



Chapter 4

Advanced Concepts in PHP

4.1. Other PHP-Database Functions

- `mysql_affected_rows ()` — Get number of affected rows in previous MySQL operation.
It helps you to know how many rows have been deleted, inserted, modified, etc. by the last `mysql_query()` operation.
`int mysql_affected_rows ( [resource link_identifier])`
- `mysql_affected_rows()` returns the number of rows affected by the last INSERT, UPDATE or DELETE query associated with `link_identifier`.
- `mysql_create_db ()` — Create a MySQL database. The syntax is:
`bool mysql_create_db (string databasename [, resource link_identifier]);`
- `mysql_create_db()` attempts to create a new database on the server associated with the specified link identifier.

4.1. Other PHP-Database Functions...

- ❑ `mysql_drop_db` — Drop (delete) a MySQL database
- ❑ `mysql_drop_db()` attempts to drop (remove) an entire database from the server associated with the specified link identifier.
`bool mysql_drop_db (string database_name [, resource link_identifier])`
- ❑ `mysql_num_rows` — Get number of rows in result.
- ❑ The syntax is:
`int mysql_num_rows (resource result);`
- ❑ `mysql_num_rows()` returns the number of rows in a result set.
- ❑ This command is only valid for SELECT statements.
- ❑ To get rows affected by INSERT, UPDATE or DELETE query, use `mysql_affected_rows()`.
- ❑ `mysql_num_fields` — Get number of fields in result. The syntax is:
`int mysql_num_fields (resource result);`

4.1. Other PHP-Database Functions...

- `mysql_field_name` — Get the name of the specified field in a result.
`string mysql_field_name (resource result, int field_index)`
- `mysql_field_name()` returns the name of the specified field index.
- `field_index` is the numerical offset of the field.
- Note that `field_index` starts at 0.

- `mysql_list_tables` — List tables in a MySQL database.
`resource mysql_list_tables (string database [, resource link_identifier])`
- `mysql_list_tables()` takes a database name and returns a result pointer much like the `mysql_query()` function.
- You can use the `mysql_tablename()` function to extract the actual table names from the result pointer, or any other result table function such as `mysql_fetch_assoc()`.

4.1. Other PHP-Database Functions...

```
<?php
$link = mysql_connect('localhost', 'root', '');
$db_list = mysql_query('SHOW DATABASES');
while ($row = mysql_fetch_row($db_list)) {
    echo $row[0] . "<br>";
}
?>
```

4.2. Working with Files

3.2.1 Uploading Files

- You may want users to upload files to your Web site.
- For example, you may want users to be able to upload resumes to your job-search Web site or pictures to your photo album Web site.
- You can display a form that allows a user to upload a file by using an HTML form designed for that purpose.
- The general format of the form is as follows:

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

```
<input type="hidden" name="MAX_FILE_SIZE" value="30000">
```

```
<input type="file" name="user_file">
```

```
<input type="submit" value="Upload File">
```

```
</form>
```

4.2. Working with Files...

- Notice the following points regarding the form:
 - **The enctype attribute is used in the form tag.** You must set this attribute to multipart/form-data when uploading a file.
 - **A hidden field is included that sends a value (in bytes) for MAX_FILE_SIZE.** If the user tries to upload a file that is larger than this value, it won't upload.
 - MAX_FILE_SIZE in your form need to consider two size settings in php.ini
 - **upload_max_filesize:** The MAX_FILE_SIZE you send in your upload form can't be larger than the value of upload_max_filesize. If you are uploading a larger file than the current value of upload_max_filesize (2MB by default), you need to increase that value by editing php.ini.
 - **post_max_size:** The total amount of information you send in a POST form can't be larger than the value of post_max_size (8MB by default). You can increase this value if necessary by editing your php.ini file.
 - **The input field that uploads the file is of type file.**

4.2. Working with Files...

- The value for `MAX_FILE_SIZE` must be sent before the file is uploaded if you want the file size limit to apply to the uploading file.
- When the user submits the form, the file is uploaded to a temporary location.
- The script that processes the form needs to copy the file to another location because the temporary file is deleted as soon as the script is finished.
- You can use `phpinfo()` to see where the temporary files are stored.
- If you don't like the location of the temporary directory, you can change it by changing **`upload_tmp_dir`** in the `php.ini` file.
- If no directory is specified in `php.ini`, a default temporary directory is used.

