RGPVNOTES.IN

Program : **B.Tech**

Subject Name: **Internet and Web Technology**

Subject Code: **CS-504**

Semester: **5th**

Unit-V Subject Notes

Subject code-CS 504 **Subject Name- Internet & Web Technology**

5.1 BASIC COMMANDS WITH PHP EXAMPLES:

There have a lot of PHP commands available for use in the various environment, especially for preparing one web application or embedding the entire server-side codebase with HTML syntax and very easy to learn for the normal developer. Some of the basic PHP commands are mentioned like below:

1. PHP Variables:

- **Types of Variable**: Variable always played important role in any kind of programming language. PHP also uses the declaration of the variable for assigning the value. One of the key features of PHP variable is, it is not required to declare the type of the variable. As PHP is a weekly type language, declare variable considering type based on the assigned value. PHP normally accepted varieties type of any variable like string, integer, float, Boolean, object, resource, array or NULL.

- **Name of the Variable**: Variable name in PHP always start with $, followed by any text or specific letter and _. PHP variable name is case sensitive, so any capital letter variable with the same name should be considered as a new variable.

- **Scope of the variable**: Maximum variables are in the local scope. Variable declare inside the function are not available out of the function, on the same approach variable declare outside of the function are not available inside the function. It is possible to declare a global variable in PHP, in that case, need to declare that variable as global specifically, or accessing the same through the global array.

2. PHP Operators:

- **Operator for assignments**: PHP normally uses one common operator for assignment which is equal to (‘=’). Left of this equal sign is the variable name and right will be the assigned value.

- **Operators for arithmetic operation**: Below operators are used for performing an arithmetic operation in PHP. Operators are ‘+’, ‘-’,’*’, ‘/’, ‘%’, ‘++’, ‘–’.

- **Operators for combination**: It is basically a combination of arithmetic operator and assignment operator. Combined operators are ‘+=’, ‘-=’, ‘*=’, ‘/=’, ‘%=’.

- **Operators for comparison**: Comparison operators are ‘==’, ‘!=’, ‘>’, ‘>=’, ‘<’, ‘<=’.

- **Operator for logical expression**: Logical operators in PHP are ‘||’, ‘&&’, ‘and’, ‘or’, ‘xor’, ‘!’.

3. PHP If Else:

- **Conditional Judgement**: For any kind of conditional requirement in the programming logic PHP used 'if else' feature like any other programming language. The basic syntax of 'IF ELSE' statement for PHP is:

IF [SPECIFIC CONDITION]{
[CODE] }ELSE IF [SPECIFIC CONDITION 2]{
[CODE] }ELSE {
[CODE] }

4. PHP Switch:

PHP is using switch case as well, like other programming languages for avoiding the nested definition of multiple 'IF ELSE'. Switch case considering multiple numbers of cases, and defining default is optional. Code structure of defining switch case is like below:

SWITCH($var){

CASE 'val 1'

[CODE] Break;

CASE 'val 2'

[CODE] Break;

CASE 'val 3'

[CODE] Break;

DEFAULT

[CODE] }

5. PHP Loop:

- **While Loop**: in PHP, while loop can be executed till the mention expression is considering as true.

WHILE [condition or expression]{
[CODE]
}

- **FOR Loop:** For loop is using for executing the same code for mention number of times.

For(exp 1, exp 2, exp 3){
[CODE]
}

- **Do While Loop**: Similar to the while loop, the code will be executed until the get true value in while expression. The main difference with while is, the code mention inside the do at least execute one whether the expression is true or not, but while not ensure the same.

DO {
[CODE]
}WHILE (condition)

- **FOREACH Loop**: This loop is accepting an array as variable and considering of executing code till the last element of the array.

FOREACH ($arr_var as $val){
[CODE]
}

5.2 CONNECTION TO SERVER

In PHP you can easily do this using the mysqli_connect() function. All communication between PHP and the MySQL database server takes place through this connection. Here're the basic syntaxes for connecting to MySQL using MySQLi and PDO extensions:

Syntax: MySQLi, Procedural way

$link = mysqli_connect("hostname", "username", "password", "database");

Syntax: MySQLi, Object Oriented way

$mysqli = new mysqli("hostname", "username", "password", "database");

Syntax: PHP Data Objects (PDO) way

$pdo = new PDO("mysql:host=hostname;dbname=database", "username", "password");

The hostname parameter in the above syntax specify the host name (e.g. localhost), or IP address of the MySQL server, whereas the username and password parameters specifies the credentials to access MySQL server, and the database parameter, if provided will specify the default MySQL database to be used when performing queries.

