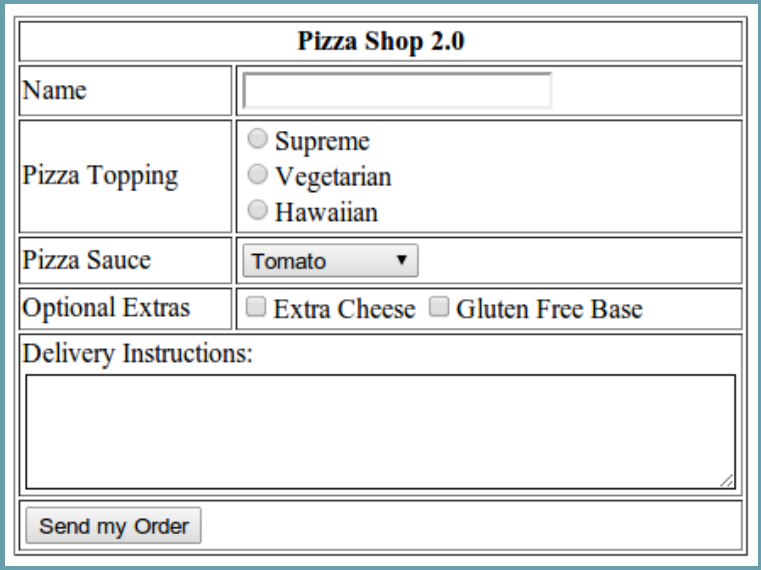
LECTURE 6 – ICT1233
FORM HANDLING



Department of ICT

Faculty of Technology

Objectives

- After the successful completion of this lecture, students should be able to,
  - *Manage form data using php*

HTML Forms

```
<html>
    <head>
        <title>A BASIC HTML FORM</title>
    </head>
    <body>
        <FORM NAME ="form1" METHOD =" "  ACTION = "">
            <INPUT TYPE = "TEXT“ NAME = “Name” VALUE="username">
            <INPUT TYPE = "Submit" NAME = "Submit1" VALUE = "Login“>
        </FORM>
    </body>
</html>
```

HTML Forms – Method Attribute

- Used to tell the browser how the form information should be sent
- The two most popular methods
  - *GET*
  - *POST*

Ex:-

- *<FORM NAME ="form1" METHOD ="GET" ACTION = “”>*
- *<FORM NAME ="form1" METHOD =“POST" ACTION = “”>*

GET Method

- Information sent from a form with the GET method is visible to everyone
- Use the GET method when the data you want to send is not crucial information that needs protection
- Has limits on the amount of information to send through GET method
- The predefined **$_GET** variable is used to collect values from a form with the method “get”

POST Method

- When Information sent from a form with the POST method,
  - *The information is invisible to others*
  - *No limits on the amount of information being sent*

- The predefined $_POST variable is used to collect values from a form sent with the method "post"

GET vs. POST

- Both GET and POST create an array (e.g. array(key => value, 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 theuser**
- $_GET and $_POST are super **globals**, which means that they are always accessible, regardless of scope
- $_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

Processing Form Data

- How does a PHP script get information from a client?
- How does a PHP script get information from the server when it is running on?
- How does PHP save information from a session with a client?

- **Answer: Using PHP superglobal arrays**

Superglobal Arrays

- Superglobals are built-in variables that are always available in all scopes
- There is no need to use "**global $variable**"; to access them within functions
  - *$_SERVER - stores data about the currently running server*
  - *$_ENV - stores data about the current client's environment*
  - *$_GET - stores data sent to the server using HTTP method GET*
  - *$_POST - stores data sent to the server using HTTP method POST*

$_REQUEST

- $_REQUEST is a super global variable which is widely used to collect data after submitting html forms
- Instead of using GET and POST arrays, you can also use the $_REQUEST array

```
<?php
      $name=$_REQUEST['name'];
      echo $name;
?>
```