4.2. Working with Files...

Accessing Information About An Uploaded File

- Along with the file, information about the file is sent with the form.
- This information is stored in the PHP built-in array called `$_FILES`.
- An array of information is available for each file that was uploaded.
- You can obtain the information from the array by using the name of the field.

`$_FILES['fieldname']['name']` – contains filename

`$_FILES['fieldname']['type']` – contains type of file

`$_FILES['fieldname']['tmp_name']` – contains temporary location of file

`$_FILES['fieldname']['size']` – contains size of file

4.2. Working with Files...

- If the user uploads a file named test.txt by using the form, the resulting array that can be used by the processing script looks something like this:

```
echo $_FILES['user_file']['name'];           // test.txt
echo $_FILES['user_file']['type'];           // text/plain
echo $_FILES['user_file']['tmp_name'];       //D:/WINNT/php92C.tmp
echo $_FILES['user_file']['size'];           // 435
```

- In this array, name is the name of the file that was uploaded, type is the type of file, tmp_name is the path/filename of the temporary file, and size is the size of the file.
- Notice that name contains only the filename, while tmp_name includes the path to the file as well as the filename.

4.2. Working with Files...

- If the file is too large to upload, the `tmp_name` in the array is set to `none`, and the `size` is set to 0.
- By default, PHP stores the temporary uploaded file in your temp directory on Windows or `/tmp` on Unix/Linux.
- You can change the location of temporary files by setting `php.ini`.
- Look in your `php.ini` file for the following line:
`upload_tmp_dir =`

4.2. Working with Files...

Moving Uploaded Files To Their Destination

- Since the temporary file created during upload is deleted when the script finishes, it has to be moved to another location if you want to store it permanently.
- The general format of the statement that moves the file is as follows:

```
move_uploaded_file(path/tempfilename, path/permfilename);
```

- You can use the following statement to move the file to your desired location, in this case, c:\data\new_file.txt:

```
move_uploaded_file($_FILES['user_file']['tmp_name'],  
                  'c:/data/new_file.txt');
```

4.2. Working with Files...

- The destination directory (in this case, c:\data) must exist before the file can be moved to it.
- This statement doesn't create the destination directory.
- Security can be an issue when uploading files.
- Allowing strangers to load files onto your computer is risky; malicious files are possible.
- So, check the files for as many factors as possible after they are uploaded, using conditional statements to check file characteristics, such as checking for the expected file type and for the size.
- In some cases, for even more security, it may be a good idea to change the name of the file to something else so users don't know where their files are or what they're called.

```
<html>
<head><title>File Upload</title></head>
<body>

<form enctype="multipart/form-data" action="up.php"
method="POST">
<input type="text" name="title" placeholder="Enter Title"><br>
<input type="hidden" name="MAX_FILE_SIZE" value="500000">
<input type="file" name="file" size="20">
<p><input type="submit" name="Upload" value="Upload
Picture">
</form>
</body>
</html>
```

```
<?php
```

```
$temp = explode(".", $_FILES["file"]["name"]);
```

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

```
if ($_FILES["file"]["error"] > 10000) {
```

```
    echo "Return Code: " . $_FILES["file"]["error"] . "<br>";
```

```
} else {
```

```
    if (file_exists($_FILES["file"]["name"])) {
```

```
        echo "<b>".$_FILES["file"]["name"] . " already exists</b>.";
```

```
    } else {
```

```
        move_uploaded_file($_FILES["file"]["tmp_name"],
```

```
        "" . $_FILES["file"]["name"]);
```

```
        $name=$_FILES["file"]["name"];
```

```
        mysql_connect("localhost","root","");
```

```
        mysql_select_db("mydb");
```

```
        $query=mysql_query("insert into book (title) values('$title')");
```

```
        if($query) {
```

```
            echo "Successfully Added";
```

```
        } else { echo "error".mysql_error(); }
```

```
    }
```

```
}
```

```
?>
```