<?php

/* Attempt MySQL server connection. Assuming you are running MySQL server with default

setting (user 'root' with no password) */

$link = mysqli_connect("localhost", "root", ""); // Check connection

if($link === false){ die("ERROR: Could not connect. " . mysqli_connect_error()); } // Print host

information

echo "Connect Successfully. Host info: " . mysqli_get_host_info($link);

?>

5.3 CREATING DATABASE, SELECTING A DATABASE, LISTING DATABASE

5.3.1 CREATE DATABASE

In this statement, after that we will execute this SQL query through passing it to the PHP mysqli_query() function to finally create our database.

<?php /* Attempt MySQL server connection. Assuming you are running MySQL server with default setting (user 'root' with no password) */ $link = mysqli_connect("localhost", "root", ""); // Check connection if($link === false){ die("ERROR: Could not connect. " .

mysqli_connect_error()); } // Attempt create database query execution $sql = "CREATE DATABASE demo";

if(mysqli_query($link, $sql)){ echo "Database created successfully"; } else{ echo "ERROR: Could not able to execute $sql. " . mysqli_error($link); } // Close connection mysqli_close($link); ?>

5.3.2 SELECT

In this statement is used to select the records from database tables. Its basic syntax is as follows:

```
SELECT column1_name, column2_name, columnN_name FROM table_name;
```

Let's make a SQL query using the SELECT statement, after that we will execute this SQL query through passing it to the PHP mysqli_query() function to retrieve the table data.

Consider our *person's* database table has the following records:

+----+------------+-----------+---------------------+

| id | first_name | last_name | email |

+----+------------+-----------+---------------------+

| 1 | Peter | Parker | peterparker@mail.com |

| 2 | John | Rambo | johnrambo@mail.com |

| 3 | Clark | Kent | clarkkent@mail.com |

| 4 | John | Carter | johncarter@mail.com |

| 5 | Harry | Potter | harrypotter@mail.com |

+----+------------+-----------+---------------------+

5.3.3 LIST DATABASES

Using the above query we will develop a PHP PDO script to display all the databases present. Here is the code:

```
<?Php

require "config.php"; //Database Connection

$result = $dbo->query("SHOW DATABASES");

while ($row = $result->fetch(PDO::FETCH_NUM))

{
        echo $row[0]."<br>";
}

?>
```

mysql_list_tables (string $database [, resource $link_identifier = NULL]) : resource

5.4 LISTING TABLE NAMES

1. We **connect to MySQL using the PDO object**. In this particular example, I am using the database "test".
2. We create our SQL statement, which is "SHOW TABLES". This SQL statement tells MySQL to return a list of the tables that exist in our **currently-selected** database.
3. We prepare the SQL statement.
4. We execute the SQL statement.
5. We fetch the results using fetchAll.
6. Finally, we loop through the results and print out the name of each table.

```
<?php
 //Selecting a list of table names from a different database.
$sql = "SHOW TABLES FROM my_other_database";
?>
```

5.5 CREATING A TABLE

CREATE TABLE statement is used to create a table in MySQL.

We will create a table named "MyGuests", with five columns: "id", "firstname", "lastname", "email" and "reg_date":

```
CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)
```

Notes on the table above:

The data type specifies what type of data the column can hold. For a complete reference of all the available data types, go to our Data Types reference.

After the data type, you can specify other optional attributes for each column:

- NOT NULL - Each row must contain a value for that column, null values are not allowed
- DEFAULT value - Set a default value that is added when no other value is passed
- UNSIGNED - Used for number types, limits the stored data to positive numbers and zero
- AUTO INCREMENT - MySQL automatically increases the value of the field by 1 each time a new record is added
- PRIMARY KEY - Used to uniquely identify the rows in a table. The column with PRIMARY KEY setting is often an ID number, and is often used with AUTO_INCREMENT

5.6 INSERTING DATA

After a database and a table have been created, we can start adding data in them.