$_SERVER

- Holds information about headers, paths, and script locations

```
<!DOCTYPE html>
<html>
<body>
	<?php
		echo $_SERVER['PHP_SELF'];
		echo "<br>";
		echo $_SERVER['SERVER_NAME'];
		echo "<br>";
		echo $_SERVER['SCRIPT_NAME'];
	?>
</body>
</html>
```

```
/myFile.php
localhost
/myFile.php
```

Action Property of Forms

- Action property indicates
  - Where to send the form data

EX:-

- <form method = "post" action = "form.php">
- The action attribute of the form element indicates that when the user clicks submit button, the form data will be posted to form.php

Using PHP_SELF in the Action Field of a Form

- **PHP_SELF** is a variable that returns the current script being executed
- Returns the name and the path of the current file (from the root folder)
  - echo $_SERVER['PHP_SELF'];
- Using PHP_SELF variable you can write more generic code which can be used on any page

```
<form name="form1" method="post" action="<?php echo
$_SERVER['PHP_SELF']; ?>" >
```

HANDLING FORM CONTROLS

Getting Values From a Text Box

- To get the text that a user entered into a text box, the text box needs the **NAME** attribute

Ex:-

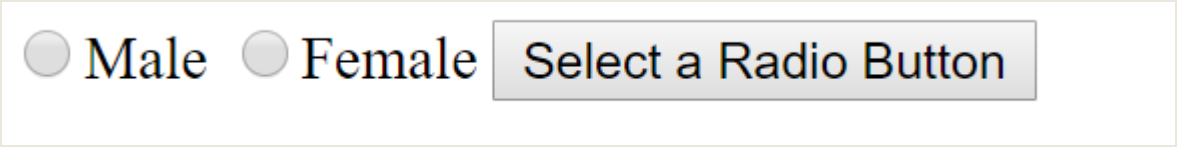
- <INPUT TYPE = "Text" VALUE ="username" NAME = "uname">
- To return data from an HTML form element, you can use the following syntax:

  $username = $_POST['uname']; or

  $username = $_GET['uname'];

PHP and Radio Buttons

```
<FORM name ="form1" method ="post" action ="radioButton.php">
    <Input type = 'Radio' Name ='gender' value= 'male'>Male
    <Input type = 'Radio' Name ='gender' value= 'female'>Female
    <Input type = "Submit" Name = "Submit1" VALUE = "Select a Radio Button">
</FORM>
```

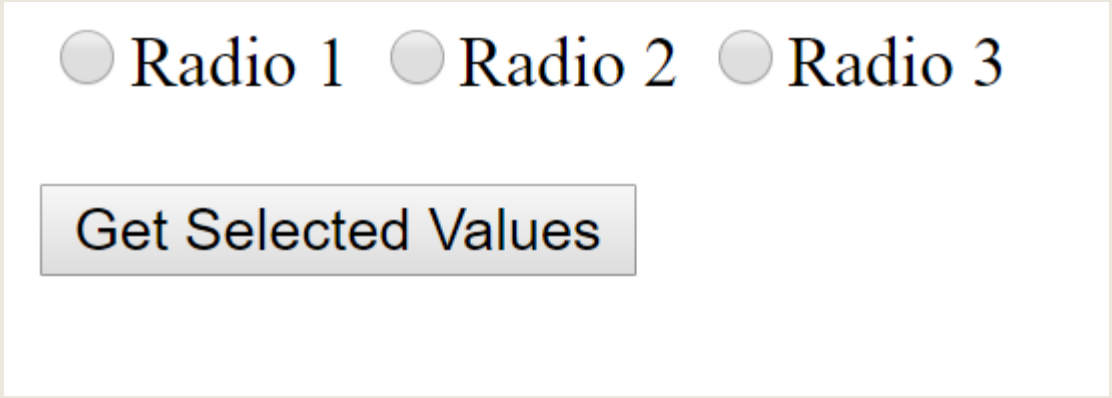


PHP and Radio Buttons

```
<?PHP
        $male_status = 'unchecked';
        $female_status = 'unchecked';
        if (isset($_POST['Submit1'])) {
                $selected_radio = $_POST['gender'];
                if ($selected_radio == 'male') {
                        $male_status = 'checked';
                        echo "Male Selected";
                }
                else if ($selected_radio == 'female') {
                        $female_status = 'checked';
                        echo "Female Selected";
                }
        }
?>
```

Exercise

- Write down the php code to display the following output and get the selected value from the radio buttons.
- When the user selects a radio button and press the button, the page should display “You have selected ***Radio x***” (X is th number of the button.



