PHP State Management

Introduction

- State management is used to help web applications to maintain their state in several HTTP requests when needed. PHP provide two different techniques to manage the state of your web application.
  - Server Side State Management
    - Session
  - Client Side State Management
    - Query String
    - Cookies

PHP Cookies

Cookies: Introduction

- What is a Cookie?
  - A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer.
- Create Cookies With PHP
  - A cookie is created with the *setcookie()* function
- Syntax
  - *setcookie(name, value, expire, path, domain, secure, httponly);*

Cookies: Introduction

- **Name –** This sets the name of the cookie and is stored in an environment variable called HTTP_COOKIE_VARS. This variable is used while accessing cookies.
- **Value –** This sets the value of the named variable and is the content that you actually want to store.
- **Expiry –** This specify a future time in seconds since 00:00:00 GMT on 1st Jan 1970. After this time cookie will become inaccessible. If this parameter is not set then cookie will automatically expire when the Web Browser is closed.
- **Path –** This specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.
- **Domain –** This can be used to specify the domain name in very large domains and must contain at least two periods to be valid. All cookies are only valid for the host and domain which created them.
- **Secure –** This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which mean cookie can be sent by regular HTTP.

Cookies: Example

```
<?php
$cookie_name = "user";
$cookie_value = “ABC";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
?>
```

Within Body

```
<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>
```

PHP Session

Session: Introduction

- A PHP session is used to store data on a server rather than the computer of the user.
- Session identifiers or SID is a unique number which is used to identify every user in a session based environment.
- Session identifiers is used to link the user with his information on the server like posts, emails etc.

Session: Starting & Storing data

- The first step is to start up a session. After a session is started, session variables can be created to store information.
- The PHP session_start() function is used to begin a new session.It als creates a new session ID for the user.
- Storing Session Data: Session data in key-value pairs using the $_SESSION[] superglobal array.The stored data can be accessed during lifetime of a session.

```
<?php

session_start();

$_SESSION["Rollnumber"] = "11";
$_SESSION["Name"] = "Ajay";

?>
```

Accessing and Destroy Session

- Data stored in sessions can be easily accessed by firstly calling **session_start()** and then by passing the corresponding key to the **$_SESSION** associative array.
- The **session_destroy()** functi on is used to completely destroy a session. The session_destroy() function does not require any argument.

```
<?php

session_start();

echo 'Name:' .
$_SESSION["Name;
echo 'Roll number:' .
$_SESSION["Rollnumber"];

session_destroy();

?>
```

session_regenerate_id

- Update the current session id with a newly generated one
- Description
  - bool session_regenerate_id ([bool $delete_old_session = false])
  - session_regenerate_id() will replace the current session id with a new one, and keep the current session information.

Example

```
<?php
session_start();
$old_sessionid = session_id();
session_regenerate_id();
$new_sessionid = session_id();
echo "Old Session: $old_sessionid<br />";
echo "New Session: $new_sessionid<br />";
?>
```

PHP File Upload

$_FILES

- $_FILES is a two dimensional associative superglobal of items which are being uploaded by via HTTP POST method.
- Attributes of $_FILES
  - [name] - Name of file which is uploading
  - [size] - Size of the file
  - [type] - Type of the file (like .pdf, .zip, .jpeg…..etc)
  - [tmp_name] - A temporary address where the file is located before processing the upload request
  - [error] -Types of error occurred when the file is uploading
- How to Use
  - $_FILES[input-field-name][name]
  - $_FILES[input-field-name][tmp_name]
  - $_FILES[input-field-name][size]
  - $_FILES[input-field-name][type]
  - $_FILES[input-field-name][error]

Create The HTML Form

```
<form action="upload.php" met
hod="post" enctype="multipart
/form-data">

    Select image to upload:
    <input type="file" name="file
ToUpload" id="fileToUpload">

    <input type="submit" value="
Upload Image" name="submit">
</form>
```

Mandatory Attributes

- form uses *method= "post"*
- *enctype="multipart/form-data",* specifies how the form-data should be encoded when submitting it to the server.
- *type="file"* attribute of the <input> tag shows the input field as a file-select control.