4.3. File Input/Output

- Many applications require the long-term storage of information.
- Information can be stored on the server in flat files or in databases.
- Flat files are text files stored in the computer file system.
- You can access and edit these files by using any text editors such as notepad or gedit.
- The information in the flat file is stored as strings, and the PHP script that retrieves the data needs to know how the data is stored.
- For example, to retrieve a customer name from a file, the PHP script needs to know that the customer name is stored in the first 20 characters of every line.
- Using a database for data storage requires you to install and learn to use database software, such as MySQL or Oracle.
- The data is stored in the database and can only be accessed by the database software.
- Databases can store very complex information that you can retrieve easily.

4.3. File Input/Output...

- You don't need to know how the data is stored, just how to interact with the database software.
- The database software handles the storage and delivers the data, without the script needing to know exactly where or how the customer name is stored.
- Flat files have some advantages over databases:
 - **Available and versatile:** You can create and save data in any operating system's file system. You don't need to install any extra software.
 - Additionally, text data stored in flat files can be read by a variety of software programs, such as word processors or spreadsheets.
 - **Easy to use:** You don't need to do any extra preparation, such as install database software, design a database, create a database, and so on.
 - Just create the file and store the data with statements in your PHP script.
 - **Smaller:** Flat files store data by using less disk space than databases.

4.3. File Input/Output...

- Databases have advantages as well:
 - **Security:** A database provides a security layer of its own, in addition to the security provided by the operating system.
 - A database protects the data from outside intrusion better than a flat file.
 - **Accessibility of data:** You can store data in a database by using a very complex data structure, specifying data types and relationships among the data.
 - The organization of the data makes it easy to search the data and retrieve what you need.
 - **Ability to handle multiple users:** When many users store or access data in a single file, such as a file containing names and addresses, a database ensures that users take their turn with the file to avoid overwriting each other's data.

4.3. File Input/Output...

Using Flat Files

- Flat files are also called text files.
- Using a flat file requires three steps:
 - Open the file.
 - Write data into the file or retrieve data from the file.
 - Close the file.
- The first step, before you can write information into or read information from a file, is to open the file.
- The following is the general format for the statement that opens a file:
`$fh = fopen("filename","mode")`
- The variable, `$fh`, referred to as a file handle, is used in the statements that write data to or read data from the opened file.
- `$fh` contains the information that identifies the location of the open file.

4.3. File Input/Output...

Mode	what it means	what happens when file does not exist
r	read only	If the file does not exist, a warning message is displayed.
r+	reading and writing	If the file does not exist, a warning message is displayed.
w	write only	If the file does not exist, PHP attempts to create it. If the file exists, PHP overwrites it.
w+	reading and writing	If the file does not exist, PHP attempts to create it. If the file exists, PHP overwrites it.
a	append data at the end of file	If the file does not exist, PHP attempts to create it.
a+	reading and appending	If the file does not exist, PHP attempts to create it.

4.3. File Input/Output...

- You can open the file `grade.txt` to read the information in the file with the following statement:
`$fh = fopen("grade.txt","r");`
- Based on this statement, PHP looks for `grade.txt` in the current directory, which is the directory where your PHP script is located.
- If the file can't be found, a warning message is displayed.
- A warning condition does not stop the script.
- The script continues to run, but the file isn't open, so statements that read or write to the file aren't executed.
- You probably want the script to stop if the file can't be opened.
- You can do this with a `die` statement, as follows:
`$fh = fopen("file1.txt","r") or die("Can't open file");`

4.3. File Input/Output...

Opening files in write mode

- You can open a file in a specified directory to store information by using the following type of statement:
`$fh = fopen("c:/testdir/happy.txt","w");`
- If the file does not exist, it is created in the indicated directory.
- However, if the directory doesn't exist, the directory is not created, and a warning is displayed.
- You can also open a file on another Web site by using a statement such as the following:
`$fh = fopen("http://www.data.com/index.html","r");`
- You can use a URL only with a read mode, not with a write mode.