Here are some syntax rules to follow:

- The SQL query must be quoted in PHP
- String values inside the SQL query must be quoted
- Numeric values must not be quoted
- The word NULL must not be quoted

The INSERT INTO statement is used to add new records to a MySQL table:

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

5.7 ALTERING TABLES

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

To add a column in a table, use the following syntax:

ALTER TABLE table_name ADD column_name datatype;

Example

ALTER TABLE Customers ADD Email varchar(255);

5.8 QUERIES

The mysqli_query() function performs a query against the database.
mysqli_query(*connection,query,resultmode*);

```
<?php
$con=mysqli_connect("localhost","my_user","my_password","my_db");
// Check connection
if (mysqli_connect_errno())
 {
 echo "Failed to connect to MySQL: " . mysqli_connect_error();
 }

// Perform queries
mysqli_query($con,"SELECT * FROM Persons");
mysqli_query($con,"INSERT INTO Persons (FirstName,LastName,Age)
VALUES ('Glenn','Quagmire',33)");

mysqli_close($con);
?>
```

5.9 DELETING DATABASE

To delete data in a table in the SQLite database from a PHP application, you use these steps:

- Connect to the SQLite database by creating an instance of the PDO class.
- Prepare a DELETE statement for execution.
- Pass values to the statement using the bindValue() method of the PDOStatement object.

```
DELETE FROM table_name WHERE some_column = some_value
```

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}

// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";
```

```
if ($conn->query($sql) === TRUE) {
   echo "Record deleted successfully";
} else {
   echo "Error deleting record: " . $conn->error;
}

$conn->close();
?>
```

5.10 DELETING DATA AND TABLES

1. Delete Data

- Connect to the MySQL database by creating a new instance of the PDO object.
- Construct a DELETE statement to delete a row, multiple rows, or all rows in a table. ...
- Execute the DELETE statement by calling the exec() method of the PDO object or the execute() method of the PDOStatement object.

```
+----+------------+-----------+---------------------+

| id | first_name | last_name | email               |

+----+------------+-----------+---------------------+

| 1 | Peter      | Parker    | peterparker@mail.com |

| 2 | John       | Rambo     | johnrambo@mail.com   |

| 3 | Clark      | Kent      | clarkkent@mail.com   |

| 4 | John       | Carter    | johncarter@mail.com  |

| 5 | Harry      | Potter    | harrypotter@mail.com |

+----+------------+-----------+---------------------+
```

```
<?php

$link = mysqli_connect("localhost", "root", "", "demo");

if($link === false){ die("ERROR: Could not connect. " . mysqli_connect_error()); }

$sql = "DELETE FROM persons WHERE first_name='John'";

