error!");
echo "Thank You!";
?>
```

OUTPUT FORM

Name
Add your name

Email
Enter a Valid Email

Phone
Add a Phone Number

Website
Your Website

Priority Low
Priority Level

Type Website Update
Type of Contact

Message
Type Your Message

CREATING THE UPLOAD SCRIPT

- Upload PHP file to the server
- **phpinfo.php** page describes the temporary directory that is used for
 - file uploads as **upload_tmp_dir**
 - **upload_max_filesize** is maximum permitted size of file
- First, check for configuration file
- In **php.ini** file → `file_uploads = on`

\$_FILES → Global variable

- It is a double dimensional array and keep all information related to upload file

\$_FILES['file']['tmp_name'] – the uploaded file in the temporary directory on the web server.

\$_FILES['file']['name'] – the actual name of the uploaded file.

\$_FILES['file']['size'] – the size in bytes of the uploaded file.

\$_FILES['file']['type'] – the MIME type of the uploaded file.

\$_FILES['file']['error'] – the error code associated with this file upload.

FILE SYSTEM

USING YOUR FILE SYSTEM

The filesystem functions allow you to access and manipulate the file system.

FUNCTION	DESCRIPTION
<code>basename()</code>	Returns the filename component of a path
<code>chgrp()</code>	Changes the file group
<code>chmod()</code>	Changes the file mode
<code>copy()</code>	Copies the file
<code>fclose()</code>	Close an open file
<code>file()</code>	Read a file into an array
<code>fopen()</code>	Open a file or URL
<code>fread()</code>	Reads from an open file

FILE PATH AND PERMISSION

- The `realpath()` function returns the absolute pathname.
- This function removes all symbolic links (like `'./'`, `'../'` and extra `'/'`) and returns the absolute pathname.
- If the path is built starting from the system root, it is called **absolute**.
- If the path is built starting from the current location, it is called **relative**

Syntax

`Realpath(path)`

Sample Code

```
<?php  
echo realpath("test.txt");  
?>
```

Output

```
C:\Inetpub\testweb\text.txt
```


The **chmod()** function changes permissions of the specified

file

PARAMETER	DESCRIPTION
file	Specifies path of file
mode	Specifies new permission The mode has 4 parameters First number - always 0 Second number - permission of owner Third number- permission of owner's user group Fourth number- permission for everybody

- POSSIBLE VALUES
- 1 = execute permission
- 2 = write permission
- 4 = read permission

Syntax → `chmod(file, mode)`

DISPLAYING DIRECTORY CONTENT

- The directory functions allow you to retrieve information about directories and their contents.
- PHP directory functions

FUNCTION	DESCRIPTION
chdir()	Changes the current directory
chroot()	Changes the root directory
closedir()	Closes the directory handle
dir()	Return an instance of the directory class

getcwd()

Returns the current working directory

opendir()

Open a directory handle

readdir()

Returns entry from a directory handle

rewinddir()

Resets a directory handle

WORKING WITH `fopen()` and `fclose()`

`fopen()`

`fopen()` contains the name of the file to be open

second parameter specifies at which mode the file should open

Mode	Description
r	Open a file for read only
w	Open a file for write only (erase the content and create a new file)
a	Open a file for write only (existing file is preserved)
x	Create a new file for write only (return false when file already exists)
r+	Open a file for read/write (pointer start at beginning of the file)

w+

Open a file for read/write
(erase the content and
create a new file)

a+

Open a file for read/write
(existing file is preserved)

x+

Create a new file for
read/write

fclose()

- **fclose()** function is used to close an open file
- It required the name of the file or a variable that hold file name

Sample code

```
<?php
$file = fopen("test.txt", "r");
//Output lines until EOF is reached
while(! feof($file)) {
    $line = fgets($file);
    echo $line. "<br>";
} fclose($file); ?>
```



THANK YOU