4.3. File Input/Output...

- To close a file after you have finished reading or writing it, use:
`fclose($fh);`
- In this statement, `$fh` is the file handle variable you created when you opened the file.

Reading from file

- You can read from a file by line by line using the `fgets` statement, which has the following general format:
`$line = fgets($fh);`
- Sometimes you want to read strings of a certain size from a file.
- You can tell `fgets` to read a certain number of characters:
`$line = fgets($fh, n);`
- This statement tells PHP to read a string that is `n-1` characters long until it reaches the end of the line or the end of the file.

4.3. File Input/Output...

- You can check the end of file by using feof function:

```
feof($fh);
```

- Example: a program that reads file line by line until end of file and displays it

```
<?php
```

```
$fh = fopen("test.txt", "r") or die("Can't open file");
```

```
while(!feof($fh))
```

```
{
```

```
    $line = fgets($fh);
```

```
    print ("<br>$line");
```

```
}
```

```
fclose($fh);
```

```
?>
```

4.3. File Input/Output...

- Another option is to read a single character at a time from a file.
- You can do this using the `fgetc()` function.
- We can replace the while loop in our original script with one that uses `fgetc()`:

```
while (!feof($fp)) {  
    $char = fgetc($fp);  
    echo ($char);  
}
```
- This code reads a single character from the file at a time and stores it in `$char`, until the end of the file is reached.
- We can also read the whole file in one go.
- This can be done using `readfile()`.

```
readfile("test.txt");
```

4.3. File Input/Output...

- A call to the `readfile()` function opens the file, echoes the content to standard output (the browser), and then closes the file.
- The prototype for `readfile()` is
`int readfile(string filename, int [use_include_path]);`
- The optional second parameter specifies whether PHP should look for the file in the `include_path`.
- The final way we can read from a file is using the `fread()` function to read an arbitrary number of bytes from the file.
- This function has the following prototype:
`string fread(int fp, int length);`
- It reads up to `length` bytes or to the end of file, whichever comes first.

4.3. File Input/Output...

Writing to a File

- Writing to a file in PHP is relatively simple.
- You can use either of the functions `fwrite()` or `fputs()`
- `fputs()` is an alias to `fwrite()`.
- We call `fwrite()` in the following:

```
fwrite($fp, $outputstring);
```

- This tells PHP to write the string stored in `$outputstring` to the file pointed to by `$fp`.
- The function `fwrite()` actually takes three parameters but the third one is optional.
- The prototype for `fwrite()` is:

```
int fwrite(int fp, string str, int [length]);
```

4.3. File Input/Output...

- The third parameter, length, is the maximum number of bytes to write.
- If this parameter is supplied, fwrite() will write string to the file pointed to by fp until it reaches the end of string or has written length bytes, whichever comes first.
- Example: write data to file

```
<?php
```

```
$fh = fopen("test.txt","a");  
$data = "This content is written to file \n";  
$data = $data . "This line is also written";  
fwrite($fh, $data);  
fclose($fh);
```

```
?>
```

4.3. File Input/Output...

Seeking in File

- The function `fseek()` can be used to set the file pointer to some point within the file.
- It moves the file pointer, depending on the parameters.
`fseek($fh, int offset, mode);`
- The value specified in `offset` is the character location to move the pointer to. You can set `mode` to:
 - `SEEK_SET` (moves to char in position `offset`),
 - `SEEK_CUR` (moves `offset` characters forward from current position), or
 - `SEEK_END` (moves `offset` characters back from the last character).
- The `rewind()` function is equivalent to calling the `fseek()` function with an `offset` of zero.