 if(mysqli_query($link, $sql))
```

{ echo "Records were deleted successfully."; }

else{ echo "ERROR: Could not able to execute $sql. " . mysqli_error($link); }

mysqli_close($link); ?>

5.11 PHP MYADMIN AND DATA BASEBUGS

5.11.1 MY ADMIN

PhpMyAdmin is a LAMP application that is written in PHP. Its specific purpose is to enable users with the ability to interact with and **administer MySQL servers**. All the information of WordPress is collected in the MySQL database, which is then coordinated with the database to create information in the WordPress site. **phpMyAdmin offers** a graphical interface of the data, tables and fields stored in the MySQL database for database administration tasks.

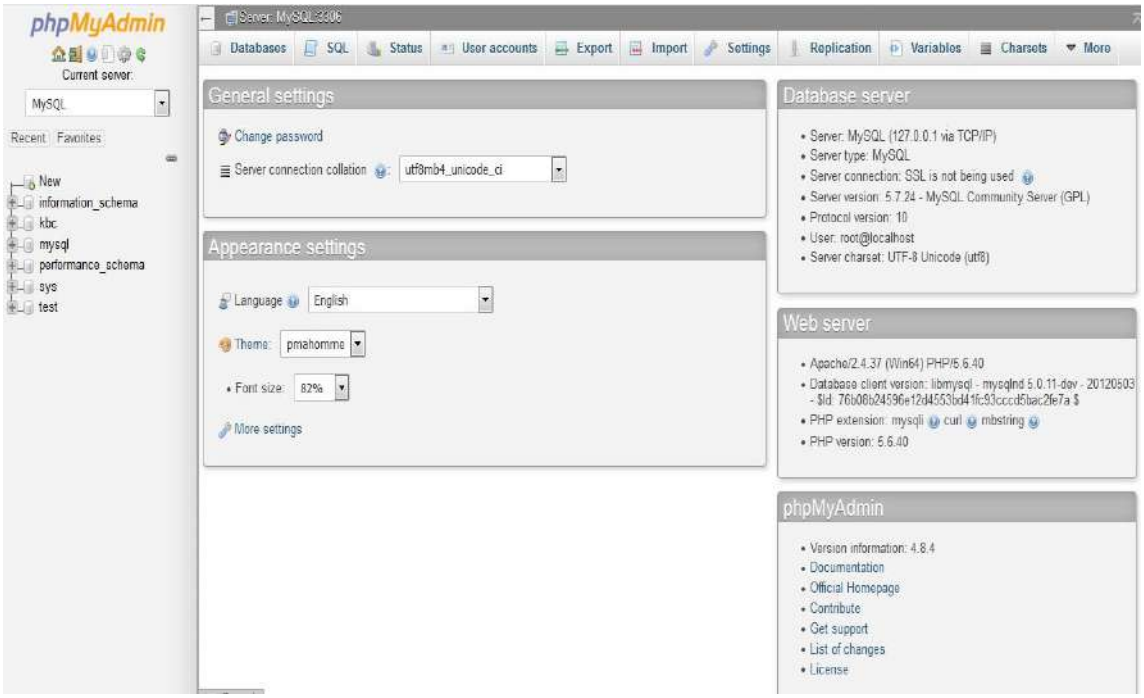

Figure 5.1 : PHPmyadmin Interface

The Prime Features of PHPmyadmin

- Insightful web interface

- Support for almost all MySQL features including **SQL-statement execution, editing and bookmarking**, Housed procedures and triggers management, databases, tables, fields and indexes management and MySQL users and privileges management.
- Data import from CSV and SQL
- Multiple servers administration
- Generating queries via Query-by-example (QBE)
- Searching worldwide in a database or a subset of it
- Generating PDF graphics of your database design

5.11.2 DATABASEBUGS

PHP defines some constants you can use to set the value of error_reporting such that only errors of certain types get reported: E_ALL (for all errors except strict notices), E_PARSE (parse errors), E_ERROR (fatal errors), E_WARNING (warnings), E_NOTICE (notices), and E_STRICT (strict notices).

While writing your PHP program, it is a good idea to use PHP-aware editors like BBEdit or Emacs. One of the special features of these editors is syntax highlighting. It changes the color of different parts of your program based on what those parts are. For example, strings are pink, keywords such as if and while are blue, comments are grey, and variables are black.

Another feature is quote and bracket matching, which helps to make sure that your quotes and brackets are balanced. When you type a closing delimiter such as}, the editor highlights the opening {that it matches.

There are following points which need to be verified while debugging your program.

- Missing Semicolons − Every PHP statement ends with a semicolon (;). PHP doesn't stop reading a statement until it reaches a semicolon. If you leave out the semicolon at the end of a line, PHP continues reading the statement on the following line.
- Not Enough Equal Signs − When you ask whether two values are equal in a comparison statement, you need two equal signs (==). Using one equal sign is a common mistake.
- Misspelled Variable Names − If you misspelled a variable then PHP understands it as a new variable. Remember: To PHP, $test is not the same variable as $Test.

- Missing Dollar Signs − A missing dollar sign in a variable name is really hard to see, but at least it usually results in an error message so that you know where to look for the problem.
- Troubling Quotes − You can have too many, too few, or the wrong kind of quotes. So check for a balanced number of quotes.
- Missing Parentheses and curly brackets − They should always be in pairs.
- Array Index − All the arrays should start from zero instead of 1.

Moreover, handle all the errors properly and direct all trace messages into system log file so that if any problem happens then it will be logged into system log file and you will be able to debug that problem

RGPVNOTES.IN

We hope you find these notes useful.

You can get previous year question papers at https://qp.rgpvnotes.in .

If you have any queries or you want to submit your study notes please write us at rgpvnotes.in@gmail.com

LIKE & FOLLOW US ON FACEBOOK
facebook.com/rgpvnotes.in