Upload File PHP Script

```
<?php   $target_dir = "uploads/";
$target_file = $target_dir .
basename($_FILES["fileToUpload"]["name
"]);
$imageFileType= strtolower(pathinfo($targ
et_file,PATHINFO_EXTENSION));
if(isset($_POST["submit"])) {
$check = getimagesize
($_FILES["fileToUpload"]["tmp_name"]);
   if($check !== false) {
echo “Image File" . $check["mime"] . ".";
   }
else {
       echo "File is not an image.";
       $uploadOk = 0; }
}
?>
```

- $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
- $imageFileType: holds the file extension of the file.

Check File Existence, Size and Format

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

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

// Allow certain file formats
if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg"
&& $imageFileType != "gif" ) {
    echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
    $uploadOk = 0;
}
```

PHP Validation

Validation: Introduction

- 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.

Validation: Functions

- **empty():** Function used to check whether a variable is empty or not.
- **preg_match():** Function searches a string for pattern, returning true if the pattern exists, and false otherwise.
- **isset ():** Function is used to check whether a variable is set or not.

Example: To Check the empty fields

```
<?php
    $var = "";
            if( empty($var ) )
            {
                    echo "The variable is empty";
            }
            else
             {
                    echo "The variable is having some value";
            }

?>
```

Preg_match() Function

- This function matches the value given by the user and defined in the regular expression.

- If the regular expression and the value given by the user, becomes equal, the function will return true, false otherwise.

Syntax:

Preg_match($Pattern , $Subject , $regs)

- Pattern – Pattern is used to search the string.
- Subject – input given by the user.
- Regs

  If matches are found for parenthesized substrings of ***pattern*** and the function is called with the third argument ***regs***, the matches will be stored in the elements of the array ***regs***.

Example:

- preg_match(" /^Manipal/ " , " Manipal Academy of Higher Education")

Example

```
<?php
$pattern = "/^[A-Z ]{1,}\.$/i";
$subject = “Manipal Academy of Higher Education";
      if (preg_match( $pattern , $subject ) )
      {
                  echo "Pattern Matched";
      }
       else
      {
                  echo "Pattern Mismatched";
      }
?>
```

It will match only the required pattern

The dot(.) is compulsory in the end of the string

PHP with MySQL

Basic PHP MySQL functions

- Connecting to a Database
- Selecting a database
- Running a query
- Using results of a query
- Closing the connection

Database Connection

- **mysqli_connect(server, username, password)**
- server default is the string **"localhost"**
- username is a string for the user name (“ABCD")
- password is a string for the password (“abcd")

$conn= mysqli_connect("localhost", “ABCD", “abcd");

For Windows Labs

$conn= mysqli_connect("localhost", “ABCD", “abcd");

For WAMP/MAMP/XAMPP with default password

$conn= mysqli_connect("localhost", "root", "");

Sample Connection Code

```
<?php
$conn = mysqli_connect("localhost", “root", "")
    or die("Could not connect: " . mysqli_error($conn));
print "Successful Connection";
mysqli_close($conn);
?>
```

Connection: MySQLi Object-Oriented

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

// Create connection
$conn = new mysqli($servername, $username, $password);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

Connection: MySQLi Procedural

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

// Create connection
$conn = mysqli_connect($servername, $username, $password);

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>
```

Connection: PDO (PHP Data Objects)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
try {
   $conn = new PDO("mysql:host=$servername;dbname=myDB", $username, $password);
   // set the PDO error mode to exception
   $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
   echo "Connected successfully";
   }
catch(PDOException $e)
   {
   echo "Connection failed: " . $e->getMessage();
   }
?>
```

Selecting a database

- **mysqli_select_db(connection , name)**
- select a database given by the string name
- The connection variable is required

e.g.

mysqli_select_db($conn, "labuser");

Connect & Select database code

```
<?php
$conn = mysqli_connect('localhost', ‘root', '')
or die ('No connection : ' . mysqli_error($conn));
mysqli_select_db($conn , ‘user') or die ('db will not open' .
    mysqli_error($conn));
print "Database  Connected";
mysqli_close($conn);
?>
```

Error Messages

- **mysqli_error(connection)**
- Returns an error string or error number (connection is optional - with last opened connection used if none supplied)
- Empty string is returned if there is no error.

Example

mysqli_error($conn);

Making A Query

- **mysqli_query(connection , query)**
- makes a select query
- query is a string for the MySQL query (in SQL)
- semicolon (**;**)should NOT be used to terminate query
- query uses valid SQL command

e.g.