4.3. File Input/Output...

Getting information about files

- Often you want to know information about a file.
- PHP has functions that allow you to find out file information about the files from within a script.
- You can find out whether a file exists with the `file_exists` statement, as follows:
`$result = file_exists(filename);`
- After this statement, `$result` contains either `TRUE` or `FALSE`.
- The function is often used in a conditional statement, such as the following:

```
<?php
    if(!file_exists("tesht.txt")) {
        echo "File not found!\n";
    }
?>
```

4.3. File Input/Output...

Function	What It Does	Output
<code>is_file("stuff.txt")</code>	Tests whether the file is a regular file, rather than a directory or other special type of file	TRUE or FALSE
<code>is_dir("stuff.txt")</code>	Tests whether the file is a directory	TRUE or FALSE
<code>is_executable("do.txt")</code>	Tests whether the file is executable	TRUE or FALSE
<code>is_writable("stuff.txt")</code>	Tests whether you can write to the file	TRUE or FALSE
<code>is_readable("stuff.txt")</code>	Tests whether you can read the file	TRUE or FALSE
<code>filesize("stuff.txt")</code>	Returns the file size in bytes	Integer or FALSE
<code>basename("/t1/do.txt")</code>	Returns the filename from the path	do.txt
<code>dirname("/t1/do.txt")</code>	Returns the directory name from the path	/t1
<code>copy(oldfile, newfile)</code>	Copy an existing file into a new file	TRUE or FALSE
<code>rename(oldname, newname)</code>	renames a file oldname to newname	TRUE or FALSE
<code>unlink("badfile.txt")</code>	deletes an unwanted file	TRUE or FALSE

4.3. File Input/Output...

□ Example:

```
if(!file_exists("library.txt"))  
    copy("book.txt", "library.txt");  
else  
    echo "File already exists!\n";  
readfile("library.txt");
```

4.4. Mathematical Functions

- PHP has built-in mathematical constants, and trigonometric, logarithmic, and base conversion functions.

- PHP constants are:

Constant	Description	Constant	Description
M_PI	Pi	M_SQRT2	sqrt(2)
M_PI_2	pi/2	M_SQRT1_2	1/sqrt(2)
M_PI_4	pi/4	M_LOG2E	log ₂ (e)
M_1_PI	1/pi	M_LOG10E	log ₁₀ (e)
M_2_PI	2/pi	M_LN2	loge(2)
M_2_SQRTPI	2/sqrt(pi)	M_LN10	loge(10)
M_E	the constant e		

function	Behavior
pow()	Takes two numerical arguments and returns the first argument raised to the power of the second. The value of pow(\$x, \$y) is x^y .
exp()	Takes a single argument and raises e to that power. The value of exp(\$x) is e^x .
log()	The “natural log” function. Takes a single argument and returns its base e logarithm. If $e^y = x$, then the value of log(\$x) is y.
log10()	Takes a single argument and returns its base-10 logarithm. If $10^y = x$, then the value of log10(\$x) is y.
pi()	Takes no arguments and returns an approximation of pi (3.1415926535898). Can be used interchangeably with the constant M_PI.
sin()	Takes a numerical argument in radians and returns the sine of the argument as a double.
cos()	Takes a numerical argument in radians and returns the cosine of the argument as a double.
tan()	Takes a numerical argument in radians and returns the tangent of the argument as a double.
asin()	Takes a numerical argument and returns the arcsine of the

acos()	Takes a numerical argument and returns the arccosine of the argument in radians. Inputs must be between -1.0 and 1.0, otherwise, it will return a result of NAN. Results are in the range 0 to pi.
atan()	Takes a numerical argument and returns the arctangent of the argument in radians. Results are in the range $-\pi / 2$ to $\pi / 2$.
atan2()	A variant of atan() that takes two arguments. Atan(\$y, \$x) is identical to atan(\$y/\$x) when \$x is positive, but the quadrant of atan2's result depends on the signs of both \$y and \$x. Range of the result is from $-\pi$ to π .
sqrt (\$num)	Returns square root of a number.
floor(\$float)	Returns the next lowest integer value by rounding down value.
ceil(\$float)	Returns the next highest integer value by rounding up value.
round(\$val [,precision])	Returns the rounded value of val to specified precision (number of digits after the decimal point). precision can also be negative or zero (default).
abs(\$num)	Returns the absolute value of number.