Answer

```
<html>
	<body>
		<form action="" method="post">
			<input type="radio" name="radio" value="Radio 1">Radio 1
			<input type="radio" name="radio" value="Radio 2">Radio 2
			<input type="radio" name="radio" value="Radio 3">Radio 3 <br><br>
			<input type="submit" name="submit" value="Get Selected Values" />
		</form>
	</body>
</html>
<?php
	if (isset($_POST['submit'])) {
		if(isset($_POST['radio'])){
			echo "You have selected :".$_POST['radio'];
		}
	}
?>
```

PHP and Dropdown List

```
<html>
	<body>
		<form action="" method="post">
			<select name="clr">
				<option value="Red">Red</option>
				<option value="Green">Green</option>
			</select>
			<input type="submit" name="btnsubmit" />
		</form>
	</body>
</html>
<?php
	if(isset($_POST['btnsubmit']))
	echo "Selected color is : ".$_POST['clr'];
?>
```

FORM VALIDATION

Validating the Input

- User input should be validated on the browser by client scripts
  - JavaScript
- Server validation is used if the user input will be inserted into a database
- **empty()**
  - Determine whether a variable is empty **if(empty($_POST["name"]))**
- **Isset()**
  - Determine if a variable is set and is not NULL **if(isset($_POST['submit']))**

Form Processing

- Confirm that valid information was entered
- **extract()** function
  - Creates variables corresponding to each key-value pair in an array
  - Easily retrieve all values sent to the PHP page
  - **Note:** Do not use extract() on untrusted data, like user input with $_GET
- Regular expressions are very helpful
- Ending a script
  - **die()** function : Terminates script execution
  - Remember to close all HTML tags

```
<?php
	extract( $_POST );
	if( isset($_POST['submit']) ) {
	.....................
	}
	// determine whether phone number is valid and print an error message if not
	if(!preg_match( "/^(\(\+[0-9]{2}\)[0-9]{9})$/","412278456") ){
	// Perform a regular expression match
	print( "<p><span style = \"color: red; font-size: 2em\">
	INVALID PHONE NUMBER</span><br />
	A valid phone number must be in the form
	<strong>(+94)412278456</strong><br />
	</span></p></body></html>" );
	die(); // terminate script execution
	}
	else{ echo "Valid phone no"; }
?>
```

Validating the input - Text

- Check if the name field only contains letters and whitespace
- If the value of the name field is not valid, then store an error message:

```
$name = $_POST["name"];
if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
        $nameErr = "Only letters and white space allowed";
}
```

- The **preg_match()** function searches a string for pattern, return true if the pattern exists, and false otherwise

Validating the Input - Email

- Validate email format
- Use PHP's **filter_var()** function to check whether an email address is well-formed
- **filter_var** — Filters a variable with a specified filter
- FILTER_VALIDATE_EMAIL filter validates an e-mail address

```
$email = $_POST["email"];
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    $emailErr = "Invalid email format";
}
```

Var_dump()

- Display structured information (type and value) about one or more variables
- Arrays and objects are explored recursively with values indented to show structure
- All public, private and protected properties of objects will be returned in the output
- **Syntax**

```
var_dump(variable 1, variable 2, ....variable n)
```

Example – var_dump()

```php
<?php
$a = array(1, 2, array("a", "b", "c"));
var_dump($a);
?>
```

```
array(3) {
  [0]=>
  int(1)
  [1]=>
  int(2)
  [2]=>
  array(3) {
    [0]=>
    string(1) "a"
    [1]=>
    string(1) "b"
    [2]=>
    string(1) "c"
  }
}
```

Summary

- HTML Forms
  - *METHOD*
  - *ACTION*
  - *SUBMIT*
- Superglobal arrays
- Capture user input
- Form validation using php

Questions.....