```
$query = "SELECT * FROM student";
$result = mysqli_query($conn, $query);
```

Connect, Select db & query

```
<?php
$conn = mysqli_connect('localhost', ‘root', '')
or die ('No connection');
mysqli_select_db($conn ,‘user') or die ('db will not open');
$query = "SELECT surname FROM student";
$result = mysqli_query($conn, $query) or die("Invalid query");
print "Successful Query";
mysqli_close($conn);
?>
```

Closing a connection

- **mysqli_close(connection)**
- closes the database connection with the link

e.g.

mysqli_close($conn);

- **mysqli_free_result(result)**
- frees up memory during a program

mysqli_free_result($result);

Other mysql Functions (1)

- **mysqli_num_rows(result)**
- returns number of rows from a select query
- **mysqli_fetch_row(result)**
- each call returns the next row as an indexed array

e.g.

```
$result = mysqli_query($conn, "select * from module");
$num = mysqli_num_rows($result);
for($i=1; $i<=$num; $i++)
  {
$row = mysqli_fetch_row($result);
echo $row[0]  . "  "  .  $row[1];
  }
```

Other mysql Functions (2)

- **mysqli_affected_rows(result)**
- used after an INSERT, UPDATE, or DELETE query to return the number of rows affected
- **mysqli_free_result(result)**

  frees memory within a php program
- **die**()

  Use to print message and exit from the current **php** script.

Other mysql Functions (3)

- **mysqli_num_fields(result)**
- returns the name of the table column whose position is given by index (0,1,...)

e.g.

```
$res = mysqli_query($conn, 'select  *  from module');
$numfields = mysqli_num_fields($res);
echo "number of fields in module is " . $numfields;
```

Other mysql Functions (4)

- **mysqli_fetch_array(result)**
- returns row information as an array

e.g.

```
$result = mysqli_query($conn, "select * from module");
while($row = mysqli_fetch_array($result))
  {
echo $row['modulecode']  ." "  . $row['modulename'];
  }
```

- Function returns TRUE while a row is fetched & FALSE when there are no rows available
- Function will fetch an array as associative or sequential

PHP MySQL Prepared Statements

- A prepared statement is a feature used to execute the same (or similar) SQL statements repeatedly with high efficiency.
- Steps to execute Prepared statements:
  - Prepare: An SQL statement template is created and sent to the database. Certain values are left unspecified, called parameters (labeled "?"). Example: INSERT INTO user VALUES(?, ?, ?)
  - The database parses, compiles, and performs query optimization on the SQL statement template, and stores the result without executing it
  - Execute: At a later time, the application binds the values to the parameters, and the database executes the statement. The application may execute the statement as many times as it wants with different values

Example: Prepared Statement

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
$conn = new mysqli($servername,
$username, $password, $dbname);

if ($conn->connect_error) {
   die("Connection failed: " . $conn-
>connect_error);
}

$stmt = $conn->prepare("INSERT INTO user
(firstname, lastname, email) VALUES (?, ?,
?)");

$stmt->bind_param("sss", $firstname,
$lastname, $email);
```

```
// set parameters and execute
$firstname = “ABC";
$lastname = “DEF";
$email = “abc@example.com";
$stmt->execute();

$firstname = “UVW";
$lastname = “XYZ";
$email = “uvw@example.com";
$stmt->execute();

echo "created successfully";

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

Sample Code 1

```
<?php
$conn = mysqli_connect('localhost', 'root', '') or die ('No connection');
mysqli_select_db($conn, 'projemp') or die ('DB will not open');
$query = "SELECT eno, ename, salary from emp order by ename asc";
$result = mysqli_query($conn, $query) or die("Invalid query");
$num = mysqli_num_rows($result);
echo "<table border='1'><tr><th>eno</th><th>ename</th>
            <th>salary</th></tr>";
for($i=0;  $i<$num;  $i++)   {
      $row = mysqli_fetch_row($result);
      echo "<tr><td>" . $row[0] . "</td><td>" . $row[1];
      echo "</td><td>" . $row[2] . "</td></tr>";
  }
echo "</table>";
mysqli_close($conn);
?>
```

For Loop