4.4. Mathematical Functions....

□ Example:

```
echo pow(2, 8);      //256
echo log10(100);    //2
echo sqrt(9);        // 3
echo ceil(4.3);      // 5
echo round(3.4);     // 3
echo round(3.6);     // 4
echo round(1.95583, 2); // 1.96
echo round(1241757, -3); // 1242000
echo pi();           // 3.1415926535898
```

4.4. Mathematical Functions....

Function	Behavior
<code>bindec()</code>	Takes a single string argument representing a binary (base 2) integer, and returns a string representation of that number in base 10.
<code>decbin()</code>	Like <code>bindec()</code> , but converts from base 10 to base 2.
<code>octdec()</code>	Like <code>bindec()</code> , but converts from base 8 to base 10.
<code>decoct()</code>	Like <code>bindec()</code> , but converts from base 10 to base 8.
<code>hexdec()</code>	Like <code>bindec()</code> , but converts from base 16 to base 10.
<code>dechex()</code>	Like <code>bindec()</code> , but converts from base 10 to base 16.
<code>base_convert()</code>	Takes a string argument (the integer to be converted) and two integer arguments (the original base, and the desired base). Returns a string representing the converted number—digits higher than 9 (from 10 to 35) are represented by the letters a-z. Both the original and desired bases must be in the range 2

4.4. Mathematical Functions....

□ Example:

```
function display_bases($val, $first_base) {  
    for ($new_base = 2; $new_base <= 9; $new_base++)  
    {  
        $converted = base_convert($val, $first_base, $new_base);  
        print("$val in base $first_base is $converted in base $new_base<BR>");  
    }  
}  
display_bases("150", 10);
```

□ Output:

```
150 in base 10 is 10010110 in base 2  
150 in base 10 is 12120 in base 3  
150 in base 10 is 2112 in base 4  
150 in base 10 is 1100 in base 5  
150 in base 10 is 410 in base 6  
150 in base 10 is 303 in base 7  
150 in base 10 is 226 in base 8  
150 in base 10 is 176 in base 9
```

Exercise

- Write a PHP function that accepts a, b, and c and solve quadratic equation
$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$
- Write a program that calculates income tax of a given salary

Employment income (per month)		Income tax payable (%)	Deductible fee
Over birr	To birr		
0	150	Exempt threshold	
151	650	10	15
651	1400	15	47.50
1401	2350	20	117.50
2351	3550	25	235.00
3551	5000	30	412.50
Over 5000		35	662.00

4.5. PHP - Form validation

□ **What is Validation?**

- Validation means check the input submitted by the user. There are two types of validation are available in PHP. They are as follows:
 - **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:
 - Name Should required letters and white-spaces,
 - Email Should required @ and dot,
 - 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

□ Example: Valid URL

```
<?php
$website = $_POST["site"];
if (!preg_match("\b(?:?:https?|ftp):\\V|www\\.)([-a-z0-9+&@#\\%?
=\\_!|:,;]*[-a-z0-9+&@#\\%=\\_!|/i", $website)) {
    echo "Invalid URL";
} else{
    echo "Your Site: ", $website;
} ?>
```

- Above syntax will verify whether a given URL is valid or not. It should allow some keywords as https, ftp, www, a-z, 0-9,..etc..

□ Example: Check Valid Email

```
<?php
$email = $_POST["email"];
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    echo "Invalid format and please re-enter valid email";
} else {
    echo "Your Email: ", $email;
}
?>
```

- Above syntax will verify whether given Email address is well-formed or not. If it is not, it will show an error message.

□ Example: shows the form with required field validation

```
<?php
if (empty($_POST["name"])) {
    echo "<b>Name: </b>Name is required";
} else { echo "<b>Name: </b>", $_POST["name"]; }
if (empty($_POST["email"])) { echo "<br><b>Email:</b>Email required";
} else { $email=$_POST["email"];
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        echo "<br><b>Email: </b>Invalid email format"; }
    else { echo "<br><b>Email: </b>", $email; } }
if(empty($_POST["gender"])) { echo "<br><b>Gender:</b>Gender
required"; } else {
    echo "<br><b>Gender: </b>", $_POST["gender"]; }
?>
```

□ **Example:**

- **PHP-Complete Form:** This example explains about time real-time form with actions. Below example will take input fields as text, radio button, drop down menu, and checked box.

```
<html><head><style>
.error {color: #FF0000;}
</style></head><body>
    <?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $class = $course = $subject = "";
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (empty($_POST["name"])) {
        $nameErr = "Name is required";
    } else {
        $name = test_input($_POST["name"]);
    }
}
```

```
if (empty($_POST["email"])) {  
    $emailErr = "Email is required";  
} else {  
    $email = test_input($_POST["email"]);  
    // check if e-mail address is well-formed  
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {  
        $emailErr = "Invalid email format";  
    }  
}  
if (empty($_POST["course"])) {  
    $course = "";  
} else {  
    $course = test_input($_POST["course"]);  
}
```

```
if (empty($_POST["class"])) {  
    $class = "";  
} else {  
    $class = test_input($_POST["class"]);  
}  
if (empty($_POST["gender"])) {  
    $genderErr = "Gender is required";  
} else {  
    $gender = test_input($_POST["gender"]);  
}  
if (empty($_POST["subject"])) {  
    $subjectErr = "You must select 1 or more";  
} else {  
    $subject = $_POST["subject"];  
}  
}
```

```
function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
```

?>

<h2>Absolute classes registration</h2>

<p>* required field.</p>

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

<table>

<tr>

<td>Name:</td>

<td><input type = "text" name = "name">

* <?php echo \$nameErr;?>

</td>

</tr>

```
<tr>
  <td>E-mail: </td>
  <td><input type = "text" name = "email">
<span class = "error">* <?php echo $emailErr;?></span>
  </td>
</tr>
<tr>
  <td>Time:</td>
  <td> <input type = "text" name = "course">
<span class = "error"><?php echo $websiteErr;?></span>
  </td>
</tr>
<tr>
  <td>Classes:</td>
  <td> <textarea name = "class" rows = "5" cols =
"40"></textarea></td>
</tr>
```

```
<tr><td>Gender:</td>
<td><input type="radio" name="gender" value =
"female">Female
<input type = "radio" name = "gender" value = "male">Male
<span class = "error">* <?php echo $genderErr;?> </span>
</td></tr><tr>
<td>Select:</td><td>
<select name = "subject[]" size = "4" multiple>
<option value = "Android">Android</option>
<option value = "Java">Java</option>
<option value = "C#">C#</option>
<option value = "Data Base">Data Base</option>
<option value = "Hadoop">Hadoop</option>
<option value = "VB script">VB script</option>
</select></td>
</tr>
```

```
<tr>
  <td>Agree</td>
  <td><input type = "checkbox" name = "checked" value =
"1"></td>
  <?php if(!isset($_POST['checked'])) { ?>
  <span class = "error">* <?php echo "You must agree to
terms";?></span>
  <?php } ?>
</tr>
<tr>
  <td>
<input type = "submit" name = "submit" value = "Submit">
  </td>
</tr>
</table>
</form>
```

```
<?php
echo "<h2>Your given values are as :</h2>";
echo ("<p>Your name is $name</p>");
echo ("<p> your email address is $email</p>");
echo ("<p>Your class time at $course</p>");
echo ("<p>your class info $class </p>");
echo ("<p>your gender is $gender</p>");
for($i = 0; $i < count($subject); $i++) {
    echo($subject[$i] . " ");
}
?>
</body>
</html>
```