```
//counts number of rows in $result
$num = mysqli_num_rows($result);

for($i=0;  $i<$num;  $i++)   {     // goes round  once for each row
      $row = mysqli_fetch_row($result);
                                          //  puts next row into $row array
      echo "<tr><td>" . $row[0] . "</td><td>" . $row[1];
      echo "</td><td>" . $row[2] . "</td></tr>";
}   // prints the first, second & third column of the current row
    // as sequential array values …  0,1,2
```

Sample Code 2

```
<?php
$conn = mysqli_connect('localhost', 'root', '') or die ('No connection');
mysqli_select_db($conn,'projemp') or die (' DB will not open');
$query = "SELECT eno, ename, salary from emp order by ename desc";
$result = mysqli_query($conn, $query) or die('Invalid query');
echo  "<table border='1'><tr><th>eno</th><th>ename</th>
            <th>salary</th></tr>";
while($row = mysqli_fetch_array($result))
 {
 echo "<tr><td>" . $row['eno'] . "</td><td>" . $row['ename'];
 echo "</td><td>" . $row['salary'] . "</td></tr>";
 }
echo "</table>";
mysqli_close($conn);
?>
```

While Loop

```
// while there is a new row to fetch
while($row = mysqli_fetch_array($result))
  {
  echo "<tr><td>" . $row['eno'] . "</td><td>" . $row['ename'];
  echo "</td><td>" . $row['salary'] . "</td></tr>";
  }   // print the columns using associative array index values
      //   'eno', 'ename'  & 'salary'
```

Next Step

- Having connected to mysql (mysqli_connect)
- Selected a database (mysqli_select_db)
- Run a query (mysqli_query)
- Process the result (mysqli_fetch_row,
  mysqli_fetch_array)

Adding variable to a query string

$value=$_POST["valuelist"];

$query = "SELECT * from emp where salary > " . $value;

[Whatever value is entered in the html front end is posted and plugged into the query string]

Where the where clause uses a text attribute inverted commas must be placed round the value

e.g. SELECT * FROM emp WHERE ename = ‘ABC’

$query = "SELECT * from emp where ename = '" . $value . "'";

[Note the single quotes placed round the variable]

Text Box

In an html form:

Enter a salary value: <input type = "text" name = "textvalue">

Backend php file dbconnect1.php:

<?php

$sal = $_POST["textvalue"];

……

$query = "SELECT * from emp where salary > " . $sal;

?>

dbconnect1.php

```
<?php
$sal=$_POST["textvalue"];   // receives posted value entered in textbox  called textvalue


$conn = mysqli_connect('localhost', 'root', '', 'labuser') or die ('No connection');

$query = "SELECT eno, ename, salary from emp where salary > " . $sal;
$result = mysqli_query($conn, $query) or die("Invalid query");

echo "<table border='1'><tr><th>eno</th><th>ename</th><th>salary</th></tr>";

while ($row = mysqli_fetch_row($result))  {
    echo "<tr><td>" . $row[0] . "</td><td>" . $row[1];
    echo "</td><td>" . $row[2] . "</td></tr>";
 }
echo  "</table>";

mysqli_close($conn);
?>
```

Select Box

In an html form:

```
<select name="valuelist">
<option value="24000">24000</option>
<option value="28000">28000</option>
<option value="32000">32000</option>
</select>
```

- Chosen valuelist option value posted to backend php file (dbconnect2.php)

```
$sal = $_POST["valuelist"];

$query = "SELECT * from emp where salary > " . $sal;
```

dbconnect2.php

```
<?php
$sal = $_POST["valuelist"];   // receives posted value option from selectbox valuelist


$conn = mysqli_connect('localhost', 'labuser', 'mcadb2020') or die ('No connection');

mysqli_select_db($conn, 'labuser') or die (' test will not open');

$query = "SELECT eno, ename, salary from emp where salary > " . $sal;
$result = mysqli_query($conn, $query) or die("Invalid query");

echo "<table border='1'><tr><th>eno</th><th>ename</th><th>salary</th></tr>";

while ($row = mysqli_fetch_row($result))  {
    echo "<tr><td>" . $row[0] . "</td><td>" . $row[1];
    echo "</td><td>" . $row[2] . "</td></tr>";
 }
echo  "</table>";

mysqli_close($conn);
?>
```

CRUD

- Create – Insert Query
- Read (Retrieve) – Select Query
- Update – Update Query
- Delete – Delete Query

CREATE

- **INSERT** is used to create a new record in the database
- Example
  - INSERT into user VALUES (1,‘ABC’, ‘IJK’);

READ

- SELECT is used to retrieve a record in the database
- Example
  - SELECT * FROM user;
  - SELECT username, password FROM user WHERE ID = 1

UPDATE

- UPDATE is used to change record(s) in the database
- Example
  - UPDATE users SET username = ‘ABC’ WHERE ID = 1
  - UPDATE users SET username = ‘ABC’ WHERE username = ‘DEF’

DELETE

- DELETE is used to remove records from the database
- Example
  - DELETE FROM users WHERE ID = 1

PHP include and require Statements

- It is possible to insert the content of one PHP file into another PHP file, with the include or require statement.
- Syntax
  - include '*filename*';
  - require '*filename*';

Example

```
<html>
<body>

    <h1>Welcome to my home page!</h1>
    <p>Some text.</p>
    <p>Some more text.</p>
    <?php include 'footer.php';?>

</body>
</html>
```

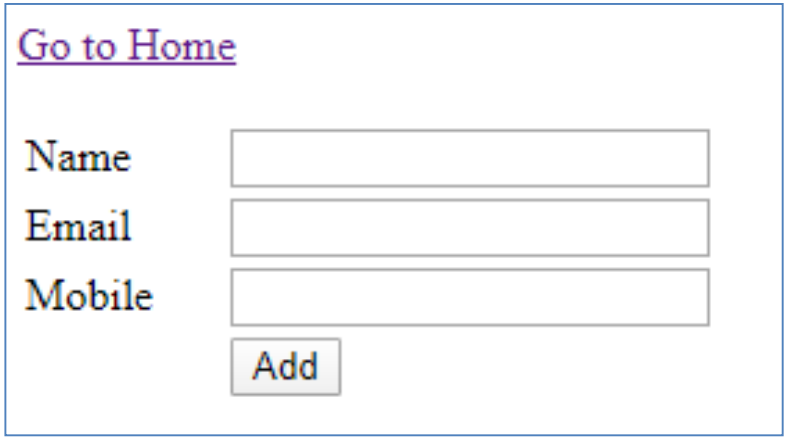
Cont..

- Difference between include and require
- When a file is included with the include statement and PHP cannot find it, the script will continue to execute
- When we using the require statement, the statement will not be executed because the script execution dies after the require statement

```
<!DOCTYPE html>
<html>
<body>

    <h1>Welcome to my home page!</h1>
    <?php require 'noFileExists.php';
    echo "I have a $color $car.";
    ?>

</body>
</html>
```

Example of CRUD: Code for CREATE

```
<?php
        if(isset($_POST['Submit'])) {
                $name = $_POST['name'];
                $email = $_POST['email'];
                $mobile = $_POST['mobile'];
                include_once("config.php");

$result = mysqli_query($mysqli, "INSERT INTO
users(name, email, mobile)
VALUES('$name','$email','$mobile')");

echo "User added successfully.
<a href='index.php'>View Users</a>";
        }
        ?>
```



*Note: The **isset**() function is an inbuilt function in **PHP** which checks whether a variable is set and is not NULL*

Example of CRUD: Code for READ

```
<?php
include_once("config.php");
$result = mysqli_query($mysqli, "SELECT * FROM users ORDER BY id DESC");
?>
```

Add New User

Name	Mobile	Email	Update
ABCDEF	1234567890	abc@example.com	Edit | Delete

Example of CRUD: Code for UPDATE

```
<?php
include_once("config.php");
if(isset($_POST['update']))
{
        $id = $_POST['id'];
        $name=$_POST['name'];
        $mobile=$_POST['mobile'];
        $email=$_POST['email'];
        $result =
mysqli_query($mysqli, "UPDATE users
SET
name='$name',email='$email',mobile='$
mobile' WHERE id=$id");
        header("Location: index.php");
}
?>
```

```
<?php
$id = $_GET['id'];
$result = mysqli_query($mysqli, "SELECT
* FROM users WHERE id=$id");
while($user_data =
mysqli_fetch_array($result))
{
        $name = $user_data['name'];
        $email = $user_data['email'];
        $mobile =
$user_data['mobile'];
}
?>
```

Example of CRUD: Code for DELETE

```
<?php
include_once("config.php");
$id = $_GET['id'];
$result = mysqli_query($mysqli, "DELETE FROM users WHERE id=$id");
header("Location:index.php");
?>
```